Numerical Linear Algebra
TMA265/MMA600
Computer exercise 5:
Solution of Helmholtz equation

Larisa Beilina, larisa@chalmers.se

Computer exercise 5 (3 b.p.).

0.1 Solution of Helmholtz equation

This exercise can be viewed as part of the Master's project "Efficient implementation of Helmholtz equation with applications in medical imaging", see Master's projects homepage for description of this project or go to the link

`http://www.math.chalmers.se/Math/Grundutb/CTH/tma265/1617/BOOK/MasterProject_Helmholtz.pdf`

Solve the Helmholtz equation

$$\begin{aligned} \Delta u(x,\omega) + \omega^2 \varepsilon'(x) u(x,\omega) &= i\omega J, \\ \lim_{|x|\to\infty} u(x,\omega) &= 0. \end{aligned} \tag{0.1}$$

in two dimensions using C++/PETSC. Here, $\varepsilon'(x)$ is the spatially distributed complex dielectric function which can be expressed as

$$\varepsilon'(x) = \varepsilon_r(x) \frac{1}{c^2} - i\mu_0 \frac{\sigma(x)}{\omega}, \tag{0.2}$$

where $\varepsilon_r(x) = \varepsilon(x)/\varepsilon_0$ and $\sigma(x)$ are the dimensionless relative dielectric permittivity and electric conductivity functions, respectively, ε_0, μ_0 are the permittivity and permeability of the free space, respectively, and $c = 1/\sqrt{\varepsilon_0 \mu_0}$ is the speed of light in free space, and ω is the angular frequency.

Take appropriate values for ω, ε', J. For example, take

$$\omega = \{40, 60, 80, 100\}, \varepsilon_r = \{2, 4, 6\}; \sigma = \{5, 0.5, 0.05\}, J = 1.$$

Analyze obtained results for different $\omega, \varepsilon_r, \sigma, J$.
Information about PETSc can be found on the link:
`https://www.mcs.anl.gov/petsc/`

Hints:

1. Study Example 12.5 of the course book [1] where is presented solution of the Dirichlet problem for the Poisson's equation on a unit square using different iterative methods implemented in C++/PETSc. C++/PETSc programs for solution of this problem are available for download from the course homepage: go to the link of the book [1] and click to "GitHub Page with MATLAB® Source Codes" on the bottom of this page or go to the link

   `https://github.com/springer-math/Numerical_Linear_Algebra_Theory_and_Applications`

   Choose then

   `PETSC_code`

   The different iterative methods are encoded by numbers 1-7 in the main program

```
Main.cpp
```

in the following order:

- 1 - Jacobi's method,
- 2 - Gauss-Seidel method,
- 3 - Successive Overrelaxation method (SOR),
- 4 - Conjugate Gradient method,
- 5 - Conjugate Gradient method (Algorithm 12.13),
- 6 - Preconditioned Conjugate Gradient method,
- 7 - Preconditioned Conjugate Gradient method (Algorithm 12.14).

Methods 1-5 use inbuilt PETSc functions, and methods 6,7 implement algorithms 12.13, 12.14 of the book [1], respectively. For example, we can run the program Main.cpp using SOR method as follows:

```
> nohup Main 3 > result.m
```

After running the results will be printed in the file result.m and can be viewed in Matlab using the command

```
surf(result).
```

2. Modify PETSc code of the Example 12.5 of [1] such that the equation (0.1) can be solved. Note that solution of the equation (0.1) is complex. You should include

```
#include <complex>
```

to be able work with complex numbers in C++. For example, below is example of definition of the complex array in C++ and assigning values to the real and imaginary parts:

```
complex<double> *complex2d = new complex<double>[nno];
 double a = 5.4;
  double b = 3.1;

 for (int i=0; i < nno; i++)
 {
   complex2d[i].real() = a;
   complex2d[i].imag() = b;
 }
   delete[] complex2d;
```

Example of the definition of the complex right hand side in PETSc is presented below:

```
PetscScalar right_hand_side(const PetscReal x, const PetscReal y)
{
  PetscReal realpart, imagpart;
  PetscReal   pi = 3.14159265359;
  realpart = pi*sin(2*pi*x)*cos(2*pi*y);
  imagpart = x*x + y*y;
  PetscScalar f(rpart, ipart);
  return f;
}
```

3. Example of Makefile for running C++/PETSc code at Chalmers is presented in Example 12.5 of [1] and can be as follows:

```
PETSC_ARCH=/chalmers/sw/sup64/petsc-3.7.4

include ${PETSC_ARCH}/lib/petsc/conf/variables
include ${PETSC_ARCH}/lib/petsc/conf/rules

CXX=g++
CXXFLAGS=-Wall -Wextra -g -O0 -c -Iinclude -I${PETSC_ARCH}/include
LD=g++
LFLAGS=

OBJECTS=Main.o CG.o Create.o DiscretePoisson2D.o GaussSeidel.o
 Jacobi.o  PCG.o Solver.o SOR.o
Run=Main

all: $(Run)

$(CXX) $(CXXFLAGS) -o $@ $<

$(Run): $(OBJECTS)
$(LD) $(LFLAGS) $(OBJECTS) $(PETSC_LIB) -o $@
```

To compile PETSc with complex numbers you need to write in Makefile:

```
PETSC_ARCH=/chalmers/sw/sup64/petsc-3.7.4c
```

4. Choose the two-dimensional convex computational domain Ω such that $\Omega = [0,1] \times [0,1]$. Choose boundary condition at the boundary of $\partial\Omega$ such that the condition $\lim_{|x| \to \infty} u(x, \omega) = 0$ is satisfied, for example, take $\partial_n u = 0$.

5. Choose the following boundary condition $u(x,\omega) = -\omega g(x,\omega)$, where $g(x,\omega)$ is given by (0.4). More precisely, solve the Helmholtz equation

$$\begin{aligned} \Delta u(x,\omega) + \omega^2 \varepsilon(x) u(x,\omega) &= f(x,\omega), \\ u(x,\omega) &= -\omega g(x,\omega). \end{aligned} \tag{0.3}$$

Take as $g(x, \omega), x = (x_1, x_2)$, the function

$$u(x_1, x_2) = \sin(2\pi x_1)\sin(2\pi x_2) + i x_1(1 - x_1)x_2(1 - x_2) \tag{0.4}$$

which is the exact solution of the equation (0.3) with the right hand side

$$\begin{aligned} f(x_1, x_2) = &-(8\pi^2)\sin(2\pi x_1)\sin(2\pi x_2) - 2ix_1(1 - x_1) - 2ix_2(1 - x_2) \\ &+ \omega^2\varepsilon(x)(\sin(2\pi x_1)\sin(2\pi x_2) + ix_1(1 - x_1)x_2(1 - x_2)) \end{aligned} \tag{0.5}$$

6. Try also the following boundary condition $\partial_n u(x, \omega) = -\omega g(x, \omega)$.
7. Values of c, μ_0, ε_0 in (0.2) are known constants.
   - Vacuum permittivity, sometimes called the electric constant ε_0 and measured in F/m (farad per meter):
     $$\varepsilon_0 \approx 8.85 \cdot 10^{-12}$$
   - The permeability of free space,or the magnetic constant μ_0 measured in H/m (henries per meter):
     $$\mu_0 \approx 12.57 \cdot 10^{-7}$$
   - The speed of light in a free space is given by formula $c = 1/\sqrt{\varepsilon_0\mu_0}$ and is measured in m/c (metres per second):
     $$c \approx 300\ 000\ 000$$

References

[1] L. Beilina, E. Karchevskii, M. Karchevskii, Numerical Linear Algebra: Theory and Applications, Springer, 2017.

[2] Christopher M. Bishop, *Pattern recognition and machine learning*, Springer, 2009.

[3] G. S. Fulcher, Analysis of recent measurements of the viscosity of glasses, *Journal of American Ceramic Society*, `https://doi.org/10.1111/j.1151-2916.1925.tb16731.x`, 1925

[4] Ian Goodfellow, Yoshua Bengio and Aaron Courville, *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`

[5] Miroslav Kurbat, *An Introduction to Machine Learning*, Springer, 2017.