Numerical Linear Algebra. Computer Labs

Notes

- To pass this course you should do **any 2** computer assignments described below.
- You can work in groups by 2 persons.
- Sent final report for every computer assignment with description of your work together with Matlab or C++/PETSc programs to my e-mail before deadline. Report should have description of used techniques, tables and figures confirming your investigations. Analysis of obtained results is necessary to present in section "Numerical examples" and summarized results - in section "Conclusions". You can download latex or pdf-template for report from the course homepage.
- Matlab and C++ programs for examples in the book [1] 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.

Computer exercise 1 (1 b.p.)

Apply Bisection Algorithm (you can find this algorithm in the book [1], see question 8.3, Algorithm 8.11) **Bisection**, to find the roots of the polynomial

$$p(x) = (x-1)^3(x-5)^2(x-7)^3 = 0$$

for different input intervals $x = [x_{left}, x_{right}]$. Use this algorithm to determine the roots **of some of your own polynomial**. Note that in the Bisection Algorithm $p(x)$ should be evaluated using Horner's rule.

Write your own Matlab's program which can use as a template the Matlab program

`Bisection.m`

from the 'GitHub Page with MATLAB® Source Codes. Confirm that changing the input interval for $x = [x_{\text{left}}, x_{\text{right}}]$ slightly changes the computed root drastically. Modify the algorithm to use the relative condition number for polynomial evaluation given by (8.26) of [1]:

$$\text{cond}(p) := \frac{\sum_{i=0}^{d} |c_i x^i|}{\left|\sum_{i=0}^{d} c_i x^i\right|}$$

to stop bisecting when the round-off error in the computed value of $p(x)$ gets so large that its sign cannot be determined. Present your results similarly with results of Figures 8.2, 8.3 in the book [1]. Compare your results with results of these figures.

Hints:

- Study Example 8.3, page 262, of [1].
- Note, that in Horner's rule a_i denote coefficients of the polynomial

$$p(x) = \sum_{i=0}^{d} a_i x^i,$$

where d is the degree of the polynomial. Thus, you need first compute coefficients of the polynomial $p(x)$. For example, exact coefficients of polynomial $p(x) = (x-9)^9$ are:

$$\begin{aligned} a = [&-387420489; 387420489; -172186884; 44641044; \\ &-7440174; 826686; -61236; 2916; -81; 1]. \end{aligned} \tag{1}$$

Use the Matlab function

```
coeffs
```

to compute the coefficients of a polynomial $p(x)$.

- Compute error bound $bp = \triangle$ as in the algorithm 8.6 of [1] for every point $x \in [x_{\text{left}}, x_{\text{right}}]$. Below is example of Matlab's function which can be used to compute this error bound:

```
function [P,bp] = evaluate_polynomial_by_Horners_rule(a,x,eps)
  % Parameters: a contains the coeficients of the plynomial  p(x)
  % P is the value of p(x) at x.
  % eps is the mechine epsilon
  d = numel(a);

  P = a(d);
  bp = abs(a(d));

  for i = d - 1:(-1):1

    P = x*P + a(i);
    bp = abs(x)*bp + abs(a(i));

  end

  %error bound
  bp = 2*(d - 1)*eps*bp;

end
```

Here, eps is the machine epsilon. Use following machine eps in Matlab:

```
 eps = 2.2204460492503131e-16
```

Computer exercise 2 (1 b.p.)

Consider the nonlinear model equation

$$y(T) = A \cdot \exp^{\frac{E}{T-T_0}}$$

presenting one of the models of the viscosity of glasses (see paper G. S. Fulcher, "ANALYSIS OF RECENT MEASUREMENTS OF THE VISCOSITY OF GLASSES" on the course homepage). Here, T is the known temperature, $y(T)$ is the known output data. Determine parameters A, E, T_0 which are positive constants by knowing T and output data $y(T)$.

Hints:

1. Transform first the nonlinear function $y(T)$ to the linear one and formulate then the linear least squares problem. Discretize T by N points and compute discrete values of $y(T)$ as $y_i = y(T_i)$ for the known values of parameters A, E, T_0. Then forget about these parameters (we will call them exact parameters A^*, E^*, T_0^*) and solve the linear least squares problem to recover these exact parameters.

2. You can choose exact parameters A^*, E^*, T_0^* as well as the temperature T as some positive constants. For example, take $E^* = 6 \cdot 10^3, A^* = exp^{-2.64}, T_0^* = 400, T = 750 + 10 * i, i = 1, ..., N$, where N is the number of discretization points. See Table II in the paper G. S. Fulcher, "ANALYSIS OF RECENT MEASUREMENTS OF THE VISCOSITY OF GLASSES" for some other possible choises of these constants.

3. Add random noise δ to data $y(T)$ using the formula

   $$y_\delta(T) = y(T)(1 + \delta\alpha),$$

   where $\alpha \in (-1, 1)$ is randomly distributed number and $\delta \in [0, 1]$ is the noise level. For example, if noise in data is 5%, then $\delta = 0.05$.

4. Solve the linear least squares problem using the method of normal equations, QR and then SVD decompositions. Analyze obtained results by computing the relative errors e_A, e_E, e_{T_0} in the computed parameters depending on the different noise level $\delta \in [0, 1]$ in data $y_\sigma(T)$ for every method.

The relative errors e_A, e_E, e_{T_0} in the computed parameters A, E, T_0 compute as:

$$\begin{aligned} e_A &= \frac{|A - A^*|}{|A^*|}, \\ e_E &= \frac{|E - E^*|}{|E^*|}, \\ e_{T_0} &= \frac{|T_0 - T_0^*|}{|T_0^*|}. \end{aligned} \tag{2}$$

Here, A^*, E^*, T_0^* are exact values and A, E, T_0 are computed one. Present results how relative errors (2) depend on the random noise $\delta \in [0, 1]$ in graphical form and in the corresponding table.

5. Choose different number of discretization points N in the interval for temperature T and present results of computations in graphical form and in the corresponding table. More precisely, present how relative errors (2) depend on the number of measurements N if you solve the linear least squares problem using 3 methods: the method of normal equations, QR and then SVD decomposition.

6. Using results obtained in items 4 and 5, analyze, what is the minimal number of observations N to get reasonable reconstruction of parameters A, E, T_0 within the noise level σ ?

Computer exercise 3 (2 b.p.)

In this exercise we will study different linear and quadratic classifiers: least squares classifier, perceptron learning algorithm, WINNOW algorithm using training sets described below. Implement in MATLAB all these classification algorithms and present decision lines for following training sets:

- I) for randomly distributed data $y_i, i = 1, ..., m$ generated by

  $$-1.2 + 0.5x + y = 0$$

  on the interval $x = [-10, 10]$. Generate random noise δ to data $y(x)$ using the formula

  $$y_\sigma(x) = y(x)(1 + \delta\alpha),$$

  where $\alpha \in (-1, 1)$ is randomly distributed number and $\delta \in [0, 1]$ is the noise level. For example, if noise in data is 5%, then $\delta = 0.05$. Perform different experiments with different number of generated data $m > 0$ which you choose as you want (for example, m=10,100,1000).

- II) Generate your own data and try separate them. You can take experimental data from the link

  https://archive.ics.uci.edu/ml/datasets.html

  Or download *.xlsx file with data for grey seals and classify time-dependent data in this file for length and weight of seals. Another option is classification of weight-dependent data on length and thikness of seals.

Try answer to the following questions:

- Analize what happens with performance of perceptron learning algorithm if we take different learning rates $\eta \in (0, 1]$? For what values of η perceptron learning algorithm is more sensitive and when the iterative process is too slow?

- Analyze which one of the studied classification algorthms perform best and why?

- Try to explain why least squares approach can fail in the case when usual linear classifier is used.

Hints:

1. In this exercise we will assume that we will work in domains with two classes: positive class and negative class. We will assume that each training example $\mathbf{x}$ can have values 0 or 1 and we will label positive examples with $c(\mathbf{x}) = 1$ and negative with $c(\mathbf{x}) = 0$.

2. We will also assume that these classes are linearly separable. These two classes can be separated by a linear function of the form

$$\omega_0 + \omega_1 x + \omega_2 y = 0, \tag{3}$$

where x, y are Cartesian coordinates. Note that the equation (3) can be rewritten for the case of a linear least squares problem as

$$\omega_0 + \omega_1 x = -\omega_2 y \tag{4}$$

or as

$$-\frac{\omega_0}{\omega_2} - \frac{\omega_1}{\omega_2} x = y. \tag{5}$$

3. Suppose that we have measurements $y_i, i = 1, ..., m$ in (5) and our goal is to determine coefficients in (5) from these measurements. We can determine coefficients $\omega_0, \omega_1, \omega_2$ by solving the following least squares problem:

$$\min_{\omega} \|A\omega - y\|_2^2 \tag{6}$$

with $\omega = [\omega_1, \omega_2]^T = [-\frac{\omega_0}{\omega_2}, -\frac{\omega_1}{\omega_2}]^T$, rows in matrix A given by

$$[1, x_k], \quad k = 1, ..., m,$$

and vector $y = [y_1, ..., y_m]^T$.

4. The Perceptron learning algorithm is presented below. Decision line then can be presented in Matlab for already computed weights by Perceptron learning algorithm using the formula (5).

Perceptron learning algorithm

Assume that two classes c($\mathbf{x}$)=1 and c($\mathbf{x}$)=0 are linearly separable.

Step 0. Initialize all weights ω_i in

$$\sum_{i=0}^{n} \omega_i x_i = 0$$

to small random numbers (note $x_0 = 1$). Choose an appropriate learning rate $\eta \in (0, 1]$.

Step 1. For each training example $\mathbf{x} = (x_1, ..., x_n)$ whose class is $c(\mathbf{x})$ do:

- (i) Assign $h(\mathbf{x}) = 1$ if
$$\sum_{i=0}^{n} \omega_i x_i > 0$$
and assign $h(\mathbf{x}) = 0$ otherwise.
- (ii) Update each weight using the formula
$$\omega_i = \omega_i + \eta \cdot [c(\mathbf{x}) - h(\mathbf{x})] \cdot x_i.$$

Step 2. If $c(\mathbf{x}) = h(\mathbf{x})$ for **all training examples** stop, otherwise go to Step 1.

Computer exercise 4 (3 b.p.)

This exercise can be viewed as a training example for Master's project "Efficient implementation of Helmholtz equation with applications in medical imaging", see Master's projects homepage for description of this project.

Solve the Helmholtz equation

$$\begin{aligned} \Delta u(x,\omega) + \omega^2 \varepsilon'(x) u(x,\omega) &= i\omega J, \\ \lim_{|x|\to\infty} u\,(x,\omega) &= 0. \end{aligned} \tag{7}$$

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

$$\varepsilon'(x) = \varepsilon_r(x)\frac{1}{c^2} - i\mu_0\frac{\sigma(x)}{\omega}, \tag{8}$$

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, 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 (7) can be solved. Note that solution of the equation (7) 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 $\Omega\,\Omega = [0,1] \times [0,1]$. 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 some functions in the form of Gaussian exp^{-x^2}.

5. Choose the following boundary condition $u(x,\omega) = -\omega g(x,\omega)$, where $g(x,\omega)$ is given by (10). More precisely, solve the Helmholtz equation

$$\begin{aligned} \Delta u(x,\omega) + \omega^2 \varepsilon(x) u(x,\omega) &= f(x,\omega), \\ u(x,\omega) &= -\omega g(x,\omega). \end{aligned} \tag{9}$$

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

$$u(x_1,x_2) = \sin(2\pi x_1)\sin(2\pi x_2) + i x_1(1-x_1)x_2(1-x_2) \tag{10}$$

which is the exact solution of the equation (9) with the right hand side

$$\begin{aligned} f(x_1,x_2) = &-(8\pi^2)\sin(2\pi x_1)\sin(2\pi x_2) - 2i x_1(1-x_1) - 2i x_2(1-x_2) \\ &+ \omega^2 \varepsilon(x)(\sin(2\pi x_1)\sin(2\pi x_2) + i x_1(1-x_1)x_2(1-x_2)) \end{aligned} \tag{11}$$

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 (8) 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.