Numerical Linear Algebra. Computer Labs

1 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 the date for 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 summarize results in the section "Conclusion". You can download latex-template for report from the course homepage.
- Matlab and C++ programs for examples in the book [2] are available for download from the course homepage: go to the link of the book [2] and click to "GitHub Page with MATLAB Source Codes" on the bottom of this page.

2 Computer exercise 1 (1 b.p.)

Apply Bisection Algorithm (you can find this algorithm in the book [2], 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 these algorithms to determine also 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 program. Confirm that changing the input interval for $x = [x_{left}, x_{right}]$ slightly changes the computed root drastically. Modify the algorithm to use the relative condition number for polynomial evaluation given by (8.26) of [2]:

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

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 Figure 1.3 of the Demmel's book [1] or Fig. 8.2, 8.3 in the book [2]. Compare your results with results of Figure 1.3 of the Demmel's book [1] or results of Fig. 8.2, 8.3 of the book [2].

Hints:

- Study Example 8.3, page 262, of [2].
- 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 [2] for every point $x \in [x_{left}, x_{right}]$. 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. Machine eps in matlab: $eps = 2.2204460492503131e - 16$;

3 Computer exercise 2 (1 b.p.)

Consider the nonlinear equation

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

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 solve then 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 T as any positive constants. For example, take $A^* = 10, E^* = 50, T_0^* = 100, T = T_0^* + 10 * i, i = 1, ..., N$, where N is the number of discretization points.

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

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

   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 are given by:

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

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 relative noise $\delta \in [0, 1]$ in graphical form and in the corresponding table.

5. Choose different number of discretization points N 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 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 should be chosen to get reasonable reconstruction of parameters A, E, T_0 within the noise level σ ?

4 Computer exercise 3 (2 b.p.)

Suppose that the nonlinear model function with known positive constants A, B is given as

$$f(x, c) = Ae^{c_1 x} + Be^{c_2 x}, \quad A, B = \text{const.} > 0, \tag{3}$$

and our goal is to fit this function using Gauss-Newton method. In other words, determine parameters $c = (c_1, c_2)$ which are positive constants by knowing x and output data $f(x, c)$.

Hints:

1. Study Example 9.6 of [2].

2. You can choose exact values of positive constants A, B and parameters c_1^*, c_2^* as well as choose interval for x as you want. For example, take $A = 5, B = 2$ and interval for x as $x \in [-3, 3]$, take parameters to be determined $c_1^* = 2, c_2^* = 3$ and discretize $x = [-3, 3]$ by N discretization points as $x_i = -3 + i * (6/N), i = 1, ..., N - 1$.

3. For iterative update of $c = (c_1, c_2)$ try to use Gauss-Newton iterative formula
$$c^{k+1} = c^k - [J^T(c^k)J(c^k)]^{-1} J^T(c^k) r(c^k), \tag{4}$$
where $k = 0, ...M$ is the number of iteration and $J(c^k)$ is the Jacobian matrix of the residual $r(c^k)$. Here, $c^0 = (c_1^0, c_2^0)$ is a good initial guess for $c = (c_1, c_2)$.

4. Try also to use Levenberg-Marquardt method. This method is based on the finding of minimum of the regularized function
$$F(c) = \frac{1}{2} r(c)^T r(c) + \frac{1}{2}\gamma(c - c_0)^T(c - c_0) = \frac{1}{2}\|r(c)\|_2^2 + \frac{1}{2}\gamma\|c - c_0\|_2^2, \tag{5}$$
where c_0 is a good initial guess for c and $\gamma \in (0, 1]$ is a small regularization parameter. In the Levenberg-Marquardt method the linear system which should be solved at every iteration k is
$$(J^T(c^k)J(c^k) + \gamma^k I)(c^{k+1} - c^k) = -J^T(c^k) r(c^k). \tag{6}$$
You can choose γ^k as a constant at every iteration, or iteratively as a sequence of regularization parameters γ_k which slowly converges to zero as
$$\gamma^k = \frac{\gamma_0}{(k + 1)^p}, \quad \gamma_0, p \in (0, 1].$$

5. Define the residual function $r(c)$ in (4) and in (6) at every iteration k as
$$r(c) = y - f(x, c), \tag{7}$$
where $y = y_i, i = 1, ..., m$ are known data points. Use information in the previous comp.ex. to generate data $y = y_i, i = 1, ..., m$.

   More precisely, compute discrete values of $y_i, i = 1, ..., m$ for the known values of parameters c_1, c_2 from (3) for discrete values of $x_i, i = 1, ..., m$. Then forget about these parameters (we will call them exact parameters c_1^*, c_2^*) and apply Gauss-Newton or Levenberg-Marquardt iterative formula to recover these exact parameters.

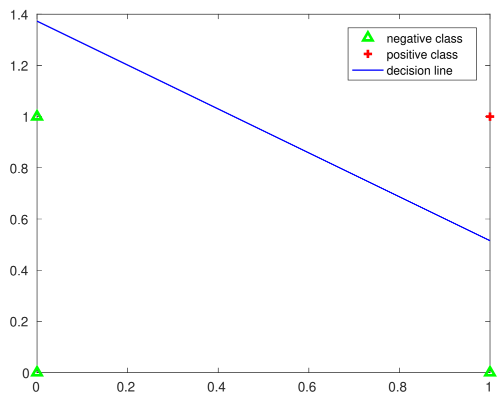
6. Add random noise δ to data $y = y_i, i = 1, ..., m$ using the formula $y_\sigma(x, c) = y(x, c)(1 + \delta\alpha)$, where $\alpha \in (-1, 1)$ is randomly distributed number and δ is the noise level (if noise in data is 5%, then $\delta = 0.05$).

7. Test the Gauss-Newton and the Levenberg-Marquardt methods for different initial guesses $c^0 = (c_1^0, c_2^0)$ for the exact parameters $c^* = (c_1^*, c_2^*)$ and different noise level δ in data $y = y_i, i = 1, ..., m$.

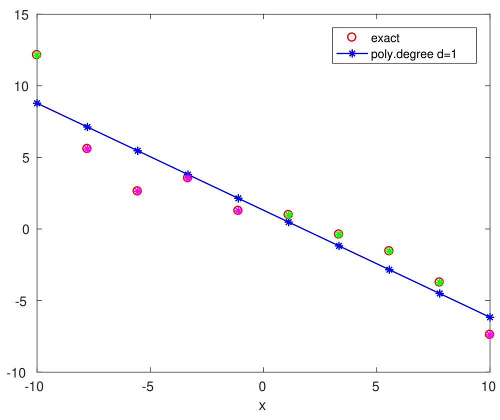
8. Analyze convergence of iterative formulas (4) and (6) by computing the relative errors e_1, e_2 for computed parameters c_1, c_2 obtained at the final iteration M in both methods as:
$$e_i = \frac{|c_i - c_i^*|}{|c_i^*|}, \quad i = 1, 2. \tag{8}$$
Here, (c_1^*, c_2^*) are exact values and $c = (c_1, c_2)$ are computed values at the final iteration M in methods (4) and (6), correspondingly.

9. Present how relative errors (8) depend on the relative noise $\delta \in [0, 1]$ in graphical form and in the corresponding table.

10. Choose different number of discretization points N of the input interval for x and present results of computations in graphical form and in the corresponding table. More precisely, present how relative errors (8) depend on the number of discretization points N of the input interval for x.



a)



b)

5 Computer exercise 4 (1 b.p.)

Least squares can be used for classification problems appearing in machine learning algorithms. In this exercise we will try to construct linear classifiers for different training sets using least squares approach and perceptron learning algorithm.

Implement in Matlab two different classifiers- least squares classifier and perceptron learning algorithm - for following training sets:

- I) for data given in the following table:

x	y	Class
1	1	positive
1	0	negative
0	1	negative
0	0	negative

  Note that boolean attributes (x, y) in the table above can present Cartesian coordinates of 4 points, see Figure a). Using Perceptron learning algorithm you should get decision line similar to the blue line of this Figure. The desision line should separate red point belonging to the positive class from the green points belonging to the negative class. However, you will get different decision line using least squares approach. Present decision lines for both approaches and compare obtained results.

- II) 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. Using least squares approach you can get similar results as presented in Figure b).

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 iterative process is too slow?

Analyze which one method (least squares or perceptron learning algorithm) performs best and why? Try to explain why least squares approach can fail if 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

$$\omega_0 + \omega_1 x + \omega_2 y = 0, \tag{9}$$

where x, y are Cartesian coordinates. Note that the equation (9) can be rewritten for the case of a linear least squares problem as

$$\omega_0 + \omega_1 x = -\omega_2 y \tag{10}$$

or as

$$-\frac{\omega_0}{\omega_2} - \frac{\omega_1}{\omega_2} x = y. \tag{11}$$

3. Suppose that we have measurements $y_i, i = 1, ..., m$ in (11) and our goal is to determine coefficients in (11) from these measurements. We can determine coefficients $\omega_0, \omega_1, \omega_2$ by solving the following least squares problem:

$$\min_{\omega} \|A\omega - y\|_2^2 \tag{12}$$

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 the following:

   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.

5. The decision line can be presented in Matlab for already computed weights by Perceptron learning algorithm using the formula (11).

6 Computer exercise 5 (2 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 course homepage for description of this project.

Solve the Helmholtz equation

$$\begin{aligned} \Delta u(x,\omega) + \omega^2 \varepsilon'(x) u(x,\omega) &= i\omega J, \\ \lim_{|x|\to\infty} u(x,\omega) &= 0. \end{aligned} \tag{13}$$

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

$$\varepsilon'(x) = \varepsilon_r(x)\frac{1}{c^2} - i\mu_0 \frac{\sigma(x)}{\omega}, \tag{14}$$

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 $\omega, \varepsilon_r, \sigma, J$.

Hints:

1. Study Example 12.5 of [2] where is presented solution of the Dirichlet problem for the Poisson's equation using PETSc. PETSc programs for solution of this problem are available download from the course homepage: go to the link of the book [2] and click to "GitHub Page with MATLAB Source Codes" on the bottom of this page.

2. Modify PETSc code of the Example 12.5 of [2] such that the equation (13) can be solved. Note that solution of the equation (13) 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 [2] 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
```

Further information about PETSc can be found on the link:

```
https://www.mcs.anl.gov/petsc/
```

4. Choose the two-dimensional convex computational domain Ω as you prefer. For example, $\Omega = [0,1] \times [0,1]$. 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. Values of c, μ_0, ε_0 in (14) 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] J. Demmel, *Applied Numerical Linear Algebra*, SIAM, 1997.

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