ME 2016 - Computing Techniques
Fall 2018 – Section A

Homework 4
Due Thursday, October 11
Electronic submission part due at 2:55 PM (5 minutes before class time)
Hardcopy part: end of class

Root Finding: Newton-Raphson method
Box your final results

Problem 1: Textbook problem 6.13 **(10 points)**
Ignore the techniques that were not studied in class and solve this problem *by hand*.

6.13 You must determine the root of the following easily differentiable function,

$$e^{0.5x} = 5 - 5x$$

Pick the best numerical technique, justify your choice and then use that technique to determine the root. Note that it is known that for positive initial guesses, all techniques except fixed-point iteration will eventually converge. Perform iterations until the approximate relative error falls below 2%. If you use a bracketing method, use initial guesses of $x_l = 0$ and $x_u = 2$. If you use the Newton-Raphson or the modified secant method, use an initial guess of $x_i = 0.7$. If you use the secant method, use initial guesses of $x_{i-1} = 0$ and $x_i = 2$.

Problem 2: Textbook problem 6.19 **(10 points)**
Solve this problem *by hand*.

6.19 You are designing a spherical tank (Fig. P6.19) to hold water for a small village in a developing country. 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 = 3 m, what depth must the tank be filled to so that it holds 30 m^3? Use three iterations of the Newton-Raphson method to determine your answer. Determine the approximate relative error after each iteration. Note that an initial guess of R will always converge.

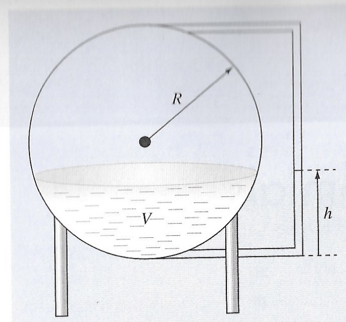


FIGURE P6.19

Problem 3: Solve this problem *by hand*. **(20 points)**
You are taking an exam where your calculator is allowed (but not a computer) and you need to compute the value of **ln (7)** to at least 5 significant figures. Unfortunately, you just realized that the natural logarithm function **ln** on your calculator no longer works (but every other function does…). Recast the problem as a root-finding problem and use the Newton-Raphson technique to estimate **ln(7)** with the desired accuracy. Devise the required stopping criterion (I gave you a formula in class that relates the number of significant figures to the stopping criterion) and use $x_0 = 2$ as your initial guess. Carry out your calculations with 10 significant figures.

Compare your final result to the (10 significant figure) exact value. What do you observe?

Problem 4: MATLAB programming **(60 points)**

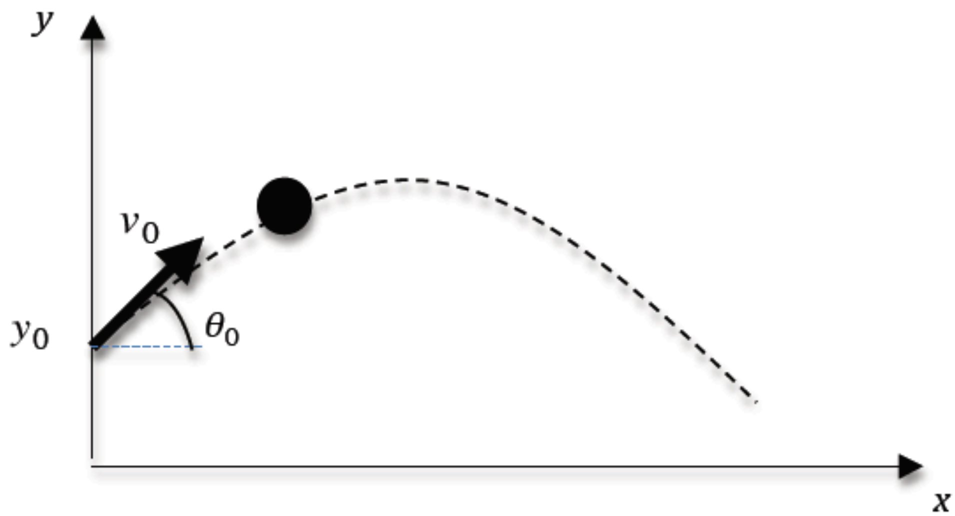
Write a generic (non-problem specific) MATLAB function that implements the Newton-Raphson method. The function should have the form `function xr = NewtonRaphson(f,dfdx,xi,es,imax)`, where `f` is a function handle that defines the root-finding problem $f(x)$=0, `dfdx` is a function handle for the derivative of f, `xi` is the initial guess, `es` is the stopping criterion for the relative approximate percent error defined in class, and `imax` is the maximum number of allowable iterations. The value of the root is returned in the output argument `xr`.

The function should print the root value, the number of iterations used and the value of the relative approximate percent error for the last iteration in the command window in a nicely formatted way. In addition, it should substitute the value of the root into the original function and display the result, to check if it makes sense. It should also warn the user in case it had to stop because $f'(x_i) = 0$ (and not because the root was found).

Problem to solve:
The trajectory of a thrown ball, as defined by its (x,y) coordinates, is given by

$$y = (\tan\theta_0)x - \frac{g}{2v_0^2\cos^2\theta_0}x^2 + y_0$$

where θ_0 is the launch angle, v_0 is the initial velocity, y_0 is the initial height from which it is thrown and g is the acceleration of gravity (see figure below).



Find the launch angle(s) (expressed in degrees) required to throw the ball to a catcher located 40 m from the thrower, if the catcher reaches it at a height of 1 m, given an initial velocity of 20 m/s and an initial height of 1.8 m Use g = 9.81 m/s.

- Plot the function to evaluate its root(s) graphically. Include this plot with your homework.
- Use your NewtonRaphson function to find the root(s); for the initial guesses, use the ***nearest integer value*** of the root based on your graphical evaluation, use a stopping criterion of 10^{-6} % and a maximum number of iterations of 1000.
- Finally, use the built-in function `fzero` with the following syntax (look it up in the documentation for details) and the same initial guesses to find the root(s) of the same problem:
  ```
  options = optimset('Display','iter');
  [root,FVAL,EXITFLAG,OUTPUT] = fzero(f,xi,options)
  ```
  This will display all the details of the steps used by `fzero` (make sure not to include a semi-colon after the `fzero` command).
- Comment on which method is the most efficient.

Instructions for Homework 4 submission:

1) Hand in paper copies of your handwritten solutions for problems 1,2 and 3.
For problem 4 include your plot, the results from your function and from `fzero` (a screen capture of the command window is OK) and a discussion (handwritten or typed).

2) Put your NewtonRaphson file in a zipped folder named HW4_LASTNAME_FIRSTNAME and upload to Canvas. **Make sure your folder has a .zip extension. Please do not use any compression software that would result in a different compressed format with another extension.**

Make sure you follow the formatting instructions and that you include comments in your program.

Please note the following, valid for all MATLAB assignments:

1. **Refrain from using the commands `clear`, `clc` and `close` (or any of their variations) in any of your m-files, as they interfere with the grading process.**

2. **When your Newton-Raphson code is run, it should produce only what is asked for (in blue in problem 4) in the assignment. Please make sure to suppress any unnecessary intermediate result in any form: no other figure, curve or value printed in the command window should appear.**

You will lose points if you ignore these instructions.

Please note that Canvas will only accept a file with a ZIP extension: all of your M-files must be in that zipped folder. If we cannot open your files because you failed to follow these instructions, you will get a zero for this part of the assignment. Finally, please keep in mind the policies about late assignments and collaboration outlined in the syllabus, and copied here for your convenience:

- Late assignments: homework and projects will be accepted up to 24 hours after the deadline with a 50% penalty. This policy will be strictly enforced and no submission will be accepted after 24 hours. Hard copies of homework will be due at the end of class. ***It is your responsibility to make sure that you have successfully uploaded and submitted all the required files in the required format to Canvas on time***: please double-check to avoid having to re-submit your work after the due date. The only acceptable proof that you have submitted your work on time is the time stamp of your Canvas submission. ***Files will not be accepted by email even if they show a "last modified date" that is before the due date.***

- Collaboration: students may *discuss* their assignments with each other, but homework and projects must be *completed individually* by each student. ***You must turn in your own work***. Copying someone else's work and submitting it as your own will not be tolerated. In particular, ***for Matlab assignments***, this policy means that students can discuss aspects such as the general approach to solve a problem or the syntax of a specific command, but ***they should not look at each other's codes***. If it is suspected that this has occurred, you will be reported to the Dean of Students for an honor code violation.