Numerical Linear Algebra
A.A. 2022/2023

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

Exam simulation

Exercise 1

Let A be an $n \times n$ matrix such that $A = B + C$, with

$$B = \begin{pmatrix} 2 & 1 & 0 & 0 & \dots & 0 \\ 1 & 4 & 2 & 0 & \dots & 0 \\ 0 & 2 & 6 & 3 & \dots & 0 \\ 0 & 0 & 3 & 8 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & n-1 \\ 0 & 0 & 0 & \dots & n-1 & 2n \end{pmatrix}; \qquad C = \begin{pmatrix} 0 & 0 & \dots & 0 & 0 & -1 \\ 0 & 0 & \dots & 0 & -2 & 0 \\ \vdots & \vdots & \iddots & -3 & 0 & 0 \\ 0 & 0 & \iddots & \iddots & \vdots & \vdots \\ 0 & -n+1 & 0 & \dots & 0 & 0 \\ -n & 0 & 0 & \dots & 0 & 0 \end{pmatrix},$$

Consider the problem of approximating the (unique) solution of the linear system

$$A\boldsymbol{x}^* = \boldsymbol{b},$$

where $\boldsymbol{b}$ is a given $n \times 1$ vector.

1. Describe the Jacobi and Gauss Seidel methods, introduce the algorithms, and discuss the main applicability and convergence results.
2. Describe the Generalized Mininal Residual method (GMRES) algorithm, introduce the main idea, the notation employed, and characterize it in the framework of Krylov solvers.
3. Propose and motive a possibile preconditioner to accelerate the convergence of the above mentioned methods. Motive the choice.
4. Let $n = 1000$. Define the matrix A using the `Eigen::SparseMatrix<double>` type.
5. Define an `Eigen` vector $\boldsymbol{b} = A\boldsymbol{x}^*$, where $\boldsymbol{x}^* = (1, 1, \dots, 1)^{\mathrm{T}}$.
6. Solve the linear system $A\boldsymbol{x}^* = \boldsymbol{b}$ using the GMRES implemented in the `gmres.hpp` template. Fix a maximum number of iterations equal to the linear system's size and assume a tolerance of 10^{-12} for the final residual. Use the diagonal preconditioner provided by the `Eigen::DiagonalPreconditioner<double>` function.
7. Compare the GMRES method with restart (`restart`$= 50$) and without restart. Comment the obtained results.

Solution:

```
#include <cstdlib>                          // System includes
#include <iostream>
#include <Eigen/SparseCore>
#include <Eigen/IterativeLinearSolvers>
```

```cpp
#include "gmres.hpp"

int main(int argc, char** argv)
{
  using namespace LinearAlgebra;
  using SpMat=Eigen::SparseMatrix<double>;
  using SpVec=Eigen::VectorXd;

  int n = 1000;
  SpMat A(n,n);                        // define matrix
  for (int i=0; i<n; i++) {
      A.coeffRef(i, i) = 2.0*(i+1);
      A.coeffRef(i,n-i-1) = -i;
      if(i>0) A.coeffRef(i, i-1) += i;
      if(i<n-1) A.coeffRef(i, i+1) += i+1;
  }
  std::cout << "Matrix size: " << A.rows() << "X" << A.cols() << std::endl;
  std::cout << "Non zero entries: " << A.nonZeros() << std::endl;

   // Create Rhs b
  SpVec e = SpVec::Ones(A.rows());
  SpVec b = A*e;
  SpVec x(A.rows());

  // Solve with GMRES method with restart
  double tol = 1.e-12;                  // Convergence tolerance
  int result, maxit = 1000;            // Maximum iterations
  int restart = 50;                        // Restart gmres
  Eigen::DiagonalPreconditioner<double> D(A); // Create diagonal preconditioner

  result = GMRES(A, x, b, D, restart, maxit, tol);
  std::cout << "GMRES with restart "    << std::endl;
  std::cout << "iterations performed: " << maxit << std::endl;
  std::cout << "tolerance achieved  : " << tol << std::endl;
  std::cout << "Error:                " << (x-e).norm()<< std::endl;

  // Solve with GMRES method without restart
  x=0*x; restart = 1000; maxit = 1000; tol = 1.e-12;
  result = GMRES(A, x, b, D, restart, maxit, tol);
  std::cout << "GMRES without restart " << std::endl;
  std::cout << "iterations performed: " << maxit << std::endl;
  std::cout << "tolerance achieved  : " << tol << std::endl;
  std::cout << "Error norm: "<<(x-e).norm()<< std::endl;

  return result;
}
```

Exercise 2

Let A be an $n \times n$ large, sparse and symmetric matrix and consider the following eigenvalue problem

$$A\boldsymbol{x} = \lambda \boldsymbol{x}$$

1. Describe the Lanczos Iteration Algorithm for computing the extremal eigenvalues and corresponding eigenvectors of A.

2. Introduce the notation employed and discuss the main properties of the Lanczos method.

3. Using the Library of Iterative Solvers for linear systems (LIS), report the full list of bash commands required to perform the computations here below in a `.txt` file. Compile the LIS script using `mpi` and run the LIS executables using 4 processors.

4. Using `wget` and `gzip`, download and unzip the matrix `gr_30_30.mtx` from the matrix market website (https://math.nist.gov/MatrixMarket/).

5. Compute the largest (in absolute value) eigenvalue of the matrix that has been previously downloaded up to a tolerance of order 10^{-8}. Report the computed values.

6. Compute the eight smallest (in absolute value) eigenvalues of the `gr_30_30.mtx` matrix and save the corresponding eigenvectors in a `.mtx` file. Explore different iterative methods and preconditioners (at least 3 alternative strategies) in order to achieve a precision smaller than 10^{-10}. Compare and comment the results.

Solution:

```
wget https://math.nist.gov/pub/MatrixMarket2/Harwell-Boeing/laplace/gr_30_30.mtx.gz

gzip -dk gr_30_30.mtx.gz

mpicc -DUSE_MPI -I${mkLisInc} -L${mkLisLib} -llis etest1.c -o eigen1

mpirun -n 4 ./eigen1 gr_30_30.mtx eigvec.txt hist.txt -e pi -emaxiter 5000 -etol 1.0e-8

mpicc -DUSE_MPI -I${mkLisInc} -L${mkLisLib} -llis etest5.c -o eigen2

mpirun -n 4 ./eigen2 gr_30_30.mtx evals.mtx eigvecs.mtx res.txt iters.txt
-ss 8 -e si -p jacobi -etol 1.0e-10 -emaxiter 2000

mpirun -n 4 ./eigen2 gr_30_30.mtx evals.mtx eigvecs.mtx res.txt iters.txt
-e si -ie ii -ss 8 -i cg -p ilu ilu_fill 3 -etol 1.0e-10

mpirun -n 4 ./eigen2 gr_30_30.mtx evals.mtx eigvecs.mtx res.txt iters.txt
-e si -ie ii -ss 8 -i bicgstab -p ssor -etol 1.0e-10
```