MATH402 Homework II **Due: 23:59 December 26**

The instructor is aware of the possibilities that exist for obtaining homework solutions outside of one's own abilities. Therefore, read the following rules carefully!

- You can collaborate or work with anyone. However, the work submitted must be your understanding. If your works contain seems quite similarity, you can **share the total points**.
- If your work involves significant help from reference books or papers, add a note in your paper to indicate this. The murkiness about "using online tutorials" is resolved below:
  - Do not submit your homework problems to someone or some agency who will do the exam for you. These people do everyone except themselves a disservice. They harm you because you won't learn and therefore won't earn (neither money nor the joy of knowing something). They mock the education process. They benefit by collecting cash because you may not know something. Don't let this be a pattern for your life. **A grade is a grade and over time will mean less and less, but knowledge acquired pays dividends for as long as you retain that knowledge.**
- **The instructor reserves the right to ask the student to explain the answers for any or all problems on the homework.** If the student is unable to provide a satisfactory answer, then it will be assumed that the work was not done in an earnest manner and as such the problem in question will receive no credit.
- Show all your work. Correct answers without sufficient explanation might **not** get full credit.
- The assignment consists of 6 questions for a total of 100 marks.
- Submit your assignment **electronically** via https://odtuclass.metu.edu.tr/.
- When you scan your hand-writing part, please make sure that one can easily read the homework when printed. Do not submit photos of handwriting homework.
- The report of MATLAB part should be written via **LaTeX**.
- Please, do not forget send **.m files**, **.tex file** of MATLAB part.
- Combine the hand-writing part and the report of MATLAB part in **a single pdf**.
- Please, do not forget write your name and surname to your documents.
- Together with your MATLAB **.m files**, make a compressed file such as ZIP (never use RAR!), of all your work (including your PDF report (.pdf and .tex files)). Upload only the compressed file having a name as **your name, surname, and homework number**, for instance, **mkutuk_hmw2**.
- The report for a programming part should consist of
  - Brief description of the problem;

  - Results such as data, graphs etc. Try to avoid attaching large set of data unless it is really necessary;
  - Discussion, comments, explanation and conclusion on your numerical observations. This part is equally important as your computer codes and the data you collect, and it helps with understanding concepts, algorithms, or other relevant issues not discussed during lectures.
  - Please, do not insert your MATLAB **.m files** into the report if not necessary.

- Every .m file should be documented, such as:

```
1
2 % Author: (Your Name)
3 %
4 % Description:
5 % (Give a brief description of what your program does.)
6 %
7 % Input:
8 % (State what the program inputs are.)
9 %
10 % Output:
11 % (State what the program outputs are.)
12 %
13 % Usage:
14 % (Give an example of how to use your program.)
```

- **Late submission is not allowed!**

- **Please write the exact wording of the Pledge, following by your signature and date, in the beginning of your paper. Otherwise, your homework will not be evaluated.**

  "*I confirm that I have read the instructions carefully and fully understood my responsibilities. I hereby declare that I have neither given nor received aid in this homework nor I have concealed any violation of the University Honor code.*"

1. (10 *pts.*) Symmetric rank-one (SR-1) quasi-Newton Method is given by

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}.$$

Using the Sherman-Morrison-Woodburry formula

$$\widetilde{A}^{-1} = A^{-1} - \frac{A^{-1} a b^T A^{-1}}{1 + b^T A^{-1} a},$$

for $\widetilde{A} = A + ab^T$ (the rank-one update of A), show that the updating method for H_k, the inverse of B_k, is

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k},$$

where

$$s_k = x_{k+1} - x_k, \quad y_k = \nabla f_{k+1} - \nabla f_k.$$

2. (25 *pts.*) Assume $f \in \mathcal{C}^1(\mathbb{R}^n)$ has Lipschitz continuous gradient with Lipschitz constant L.

   a) (8 *pts.*) Show that

$$f(x+p) \le f(x) + \nabla f(x)^T p + \frac{L}{2}\|p\|^2, \qquad \forall x, p \in \mathbb{R}^n.$$

   b) (4 *pts.*) Show that

$$f(x) - f(x - \alpha \nabla f(x)) \ge \alpha \left(1 - \frac{L\alpha}{2}\right) \|\nabla f(x)\|^2, \quad \text{for any } x \in \mathbb{R}^n, \text{ and } t > 0.$$

   c) (5 *pts.*) Let $\{x_k\}$ be the sequence generated by the steepest descent method with the exact line search to minimize $f(x)$ over $\mathbb{R}^n$. Show that

$$f(x_k) - f(x_{k+1}) \ge \frac{1}{2L}\|\nabla f(x_k)\|^2.$$

   d) (8 *pts.*) Assume that f is bounded below by $\underline{f}$ over $\mathbb{R}^n$ and let $\{x_k\}$ be the sequence generated by the steepest descent method with the exact line search to minimize $f(x)$ over $\mathbb{R}^n$.

      i) (4 *pts.*) Show that $\{f(x_k)\}$ is nonincreasing and, in addition,

$$f(x_{k+1}) < f(x_k) \qquad \text{for any } \ k \ge 0$$

      unless $\nabla f(x_k) = 0$.

      ii) (4 *pts.*) Show that $\nabla f(x_k) \to 0$ as $k \to \infty$.

3. (15 *pts.*) By using the conjugate gradient method, we want to solve the following optimization problem

$$\min f(x) = \frac{1}{2}x^T Ax - b^T x$$

or equivalently

$$Ax = b,$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric and positive definite matrix.

(a) (8 *pts.*) Prove that the conjugate directions satisfy

$$p_i^T p_j \geq 0 \quad \forall i, j.$$

(b) (7 *pts.*) Show that the following formula for α_i obtained in the class

$$\alpha_i = \frac{b^T p_i}{p_i^T A p_i}$$

is equivalent to

$$\alpha_i = \frac{r_i^T r_i}{p_i^T A p_i}.$$

4. (20 *pts.*) Consider the following constrained optimization problem

$$\begin{aligned} &\text{minimize} \quad x_1 x_2 \\ &\text{subject to} \quad 4x_1^2 + x_2^2 = 4. \end{aligned}$$

a) (4 *pts.*) Write the corresponding Lagrangian function and compute its gradient.

b) (5 *pts.*) Find all stationary points of the Lagrangian function.

c) (5 *pts.*) Apply one iteration of the **method of multipliers** to above optimization problem starting with the initial guesses $x_0 = (0, -1)^T$ and $\lambda = 1$ and choosing the step–length $\alpha_0 = 1$.

d) (6 *pts.*) Compute the tangent cone and the null space of the constraint Jacobian at the stationary points.

5. (15 *pts.*) In this exercise, you will write a Matlab routine to implement **Symmetric Rank–1 Inverse Update with Armijo Condition** in the form of,

```
[X,Grad,ite] = SR1_inverse(fhandle,x0,tol,H0,maxit,alpha0,c,mu,amax)
```

taking the following input arguments:

- objective function `fhandle`, initial guess `x0`,
- tolerance value for the termination condition, i.e., $\|\nabla f(x_k)\| <$ `tol`,
- initial inverse matrix `H0`, maximum number of iteration `maxit`,
- initial step–length `alpha0`, Armijo constant `c`,
- backtracking parameter `mu`, maximum number of iterations `amax`

and return

- a matrix `X = [x0; x1; x2; ...]` containing the whole iterations, a vector `Grad` containing $\|\nabla f(x_k)\|$, and the number of iterations `ite`.

Symmetric Rank–1 Inverse Update with Armijo Condition

Input: Given the objective function $f(x)$, initial guess x_0, tolerance number `tol`, initial inverse matrix `H0`, maximum number of iteration `maxit`, Armijo constant c, backtracking constant μ, maximum number of iteration for Armijo `amax`, and initial guess for step–length α_0.
Set $k := 0$.
while $\|\nabla f(x_k)\| \geq$ `tol` and $k \leq$ `maxit` **do**
  Compute the search direction $p_k = -H_k \nabla f(x_k)$.
  Compute α_k by using backtracking approach with Armijo condition:
  Set $j := 0$.
  while $f(x_k + \alpha_j p_k) \geq c\alpha_j \nabla f(x_k)^T p_k$ and $j \leq$ `amax` **do**
    $\alpha := \alpha * \mu$.
    Set $j := j + 1$.
  end while
  Update the solution $x_{k+1} = x_k + \alpha_k p_k$.
  Set $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.
  Update inverse matrix

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k}$$

  Set $k := k + 1$.
end while

- Take the Rosenbrock banana function

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

  as a benchmark function. Write a file `Rosenbrock.m` for the Rosenbrock banana function `[f,df,Hess] = f(x)`, which returns the values of the function f, gradient $\nabla f(x)$, and Hessian $\nabla^2 f(x)$, respectively.
- Run your code for different values of `tol` $= 10^{-3}, 10^{-6}, 10^{-9}$, initial guesses $x_0 = (-0.5, 1)^T$ and $x_0 = (1.1, 1.1)^T$, $H_0 = I$, `maxit` $= 10000$, `alpha0` $= 1$,`c` $= 1e-4$, `mu` $= 0.5$, and `amax` $= 100$.
- Display your results for each initial guesses as the following:

```
Tol   | iteration    |    x value    | Norm_Gradient
----- |--------- ----|-----------------------
1e-3  |   iters_1    |     x_1       |  grad_1
1e-6  |   iters_2    |     x_2       |  grad_2
1e-9  |   iters_3    |     x_3       |  grad_3
```

We note that the norm of gradient is given for the last step.

- Plot the norm of the gradient for each tolerance in the same figure.
- In this exercise, you need to write also a function

  `function [alpha] = armijo(fhandle, x, p, alpha0, c, mu, amax)`

  that returns the step–length that satisfies Armijo condition. As input arguments, the function accepts
  - a function handle `fhandle`, current iterate `x`, descent direction `p`, initial step–length `alpha0`, Armijo constant `c` backtracking parameter `mu`, and maximum number of iterations `amax`.

6. (15 *pts.*) In this exercise, you will write a Matlab routine to implement **Conjugate Gradient** in the form of,

   `[X,res,ite] = conj_grad(A,b,x0,tol,maxit)`

   taking the following input arguments:

   - matrix $A \in \mathbb{R}^{n\times n}$ and vector $b \in \mathbb{R}^n$
   - initial guess `x0`,
   - tolerance value for the termination condition, i.e., $\|r_k\| <$ `tol`,
   - maximum number of iteration `maxit`,

   and return

   - a matrix `X = [x0; x1; x2; ...]` containing the whole iterations, a vector `res` containing $\|r_k\|$, and the number of iterations `ite`.

Conjugate Gradient Method

Input: Given a matrix $A \in \mathbb{R}^{n\times n}$, a vector $b \in \mathbb{R}^n$, an initial guess x_0, tolerance number `tol`, and maximum number of iteration `maxit`.
Set $k := 0$.
Compute $r_0 = Ax_0 - b$ and $p_0 = -r_0$.
while $\|r_k\| \geq$ `tol` and $k \leq$ `maxit` **do**
$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}$
$x_{k+1} = x_k + \alpha_k p_k$
$r_{k+1} = r_k + \alpha_k A p_k$
$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$
$p_{k+1} = -r_{k+1} + \beta_k p_k$
$k := k + 1$
end while

- Consider the following input data set:

  - A is the Hilbert matrix, whose elements are

$$A_{\mathrm{i,j}} = 1/(i + j - 1).$$

  - vector $b = (1, 1, ..., 1)^T$
  - initial guess `x0` $= 0$
  - the number of iterations required to reduce the residual below 10^{-6} with `maxit=1000`.

- Run your code for different dimension values of $n = 5, 8, 12, 20$.
- Display your results as the following:

```
n   | iteration    |    x value   | Norm of residual
--- |--------- ----|-----------------------
5   |   iters_1    |     x_1      |  res_1
8   |   iters_2    |     x_2      |  res_2
12  |   iters_3    |     x_3      |  res_3
20  |   iters_4    |     x_4      |  res_4
```