4M053 : Grands systèmes linéaires
M.-S.Dupuy & X.Claeys

Projet

This project must be conducted individualy. It must be uploaded on the moodle page no later than Sunday the 1rst of January 2023 at 22 :00.

Your work shall take the form of a .zip file containing the following items :
— the source code of your work,
— a readme file sumarizing the content of your work and the way to use your source code.

You are expected to devise a test protocole upon your own intiative so as to make sure that your work is reliable, bug-free and conforms with the subject below. During the examination, you will be asked to report on your test protocole.

The goal of this project is to devise a GMRes iterative solver. It is assumed that TP1, TP2 and TP3 have been completed. In particular we assume available a class `MapMatrix` properly templated so that `MapMatrix<std::complex<double>>` can be used to model complex valued sparse matrices. We also assume that a class `DenseMatrix` is available in accordance with TP1. In the following, we shall refer to the type alias `typedef std::complex<double> Cplx;`

Question 1 : complex valued LU solver

Modify the code of the class `LUSolver` so that the type of the coefficients be a template parameter `value_t`. This class should be modified so as to provide a LU solver for complex valued dense matrices `DenseMatrix<Cplx>` i.e. for the special case where `value_t = Cplx`. In particular it should offer a member-function `std::vector<value_t> Solve(const std::vector<value_t>&)`.

Question 2 : dense least square

We are now interested in the equation $\mathrm{A}\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{b} \in \mathbb{C}^n$ and $\mathrm{A} \in \mathbb{C}^{n\times m}$ is a densely populated matrix satisfying $\ker(\mathrm{A}) = \{0\}$ and $n \geq m$ with the possibility that $n > m$. Hence the matrix A is not necessarily square and the equation under consideration does systematically admit a solution. We wish to compute $\boldsymbol{x} \in \mathbb{C}^m$ satisfying

$$|\mathrm{A}\boldsymbol{x} - \boldsymbol{b}| = \min_{\boldsymbol{y}\in\mathbb{C}^m} |\mathrm{A}\boldsymbol{y} - \boldsymbol{b}| \tag{1}$$

A possible approach consists in minimizing the quadratic functional $\Phi : \mathbb{C}^m \to \mathbb{R}_+$ defined by $\Phi(\boldsymbol{x}) := \frac{1}{2}|\mathrm{A}\boldsymbol{x} - \boldsymbol{b}|^2$, and writing that the gradient of this functional vanishes at the optimum : $\nabla\Phi(\boldsymbol{x}) = 0$.

Let us denote $\mathrm{A}^* := \overline{\mathrm{A}}^\top$. A straightforward calculus shows that $\nabla\Phi(\boldsymbol{x}) = \mathrm{A}^*\mathrm{A}\boldsymbol{x} - \mathrm{A}^*\boldsymbol{b}$ so that Problem (1) is reduced to the equation $\mathrm{A}^*\mathrm{A}\boldsymbol{x} = \mathrm{A}^*\boldsymbol{b}$. The method of the normal equation then consists in solving the linear system $\mathrm{A}^*\mathrm{A}\boldsymbol{x} = \mathrm{A}^*\boldsymbol{b}$ where $\mathrm{A}^*\mathrm{A} \in \mathbb{C}^{m\times m}$ is now a square symetric positive definite matrix.

Write a function `NormalSolve(const DenseMatrix<Cplx>& A, std::vector<Cplx>& x,const vector<Cplx>& b)` that takes `A` and `b` as input, applies the method of the normal equation to solve the equation $\mathrm{A}\boldsymbol{x} = \boldsymbol{b}$, assuming that $\ker(\mathrm{A}) = \{0\}$, and stores the result in `x`.

Question 3

Implement a function GMResSolve(const MapMatrix<Cplx>& A, std::vector<Cplx>& x, const std::vector<Cplx>& b, int restart, double tol, int maxit) that models a *restarted* GMRes method for solving the linear system $\mathrm{A}\boldsymbol{x} = \boldsymbol{b}$. Here the matrix A is assumed complex valued and invertible, the parameter restart refers to the dimension of the Krylov subspaces, and the process should be stopped whenever $|\boldsymbol{b} - \mathrm{A}\boldsymbol{x}|/|\boldsymbol{b}| <$ tol or after maxit iterations. To deal with the least square problem related to Arnoldi matrices, you can either use the method of the normal equation and the function NormalSolve of the previous question, or use a QR decomposition based on successive Givens rotations.

Question 4

Together with this subject, you are supplied with a .zip archive containing files named matrix_j.txt with j=1,...,5, each one of which contains data of a sparse matrix under the following format

```
nr   nc
j1   k1   v1_real   v1_imag
j2   k2   v2_real   v2_imag
 ⋮    ⋮    ⋮
```

where nr,nc refer to the number of rows and columns of the matrix, vl_real and vl_imag are the real an imaginary part of a complex value v_l =vl_real + i*vl_imag, and the triples j_l, k_l, v_l are the triples (row position, column position, value) involved in the coordinate format.

The archive also contains files named rhs_j.txt with j=1,...,5 that contains the coefficients of vectors to be considered as right hand sides of linear systems. Each of these files describes a complex valued vector $\boldsymbol{b} = (b_1, \dots, b_n)$ with the format

```
b1_real   b1_imag
b2_real   b2_imag
  ⋮         ⋮
bn_real   bn_imag
```

consisting in one column storing the real parts of the coefficients b_j, and another column storing the imaginary parts of the coefficients b_j.

For each of the 5 linear system provided, print a picture (in pdf, png, jpg, .eps or whatever format) representing the convergence history (i.e. a plot of the quadratic norm of the relative residual versus the iteration count) of restarted GMRes with tol = 1e-6, maxit = 1e4 and several values of restart.