TP 3 : LU solver

Exercise 1 : Gaussian elimination

Question 1.1 Write a function `void LUFactorize(DenseMatrix& mat)` that turns a matrix `mat` into its LU factorization without any pivoting.

Question 1.2 Write a function `void LUFactorize(DenseMatrix& mat, std::vector<int>& pivot)` that turns a matrix `mat` into its LU factorization with column pivoting, storing the column-wise choice of pivots into `pivot`. The vector `pivot` will model a permutation of the set $\{0, \ldots, n-1\}$ where n stands for the size of `mat`.

Exercise 2 : LU solver class

Question 2.1 Write a class `LUSolver` that models a direct solver based on the LU factorization of some matrix. This class shall comply with the following specifications :

Data members :

- `int nr` : the number of rows
- `int nc` : the number of columns
- `DenseMatrix` LU : a matrix of size `nr`×`nc` to be used as a container for the storage of the LU factorization i.e. both L and U in the same container.

Functions/member functions :

- a constructor `LUSolver(const DenseMatrix& m)` intializing `nr`,`nc` and loading the LU factorization of the matrix `m` into `LU`.
- a copy constructor
- a copy assignment operator `=`
- operator `( , )` that takes a pair of integers (j, k) and returns the (r-valued) coefficient located at the j-th row and k-th column. This operator shall provide read only access to the coefficients of the factorization.
- output stream operator `<<`

- member-function `std::vector<double> Solve(const std::vector<double>& b)` that solves the corresponding upper triangular linear system with `b` as right-hand side, and returning the solution.
- friend function `int NbRow(const DenseMatrix&)` that returns the number of rows
- friend function `int NbCol(const DenseMatrix&)` that returns the number of columns

Question 2.2 Modify the class `LUSolver` so as to incorporate column pivoting.