Homework # 4

Due: 03/31/2017 12:00 noon
Please read the following instructions carefully

- you should submit a **single pdf file** which contains the following things
  - all the codes (copy pasted or published from MATLAB)
  - hand written things or typed material without using MATLAB programming (indicated by (paper and pencil) in the questions)
  - results of each problem just below your code for the corresponding problem

  If there is no pdf file without these things your homework will not be graded.

- You should put all your .m files in a directory named `HW4_<your EID>`, zip the directory and submit it. The .m files should be named as `HW4_1.m`,`HW4_2.m`.... if there are subdivisions you can name it as `HW4_1a.m` etc. It's **very important your zip file should open to a directory which contains all your .m files**, if it just spits out individual files it will clutter our directory and we won't grade at all.

- Finally **most important of all things your pdf file should not be included in the zip file and it should be submitted separately**. There is an interface in canvas which we can use to cycle through all of your PDF files when grading but it works only if the PDF file is submitted directly. Name your pdf file as `HW4_<your EID>.pdf`

- The questions which are *italicized* are either important or tricky or both.

1. Consider the following eigenvalue problem

$$Ax = \lambda x,$$

where

$$A = \begin{bmatrix} 2 & 8 & 10 \\ 8 & 4 & 5 \\ 10 & 5 & 7 \end{bmatrix}.$$

   (a) Write a MATLAB program **Powereig** which uses the power method to determine the maximum eigenvalue of the system. Take the initial guess as $x_0 = (1, 1, 1)^T$ and iterate until the relative error

$$err_{rel} = \frac{\left\|\lambda^{(k+1)} - \lambda^{(k)}\right\|_2}{\left\|\lambda^{(k)}\right\|_2}$$

   is less than 0.001. Report the number of iterations. (10 points)

   (b) Find the inverse of A using `inv()` (built-in) function in MATLAB and use it to compute the minimum eigenvalue of A using the power method. Report the number of iterations. (10 points)

   (c) Check your solutions from (a) and (b) with the maximum and minimum eigenvalues of A obtained from `eig()` (built-in) function in MATLAB. (5 points)

2. You are designing a spherical tank to hold water for a small village. The volume of liquid it can hold can be computed as

$$V = \pi h^2 \frac{[3R - h]}{3}.$$

where V = volume $[m^3]$, h = depth of water in tank $[m]$, and R = the tank radius $[m]$. If $R = 3m$, what depth must the tank be filled to so that it holds $30m^3$.

$$err_{rel} = \frac{\left\| h^{(k+1)} - h^{(k)} \right\|_2}{\left\| h^{(k)} \right\|_2}.$$

(a) Plot the volume $V = f(h)$ over the interval $h \in [0, R]$. (5 points)

(b) Write a MATLAB program called **Bisect** that implements the Bisection method. Using the **Bisect** program, find the depth (h) by starting with a bracketing region of $[0, R]$ and iterating until the relative error err_{rel} is less than 10^{-4}. (10 points)

(c) Write a MATLAB program called **NewtRaph** that implements the Newton method. Using **NewtRaph** program, find the depth (h) by taking the initial guess as $h^{(0)} = R$ and iterating until the relative error err_{rel} is less than 10^{-4}. (10 points)

(d) *Write a MATLAB program called* **FixedPoint** *that implements the Fixed-Point method. Using* **FixedPoint** *program, find the depth (h) by taking the initial guess as $h^{(0)} = R$ and iterating until the relative error err_{rel} is less than 10^{-4}. (This is slightly tricky you have to come up with a right $g(h)$ to get the converged correct solution) (10 points)*

(e) Plot the relative error err_{rel} vs. the number of iterations for the three methods (on the same plot). Add proper x, y labels and a legend for all the curves in the plot. Discuss the performance of each of the methods. (in terms of computation time and iterations) (10 points)

3. Consider the nonlinear system of equations $\mathbf{f}(x, y) = [f_1(x, y), f_2(x, y)]$,

$$\begin{aligned} f_1(x, y) &= 1 - 2y^3 + 2x^2 - 4x \\ f_2(x, y) &= x^4 + 4y^4 + 4y - 4. \end{aligned}$$

We look for the root of $\mathbf{f}(x, y) = \mathbf{0}$. Note that we use **boldface** letters for vectors.

(a) Write a MATLAB program called **NewtRaphm** that finds the root of multidimensional root finding problem using the Newton method taught in class. Using your **NewtRaphm** program, find the root of $\mathbf{f}(x, y) = \mathbf{0}$. In particular, take the initial guess as $(x^{(0)}, y^{(0)}) = (0.7, 0.7)$ and iterate until the relative error err_{rel} (defined in problem 2, here $x^{(k)}$ and $y^{(k)}$ are the 2 components of $\mathbf{x}^{(k)}$) is less than 10^{-4}. Save the roots $(x^{(k)}, y^{(k)})$ at each iteration, then plot the trajectory of the roots over the contour plots of $f_1(x, y)$ and $f_2(x, y)$ (Figure 1). To help you with this process, we have provided a MATLAB function **PlotTrajectory** which plots this figure. The template for **PlotTrajectory** is `PlotTrajectory(no,xk,yk)`, where `no` stands for problem number which in this case is 3, `xk` stands for the vector $x^{(k)}$ and `yk` for the vector $y^{(k)}$. Although it should work like a "black box" we encourage you to look inside the function to improve your plotting skills in MATLAB. (10 points)

(b) Write a MATLAB program called **FixedPointm** that implements a *multi-dimensional* Fixed-Point method. In particular, similar to one-dimensional fixed point method taught in class, we can rewrite the above root finding problem for $\mathbf{f}(x, y) = \mathbf{0}$ as

$$[x, y] = \mathbf{g}(x, y).$$

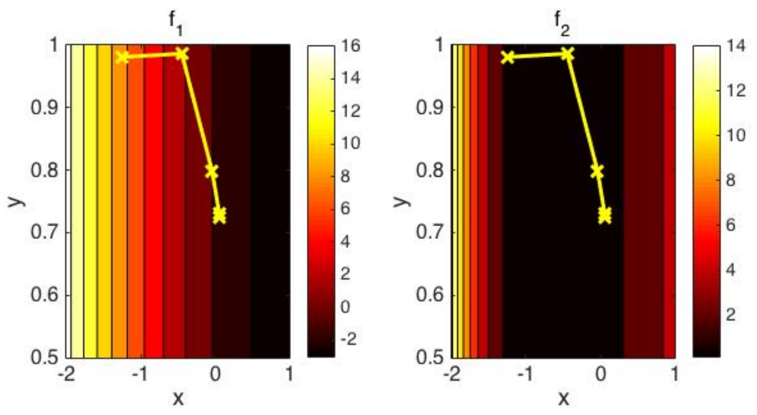

Figure 1: Trajectory of $(x^{(k)}, y^{(k)})$ for Newton method.

where $\mathbf{g}(x, y)$ in this case is given by

$$g_1(x, y) = \frac{1}{4}(2x^2 - 2y^3 + 1),$$

$$g_2(x, y) = \frac{1}{12}(-x^4 - 4y^4 + 8y + 4),$$

and $g_1(x, y)$ and $g_2(x, y)$ are the two components of $\mathbf{g}(x, y)$. Then a multidimensional fixed point method can be stated as

$$[x^{k+1}, y^{k+1}] = \mathbf{g}(x^k, y^k).$$

Using your **FixedPointm** program, find the root of $\mathbf{f}(x, y) = \mathbf{0}$ by taking the initial guess as $(x^{(0)}, y^{(0)}) = (0.7, 0.7)$ and iterating until the relative error err_{rel} is less than 10^{-4}. Save the roots $(x^{(k)}, y^{(k)})$ at each iteration, then plot the trajectory of the roots over the contour plots of $f_1(x, y)$ and $f_2(x, y)$ (Figure 2). Here also use the function `PlotTrajectory.m` with problem number as 3. (10 points)

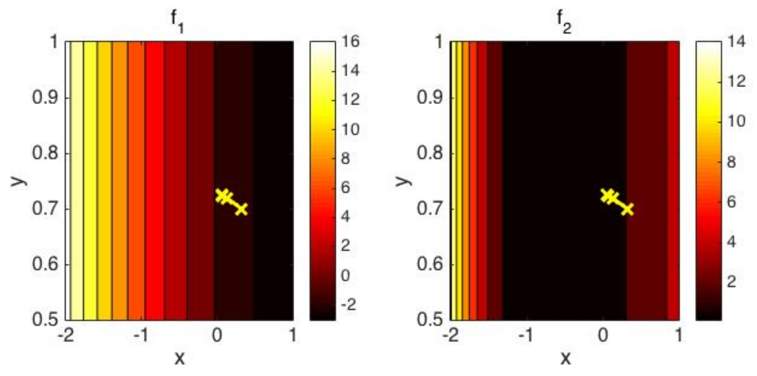

Figure 2: Trajectory of $(x^{(k)}, y^{(k)})$ for Fixed-Point method.

(c) Plot the relative error err_{rel} vs. the number of iterations for the two methods (on the same plot). Add proper labels and a legend. Discuss the performance (computation time and the number of iterations) of each method. (10 points)