Norwegian University of Science and Technology
Department of Mathematical Sciences

MA2501: Numerical Methods
Course Lecturer: Asif Mushtaq

Semester Project

- This project counts for 30% of the final grade.
- You have to work in groups.
- You have to produce a report written on a computer with your solutions (preferably in LaTeX, but there is nothing wrong with using something else).

  You do not need to include code in your report. It is enough to refer to the code files produced. It is fine to just go through the tasks point by point and answer them. Just try to keep the document organized and readable.
- Try to write somewhat efficient, organized, and well documented MATLAB code.
- The report, and the code produced, should be sent electronically to asifm@math.ntnu.no. The deadline for everything is **17 April, 2016 at 24:00**. Mark all the material with your candidate number(s), not your name(s).

MATLAB advice

All MATLAB code created in this project should satisfy the following criteria:

- Function files should contain a help text. A user should be able to use the routine from the information given by the command:
  `help [function name]`.
- The code should be self-documented, with a reasonable amount of comments in the code.
- For the iterative methods, a warning message should be printed if the iterations do not converge.

In addition try to make use of MATLAB's proficiency at working with vectors and matrices, i.e. avoid unnecessary for-loops. Also try to avoid stuff like repeating the same computation several times, rather than computing the result once and storing it. Feel free to make use of built-in functions like `norm`, `max`, `sort`, etc.

PART I

Note: For problem 1 you should hand in a written answer to all the questions. You can use LaTeX template to write the project report. It is sufficient to just answer the questions.

(1) Given the function

$$f(x) = x^3 + x^2 - 3x - 3, \qquad x \in [1, 2].$$

(a) Prove that f has at least one root in the interval $[1, 2]$.

(b) Plot $f(x)$ by using MATLAB, and locate the root(s). Are there more than one in the given interval?

(c) Show that the equation $f(x) = 0$ can be reformulated in either of the forms

$$x = \sqrt[3]{-x^2 + 3x + 3} = g_1(x), \tag{1}$$

$$x = \frac{-x^2 + 3x + 3}{x^2} = g_2(x), \tag{2}$$

$$x = \sqrt{\frac{-x^2 + 3x + 3}{x}} = g_3(x), \tag{3}$$

Convince yourself that $\sqrt{3}$ is the root of f, as well as the fixed point of each of the right hand side functions above.

(d) For each $g_i(x)$ above, make the iteration scheme $x_{k+1} = g_i(x_k)$. Start with $x_0 = \sqrt{3}$ and do 50 iterations with each of these schemes. Describe what you observe, and explain it.

(e) Consider again the potentially useful iteration schemes. Using $x_0 = 1$, try to find an upper bound on the maximum number of iterations k required to ensure that

$$|x_k - \xi| < 10^{-10},$$

assuming you do not know the fixed point ξ.

Verify your result with a numerical experiment.

Implementation

Some hints:

- Before starting, make sure that you master MATLAB on an elementary level. See e.g. the slides at the homepage. You can also consult the *Getting started page* at the MATLAB documentation, there are plenty of tips there.

  You should know:

  - functions in MATLAB, see: `function, nargin, nargout.` Remember that functions in MATLAB can use function handles, vectors, matrices, etc. as arguments.
  - control sentences: `if-else, for, while, break.`
  - Vector and matrix computations. For example: the linear system $\mathbf{Ax} = \mathbf{b}$ is solved by `x=A\b.` Do not mix row- and column vectors.
  - Other useful commands: `norm, sprintf, fprintf, disp.`

- Start as simple as possible. Do one thing at the time, and check that it is correct before proceeding. Compare with hand calculations on simple problems, if you find this easier.

- Make sure you understand how Newtons method for systems of equations works.

Note: For problem 2 we only want you to hand in the two MATLAB functions `testProblem` and `newton`, written according to the given specifications.

(2) You are supposed to write a library routine for solving a system of nonlinear equations

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \qquad \mathbf{f} : \mathbb{R}^m \to \mathbb{R}^m,$$

using Newtons method.

Specifications:

- *Arguments in:*
  - A handle to a function with a vector $\mathbf{x}$ as input parameter, returning $\mathbf{f}(\mathbf{x})$ and $J(\mathbf{x})$.
  - A vector with starting values $\mathbf{x}^{(0)}$.
  - An error tolerance ϵ.
  - The maximum allowed number of iterations `nMax`
- *Arguments out:*
  - The solution $\mathbf{x}$.

  - The number of iterations `nIt`.
  - A flag, `iFlag` telling whether the iterations were successful or not.
- *And:*
  - Make a help text (a user should be able to use the routine from the information given by `help`).
  - Stop the iterations when $\max_i |f_i(x)| \leq \epsilon$ (success) or `nIt` $\geq$ `nMax` (fiasco).
  - Write a warning message if the iterations do not converge.
  - The code should be self-documented, with a reasonable amount of comments in the code.

As a testproblem, use the equation

$$\begin{aligned} f_1(x_1, x_2) &= x_1^2 - 2x_1 - x_2 + 0.5 = 0, \\ f_2(x_1, x_2) &= x_1^2 + 4x_2^2 - 4 = 0. \end{aligned} \tag{4}$$

(a) (Preparation)
Find the Jacobian $J_f(\mathbf{x})$ of $\mathbf{f}(\mathbf{x})$ given in (4).
Do one Newton iteration by hand, starting with $\mathbf{x}^{(0)} = (1, 1)^T$.

(b) Write a MATLAB-function, `testProblem`, taking a vector $\mathbf{x}$ as an input argument, and returning the vector $\mathbf{f}(\mathbf{x})$ and the Jacobian $J_f(\mathbf{x})$ for the problem (4). The first line of this function will typically be

```
function [f,J] = testProblem(x)
```

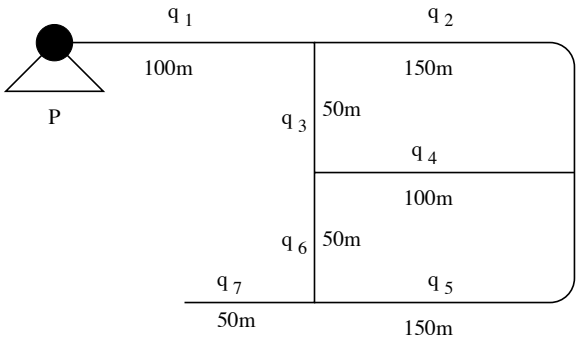
(c) Write the library function by the specifications given above. The first line will typically be

```
[x,nIt,iFlag] = newton(f,x0,tol,nMax)
```

(d) Use the function to solve the problem (4).

Application (Water flow network)

Water flows through a pipe network as shown in the picture below.



We would like to know the distribution of the water flow, that is the flow rate q_i, $i = 1, 2 \ldots, 7$ through each pipe in the net. The pump produce an outlet gauge pressure of $4.1 \cdot 10^5$ Pa.

The following controls the flow:

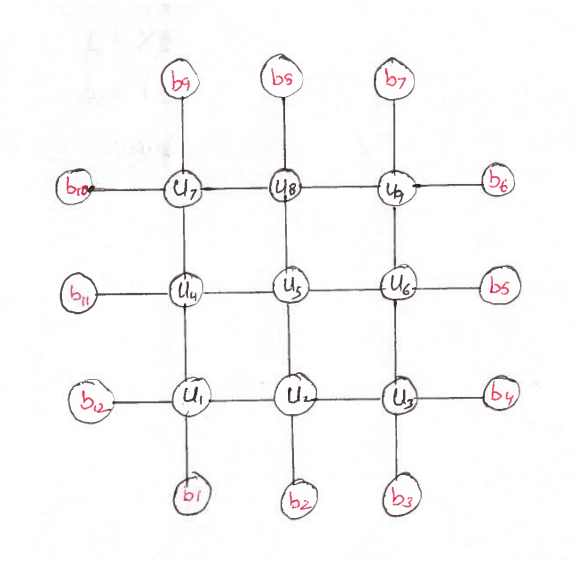
- In each junction the rate of water that enters the junction is equal to the rate of water that leaves. For the junction up in the middle, that gives
$$q_1 - q_2 - q_3 = 0.$$
Similar equations can be obtained for the remaining junctions. .
- In each of the pipes, the pressure is reduced due to friction. The pressure drop is given by
$$\Delta P = \frac{8 f \rho L}{\pi^2 d^5} q^2$$
where f is a friction factor, ρ is the density of water, L is the length of the pipe, q is the flow rate and d is the inside diameter of the flow. For the current network:
$$f = 0.00225, \quad \rho = 998\text{kg/m}^3, \quad d = 0.15\text{m},$$
- The sum of the pressure drops around a loop in the network is equal to 0.

Note: For problem 3 we want you to hand in the MATLAB-function described below and the script in which the function is called and the output presented. Also include a brief derivation of the system of equations in your written answer.

(3) Consider the pipe network problem detailed above. Find seven equations that determine the flow rates q_i, $i = 1, 2 \ldots, 7$ (in m^3/s) in the pipes. These can be written as a system of nonlinear equations $\mathbf{f}(\mathbf{q}) = \mathbf{0}$ with $\mathbf{q} = (q_1, q_2, \ldots, q_7)^T$. Create a MATLAB-function similar to `testProblem`, taking a vector $\mathbf{q}$ as an input argument, and returning the vector $\mathbf{f}(\mathbf{q})$ and the Jacobian $J_f(\mathbf{q})$, for these flow network equations. Solve the equations using your function `newton`.

PART II

The numerical solution of PDEs may require solving a linear system that has large number of zeros and a banded structure of the non-zero elements. Such system can be illustrated in following figure:



The two-dimensional potential equation can be solved numerically by defining a mesh of points in the region of the interest and approximating the derivatives by finite differences. For example, to solve the potential equation

$$u_{xx} + u_{yy} = 0, \quad 0 \leq x \leq 1, \quad 0 \leq y \leq 1$$

with boundary conditions

$$u(0,y) = y_2, \quad u(1,y) = 1, \quad 0 \le y \le 1$$
$$u(x,0) = x_2, \quad u(x,1) = 1, \quad 0 \le x \le 1$$

and a mesh size $\Delta x = \Delta y = 0.2$, we obtain the following linear system:

$$\begin{aligned}
4u_1 - u_2 - u_4 &= b_1 + b_{12} \\
-u_1 + 4u_2 - u_3 - u_5 &= b_2 \\
-u_2 + 4u_3 - u_6 &= b_3 + b_4 \\
-u_1 + 4u_4 - u_5 - u_7 &= b_{11} \\
-u_2 - u_4 + 4u_5 - u_6 - u_8 &= 0 \\
-u_3 - u_5 + 4u_6 - u_9 &= b_5 \\
-u_4 + 4u_7 - u_8 &= b_9 + b_{10} \\
-u_5 - u_7 + 4u_8 - u_9 &= b_8 \\
-u_6 - u_8 + 4u_9 &= b_6 + b_7
\end{aligned}$$

This system is well suited for iterative solution scheme like Jacobi, Gauss-Seidal and SOR methods. For $\mathbf{b} = [0.1, 0.25, 0.9, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.8, 0.5, 0.2]^T$.

(1) The Jacobi method is based on the transformation of the linear system $\mathbf{Au} = \mathbf{b}$ into the system $\mathbf{u} = \mathbf{Cu} + \mathbf{d}$ and generating a sequence of approximations $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \ldots$, where

$$\mathbf{u}^{(k)} = \mathbf{C}\mathbf{u}^{(k-1)} + \mathbf{d}$$

This methodology is similar to the fixed-point iteration method for non-linear functions. Write a MATLAB-function whose first typical line will be

```
function x = Jacobi(A, b, initial-condition, tol, nMax)
```

Output should be:

(a) Final approximate solution(starting with an initial solution vector that is identically=0).

(b) Messages which state Jacobi is converges or not.

(c) Report solution output after 10 and 40 iterations up-to 4 decimal places. Write what you observe when you increase the number of iterations.

(2) In Gauss-Siedal iterative method, if we introduce additional parameter ω then the iterative process can be accelerated to the convergence. The parameter ω controls the proportion of the update that comes from the previous solution and proportion that comes from the current calculation. For $0 < \omega < 1$, the method is know as *successive-under-relation(SUR)*, for $1 < \omega < 2$, the method is know as *successive-over-relation(SOR)* and for $\omega = 1$ this becomes Gauss-Seidal method.

Write a MATLAB-function whose first typical line should be

```
function x = SOR(A, b, initial-condition, w, tol, nMax)
```

(a) Document solution output with number of iteration. MATLAB-Programme should also produce the convergence or divergence messages of the process.

(b) What will be the optimal value of ω, if all initial values are set equal to zero with tolerance 10^{-k}, where $k = 4, 6$

(c) Write your observation about these iterative methods and which method you found better to solve the given system and why?