Numerical Linear Algebra

A.A. 2022/2023

Lecturer: Prof. P. F. Antonietti TA: Dott. M. Botti

Written test 04/09/2023

First Name: .... Last Name: ....

This exam has 2 questions, with subparts. You have **2 hours** to complete the exam. There are a total of 30 points available (sufficiency at 18 points). You cannot consult any notes, books or aids of any kind except for the codes implemented during the lab sessions. Write answers legibly in the additional sheets provided, and show all of your work. Please **write your name** on the exam itself and on the extra sheets. Concerning the implementations using Eigen, upload only the `.cpp` main files. For the exercises requiring LIS, report the bash commands used to perform the computations in a unique `.txt` file. **Upload the files** following the received instructions.

Exercise 1

1. Consider the following problem: find $\boldsymbol{x} \in \mathbb{R}^n$, $A\boldsymbol{x} = \boldsymbol{b}$, where $A \in \mathbb{R}^{n\times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$ are given. State under which conditions the mathematical problem is well posed.

2. Describe the general form of a linear iterative method for the approximate solution of $A\boldsymbol{x} = \boldsymbol{b}$ and describe the stopping criteria.

3. State the necessary and sufficient condition for convergence.

4. Describe the Generalized Minimal Residual Method (GMRES). Recall the interpretation of the scheme as a Krylov subspace method and the main theoretical results.

5. Download the matrix `Aex1.mtx` from the Exam folder in webeep and save it on the `/shared-folder/iter_sol++` directory. Load the matrix in a new file called `exer1.cpp` using the `unsupported/Eigen/SparseExtra` module and check if the matrix is symmetric. Report on the sheet $\|A\|$ where $\|\cdot\|$ denotes the Euclidean norm.

```
Matrix size:500 X 500
Non zero entries:7708
Norm of A: 105.193
Norm of skew-symmetric part: 47.9613
The matrix is non-symmetric since the norm of the skew-symmetric part of A is
positive with magnitude comparable to the one of the norm of A.
```

6. Define the `Eigen` vector $\boldsymbol{b} = (1, 1, \ldots, 1)^{\mathrm{T}}$ and find the approximate solution $\boldsymbol{x}_j$ of $A\boldsymbol{x} = \boldsymbol{b}$ using the Jacobi method (implemented in the `jacobi.hpp` template). Fix a maximum number of iterations which is sufficient to reduce the (relative) residual below than 10^{-5} and take $\boldsymbol{x}_0 = \boldsymbol{b}$ as initial guess. Report on the sheet the iteration counts and $\|\boldsymbol{x}_j\|$.

```
Jacobi method
iterations performed 14
tolerance achieved 9.343e-06
norm of computed sol 7.07627
```

7. Compute the approximate solution $\boldsymbol{x}_g$ of $A\boldsymbol{x} = \boldsymbol{b}$ obtained using the GMRES method without *restart*. Fix a maximum number of iterations which is sufficient to reduce the residual below than 10^{-10} considering $\boldsymbol{x}_j$ (computed in the previous point) as initial guess. Use the `Eigen` diagonal preconditioner. Report the iteration counts and $\|\boldsymbol{x}_j - \boldsymbol{x}_g\|$.

```
GMRES method
iterations performed: 14
tolerance achieved  : 3.52916e-11
distance btwn 2 sols: 3.90184e-05
```

8. Repeat the previous points using the iterative solvers available in the LIS library. First, compute the approximate solution obtained with the Jacobi method prescribing a tolerance of 10^{-5}. Then, using the GMRES solver compute the approximate solution up to a tolerance of 10^{-10}. Find a preconditioning strategy that yield a decrease in the number of required iterations with respect to the GMRES method without preconditioning. Report on the sheet the iteration counts and the relative residual at the last iteration.

```
mpicc -DUSE_MPI -I${mkLisInc} -L${mkLisLib} -llis test1.c -o test1

mpirun -n 4 ./test1 Aex1.mtx 1 sol.mtx hist.txt -i jacobi -tol 1.e-5
--> Jacobi: number of iterations = 14, relative residual    = 9.603771e-06

mpirun -n 4 ./test1 Aex1.mtx 1 sol.mtx hist.txt -i gmres -tol 1.e-10
--> GMRES: number of iterations = 17, relative residual    = 2.509352e-11

mpirun -n 4 ./test1 Aex1.mtx 1 sol.mtx hist.txt -i gmres -tol 1.e-10
-p ssor -ssor_omega 1.4
--> GMRES: number of iterations = 12, relative residual    = 2.682046e-11
```

Full implementation of Exercise 1

```
#include <iostream>
#include <Eigen/SparseCore>
#include <Eigen/IterativeLinearSolvers>
#include <unsupported/Eigen/SparseExtra>
#include "gmres.hpp"
#include "jacobi.hpp"

int main(int argc, char** argv)
{
  using namespace LinearAlgebra;
  // Some useful alias
  using SpMat=Eigen::SparseMatrix<double>;
  using SpVec=Eigen::VectorXd;
  // Load matrix
  SpMat A;
  Eigen::loadMarket(A, "Aex1.mtx");

  // Check matrix properties
  std::cout << "Matrix size:" << A.rows() << " X " << A.cols() << std::endl;
  std::cout << "Non zero entries:" << A.nonZeros() << std::endl;
  SpMat B = SpMat(A.transpose()) - A;
  double Anorm = A.norm();
  std::cout << "Norm of A: " << Anorm << std::endl;
  std::cout << "Norm of skew-symmetric part: " << B.norm() << std::endl;

  double tol = 1.e-5;              // Convergence tolerance
  int result, maxit = 1000;        // Maximum iterations
  SpVec b = SpVec::Ones(A.rows());
  std::cout << "norm of rhs: " << b.norm() << std::endl;
  SpVec x(A.rows());
  Eigen::DiagonalPreconditioner<double> D(A);

  // Jacobi method
  result = Jacobi(A, x, b, D, maxit, tol);
  std::cout << "Jacobi method " << std::endl;
  std::cout << "iterations performed " << maxit << std::endl;
  std::cout << "tolerance achieved " << tol << std::endl;
  std::cout << "norm of computed sol " << x.norm() << std::endl;

  // GMRES method with diagonal precond
  SpVec xg(A.rows());
  xg = x; tol = 1.e-10; maxit = 500;
  int restart = maxit;  // set restart = maxit to avoid restart
  result = GMRES(A, x, b, D, restart, maxit, tol);
  std::cout << "GMRES method " << std::endl;
  std::cout << "iterations performed: " << maxit        << std::endl;
  std::cout << "tolerance achieved  : " << tol          << std::endl;
  std::cout << "distance btwn 2 sols: " << (x - xg).norm() << std::endl;

  return result;
}
```

Exercise 2

1. Consider the rectangular linear system $A\boldsymbol{x} = \boldsymbol{b}$, where A is an $m \times n$ matrix, $m \geq n$. Provide the definition of the solution in the least-square sense and state under which condition the problem is well posed.

2. Describe the QR factorization of the rectangular matrix A.

3. Discuss how the QR factorization can be employed to solve the above linear system in the least-square sense.

4. Download the matrix `Aex2.mtx` from the Exam folder in webeep and save it on the `/shared-folder/iter_sol++` directory. Load the matrix in a new file called `exer2.cpp` using the `unsupported/Eigen/SparseExtra` module. Report on the sheet $\|A^T A\|$.

```
Matrix size:800 X 400, Non zero entries:8431
Norm of A: 70.8894, Norm of A^T*A: 335.01
```

5. Define an `Eigen` vector $\boldsymbol{b} = A\boldsymbol{x}^*$, where $\boldsymbol{x}^* = (1, 1, \ldots, 1)^{\mathrm{T}}$. Report on the sheet $\|\boldsymbol{b}\|$.

```
norm of rhs: 31.7075
```

6. Use the `SparseQR` solver available in the `Eigen` library to compute the approximate solution of the least-square problem associated to $A\boldsymbol{x} = \boldsymbol{b}$. Report on the sheet the Euclidean norm of the error $\|\boldsymbol{x}_{SQR} - \boldsymbol{x}^*\|$, where $\boldsymbol{x}_{SQR}$ is the obtained approximate solution.

```
Solution with Eigen QR:
effective error: 4.20802e-14
```

7. Use the `LeastSquaresConjugateGradient` solver available in the `Eigen` library to compute the approximate solution of the previous least-square problem up to a tolerance of 10^{-10}. Report on the sheet the iteration counts and the error norm $\|\boldsymbol{x}_{LSCQ} - \boldsymbol{x}^*\|$, where $\boldsymbol{x}_{LSCQ}$ is the obtained approximate solution.

```
Solution with Eigen LSCG:
#iterations:     97
relative residual: 9.95923e-11
effective error: 5.74656e-09
```

8. Export the matrix $A^T A$ in the matrix market format (save it as `AtA.mtx`) and move it to the `lis-2.0.34/test` folder. Using the Conjugate Gradient iterative solver available in LIS compute the approximate solution of the linear system $A^T A\boldsymbol{x} = A^T\boldsymbol{b}$ up to a tolerance of 10^{-10}. Report the iteration counts and the relative residual at the last iteration.

```
mpirun -n 4 ./test1 AtA.mtx 2 sol.mtx hist.txt -i cg -tol 1.e-10
--> CG: number of iterations = 114, relative residual  = 9.222078e-11
```

Full implementation of Exercise 2

```
#include <Eigen/SparseCore>
#include <Eigen/SparseQR>
#include <iostream>
#include <string>
#include <Eigen/IterativeLinearSolvers>
#include <unsupported/Eigen/SparseExtra>

int main(int argc, char** argv){
  using namespace Eigen;

  // Some useful alias
  using SpMat = SparseMatrix<double>;
  using SpVec = VectorXd;

  // Load matrix
  SpMat A;
  Eigen::loadMarket(A, "Aex2.mtx");
```

```cpp
  // Check matrix properties
  std::cout << "Matrix size:" << A.rows() << " X " << A.cols() << std::endl;
  std::cout << "Non zero entries:" << A.nonZeros() << std::endl;
  std::cout << "Norm of A: " << A.norm() << std::endl;
  SpMat A_square = A.transpose()*A;
  std::cout << "Norm of A^T*A: " << A_square.norm() << std::endl;

  // Create Rhs b
  SpVec e = SpVec::Ones(A.cols());
  SpVec b = A*e;
  std::cout << "norm of rhs: " << b.norm() << std::endl;
  SpVec x(A.cols());

  // solve with Eigen QR factorization
  Eigen::SparseQR<Eigen::SparseMatrix<double>, COLAMDOrdering<int>> solver;
  solver.compute(A);
  if(solver.info()!=Eigen::Success) {
      std::cout << "cannot factorize the matrix" << std::endl;
      return 0;
  }
  x = solver.solve(b);
  std::cout << "Solution with Eigen QR:" << std::endl;
  std::cout << "effective error: "<<(x-e).norm()<< std::endl;

  // solve with Eigen LeastSquareConjugateGradient solver
  double tol = 1.e-10;            // Convergence tolerance
  int maxit = 1000;               // Maximum iterations

  LeastSquaresConjugateGradient<SparseMatrix<double> > lscg;
  lscg.setMaxIterations(maxit);
  lscg.setTolerance(tol);
  lscg.compute(A);
  x = lscg.solve(b);
  std::cout << "Solution with Eigen LSCG:" << std::endl;
  std::cout << "#iterations:     " << lscg.iterations() << std::endl;
  std::cout << "relative residual: " << lscg.error()    << std::endl;
  std::cout << "effective error: " << (x-e).norm()      << std::endl;

  // Export matrix
  std::string matrixFileOut("./AtA.mtx");
  saveMarket(A_square, matrixFileOut);
  return 1;
}
```