**CS 514/ Math 514 Numerical Analysis**
Spring 2018
Homework 3
Given: Feb. 20, 2018; Due: March 6, 2018 11:45 A.M. (15 minutes before class)

You are permitted to discuss these problems with **one** other student, and you should indicate the name of your collaborator in your answer to the first problem in this set. If you worked on your own, please indicate this on your answer. You should write up your own solution, and you are not permitted to either share a written copy of your solution or copy some one else's written solution. Similarly with programs, you are not permitted to share your code with another student. If you need help with debugging your code, you should describe what the problem is without sharing or showing your code to another student. If you are stuck on a problem or on a code, you could ask the TAs for help on Piazza. Please do not discuss solutions to the HW problems on Piazza before they are graded.

If you copy someone else's work, or let your work be copied, you will get zero points for the entire HW, a loss of one letter grade in the course, and you will be reported to the Dean of Students. I follow the course policies described by Professor Gene Spafford (of Purdue Computer Science department) at the URL `http://spaf.cerias.purdue.edu/~spaf/cpolicy.html` If you are not familiar with this policy, please read it! Please submit your assignments using Blackboard, preferably using either a Tex-generated or Word document.

0. If you collaborated with a student in answering these questions, please write down your collaborator's name. If you did not collaborate with anyone, please indicate that too.

1. **15 points**
   This problem is derived from the text book Problem 3.6.11. One way to compute a multiple root $x^*$ of a smooth nonlinear function $f(x)$ using Newton's method is to consider the function $f(x)/f'(x)$.

   (a) Show that the function $f(x)/f'(x)$ has a simple root at $x^*$.

   (b) Show that the Newton iteration yields

   $$x_{k+1} = x_k - \frac{f(x_k)f'(x_k)}{f'(x_k)^2 - f(x_k)f''(x_k)}.$$

   (c) Write a program implementing this iteration, and use it to compute the multiple root of $f(x) = (x-1)^2 \exp(x)$. Try different initial guesses, 0, 2, 3 and report the number of iterations your method takes and the root that it finds. Explain what you observe.

2. **15 points**

   (a) The function
   $$f(x) = -\frac{1}{(x-0.3)^2+0.01} - \frac{1}{(x-0.9)^2+0.04}$$
   has two local minima in the intervals $[0.2\ 0.4]$ and $[0.8\ 1.0]$. Find the minimizers and the minimum function values using the `fminbnd` function in Matlab. Set the tolerance with which the minimizer is computed to be $10^{-8}$, and use the `optimset` parameters to display the procedure used in each iteration in the algorithm. You need to submit the values of the iteration number, $x$, $f(x)$, and the minimization procedure used, in the form of a table.

   (b) Implement the Golden Section search algorithm that we discussed in class to compute the minimizer of a unimodal function. Use it to compute the two minimizers of the function $f(x)$ considered in the previous part, again for the two intervals $[0.2\ 0.4]$ and $[0.8\ 1.0]$. You should set the tolerance such that the length of the initial interval is reduced by a factor of $10^{-3}$. Submit your code, and a table of values of the iteration number, $x$, and $f(x)$, as you did for the previous part.

3. **5 points**
   This problem concerns Algorithmic Differentiation (AD). In the file `valder.m`, look at the definition of $u^v = \exp(v * \log(u))$, when both $u$ and $v$ are of the `valder` datatype. Let $x = \log(u)$, $y = v * x$, and $z = \exp(y)$. State the values of the `val` and `der` properties of each intermediate variable $x$, $y$ and $z$, and explain how $u^v$ is correctly obtained.

4. **10 points**
   In this problem we will use an Automatic Differentiation (AD)-enabled Newton's method for a nonlinear equation in one variable. Recall that the code for computing Newton's method using AD was given to you in the Matlab function `newtonad.m`, which you can use to solve this problem.

   The vertical distance $y$ that a parachutist falls before opening the parachute is
   $$y = \log(\cosh(t\sqrt{gk}))/k,$$
   where $t$ is the elapsed time in seconds, $g = 9.8065\ \mathrm{m/s^2}$ is the acceleration due to gravity, and $k = 0.00341\ \mathrm{m^{-1}}$ is a constant related to air resistance. Use the `newtonad.m` code to determine the time elapsed when the parachutist would have fallen a distance of $1000\ \mathrm{m}$. (You can check your answer using the `fzero` function.)

5. **20 points**
   Now we consider a problem that uses AD to solve a nonlinear system of equations in three variables.

(a) First write a function `newtonad3d.m` which computes the function values and the $3 \times 3$ Jacobian matrix corresponding to a generic nonlinear vector function `func3d.m`, with three component functions in three variables. The input to `newtonad3d.m` is the initial guess of the values of the three variables, and it makes use of the nonlinear function `func3d.m`. Your program should use the `valder` datatype to compute the function values and the Jacobian using AD.

Studying how the given function `gradad2.m` computes the gradient of a scalar function should help you, since a Jacobian matrix is a collection of the gradients of the component functions. Set the Newton iteration to terminate when the 1-norm of the update to the previous iterate is less than $10^{-8}$. (This is the vector $p_k$ in Newton's method for systems given on page 254 of the text.) Submit your documented code; it will be evaluated for correctness and (reasonable) efficiency.

(b) Use the code you have written to solve the system of nonlinear equations

$$\begin{aligned} 16x^4 + 16y^4 + z^4 &= 16, \\ x^2 + y^2 + z^2 &= 3, \\ x^3 - y &= 0, \end{aligned}$$

using the initial guess $[1; 1; 1]$. Report the solution you compute, the function values at the computed solution, and the number of iterations. Report also the norm of the update to the previous iterate (again the vector $p_k$ as earlier) as a function of the iteration number. What is the observed convergence rate of the Newton method? Use the norm of the update to answer this question.

(c) Lorenz derived a system of ordinary differential equations to describe buoyant convection in a fluid as a model for atmospheric circulation. At steady state, the convective velocity $x$, the temperature gradient $y$, and the heat flow $z$, with the use of the typical values of the parameters in the system, satisfy the nonlinear equations

$$\begin{aligned} -10x + 10y &= 0, \\ 28x - y - xz &= 0, \\ xy - (8/3)z &= 0. \end{aligned}$$

Use your AD-enabled Newton method to compute *three* distinct roots of the system. Report your starting guesses, the roots, the number of iterations, and the convergence rate to the solution, for each root.

**Hint**: It would help to estimate the solutions to find the initial guesses.