CS 322: Problem Set 6

Due: Friday, August 2, 2002 (In Lecture)

The policies for this homework assignment are as follows:

- **Because this is due on the last day of class, no homework will be accepted past 5:00 P.M. on the day it is due. Assignments submitted between 2:15 P.M. and 5:00 P.M. that day will be considered late. Solutions will be given to students once they've submitted their assignment.**
- You may work with at most one other person on both questions. Consult the course website for the Academic Integrity rules.
- Problem sets will be weighted equally in determining your final grade. The number of questions or total number of points on a given problem set is irrelevant.
- Writing will be graded on content, as well as on grammar, spelling, punctuation, etc.
- Mathematical exercises will be graded on the overall set-up of the problem as well as correctness.
- Points will be deducted on the Matlab questions for poorly commented code and inefficient code/redundant computations.
- Submit your assignment in ONE part. At the top of the first page, write your name, the course number, the problem set number, your e-mail address, your student ID number, and the date. This information should be repeated for the second partner (if applicable).

1. (10 points) ***Down in the Valley...***
   Consider minimizing the function $f(x, y) = (x-1)^2 + 10(x^2 - y)^2$. This function has a unique minimum at $(x, y) = (1, 1)$. However, this function often proves difficult for optimization software to minimize due to the steep sides of the valley in which the minimum lies. On this exercise, we will minimize this function using both the Steepest Descent Method and Newton's Method.

   First, implement the Steepest Descent Method. See page 310 in your book for the method. You will note that the second bullet says to "Inexpensively choose λ_c so $f(t_c + \lambda_c s_c)$ is sufficiently less than $f(t_c)$." This doesn't correspond to a unique type of implementation. Here is how we will choose λ_c. Define $f_c(\lambda) = f(t_c + \lambda s_c)$. Assume that $\lambda_2 > \lambda_1$ and either $f_c(\lambda_2) < f_c(0)$ or $f_c(\lambda_1) < f_c(0)$. (In practice this will mean cutting the interval in half up to a maximum number of times (20) to ensure that this property occurs.) Define $q(\lambda)$ to be the quadratic that interpolates $(0, f_c(0)), (\lambda_1, f_c(\lambda_1))$, and $(\lambda_2, f_c(\lambda_2))$. Let λ_c be the minimizer on $[0, \lambda_2]$ of $q(\lambda)$. Your implementation should be a Matlab function called SteepestDescentStep.m and should satisfy the following function prototype:

```
function [tnew,fnew,gnew] = SteepestDescentStep(tc,fc,gc,Lmax,fname,gname)
%
% Input:
% tc = a column vector that corresponds to the current position
% fc = f(tc)
% gc = g(tc)
% Lmax = maximum step length
% fname = name of function to evaluate f
% gname = name of function to evaluate g
%
% Output:
% tnew = a column vector with the new position
% fnew = f(tnew)
% gnew = g(tnew)
%
% The Steepest Descent Step is computed according to the specifications above.
```

Second, implement Newton's Method for minimizing a function of several variables. Your function should be called Newton.m and should satisfy the following specification:

```
function [tnew] = Newton(tc,gname,Hname)
%
% Input:
%
% tc = current point
% gname = name of function to evaluate gradient
% Hname = name of function to evaluate Hessian
%
% Output:
% tnew = Newton step.
```

Now, you're ready to write a script that will employ both the Steepest Descent Method and Newton's Method to minimize $f(x, y)$.

Your script should be called Valley.m and should do the following:

- Make a detailed contour plot (with about 150 contour lines) of this function on $[-1, 1.5] \times [-1, 1.5]$. Use the hsv colormap and the contourf command. This plot should be in a figure window by itself.
- Using $(-0.5, 1.5)$ as a starting point, a maximum of 500 iterations, and a maximum step length of 1, execute the Steepest Descent Method until either the maximum number of iterations is reached or the function value becomes less than or equal to $1e-8$. (This is an acceptable stopping heuristic, since we know that the minimum value of the function is 0.) You should store the values of t from each iteration. After SteepestDescentStep returns, you should make the above contour plot in a second figure window and then superimpose the successive Steepest Descent iterates on this plot using both 'k-' and 'k.'.
- Using the same starting point as above and a maximum of 500 iterations, execute Newton's using the same stopping criteria. Again, store all of the values of t from each iteration. After Newton returns, make the above contour plot in a third figure window and superimpose the successive Newton iterates on this plot using both 'k-' and 'k.'.

If any of the iterates go outside the range of the contour plot (for either the Steepest Descent Method or Newton's Method), then you should enlarge the contour plot so that the iterates are all superimposed on the countour plot.

Also, note that you will need Matlab functions to evaluate the function, the gradient, and the Hessian at a specific point. You should compute the gradient and Hessian on paper. This is part of what you will turn in for this question.

Finally, write a short (1 paragraph) summary discussing the relative performances of the Steepest Descent Method and Newton's Method as applied to this particular objective function.

Turn in your computations of the gradient and Hessian, your Matlab script, all of your Matlab functions, and your three plots. It is OK to turn in black and white plots.

2. (10 points) Initial Value Problems

(a) ***A Chemical Reaction***

The irreversible chemical reaction in which two molecules of solid potassium dichromate ($K_2Cr_2O_7$), two molecules of water (H_2O), and three atoms of solid sulfur (S) combine to yield three molecules of the gas sulfur dioxide (SO_2), four molecules of solid potassium hydroxide (KOH), and two molecules of solid chromic oxide (Cr_2O_3) can be represented symbolically by the stoichiometric equation:

$$2K_2Cr_2O_7 + 2H_2O + 3S \rightarrow 4KOH + 2Cr_2O_3 + 3SO_2.$$

If n_1 molecules of $K_2Cr_2O_7$, n_2 molecules of H_2O, and n_3 molecules of S are originally available, the following differential equation describes the amount $x(t)$ of KOH after time t :

$$\frac{dx}{dt} = k\left(n_1 - \frac{x}{2}\right)^2 \left(n_2 - \frac{x}{2}\right)^2 \left(n_3 - \frac{3x}{4}\right)^3,$$

where k is the velocity constant of the reaction. If $k = 6.22 \times 10^{-19}, n_1 = n_2 = 2 \times 10^3$, and $n_3 = 3 \times 10^3$, how many units of potassium hydroxide will have been formed after 0.2s?

Answer this question by writing a script, Reaction.m, that uses the fourth-order Runge Kutta method with $h = 0.001$. For the purposes of this question, you should write your own fourth-order Runge Kutta Method; call this function RK.m. To do so, you may modify RKStep.m from your book. You will also need to write a function that evaluates $\frac{dx}{dt}$. Make an appropriately labeled plot of $x(t)$ vs. time.

Hand-in Reaction.m, RK.m, your derivative function, and your plot.

(b) ***Predator and Prey***

Consider the problem of predicting the population of two species, one of which is a predator, whose population at time t is $x_2(t)$, feeding on the other, which is the prey, whose population is $x_1(t)$. We will assume that the prey always has an adequate food supply and that its birth rate at any rime is proportional to the number of prey alive at that time; that is, birth rate (prey) is $k_1x_1(t)$. The death rate of the prey depends on both the number of prey and predators alive at that time. For simplicity, we assume death rate (prey) $= k_2x_1(t)x_2(t)$. The birth rate of the predator, on the other hand, depends on its food supply, $x_1(t)$, as well as on the number of predators available for reproduction purposes. For this reason, we assume that the birth rate (predator) is $k_3x_1(t)x_2(t)$. The death rate of the predator will be taken as simply proportional to the number of predators alive at the time that is, death rate (predator) $= k_4x_2(t)$.

Since $x_1'(t)$ and $x_2'(t)$ represent the change in the prey and predator poulations, respectively, with respect to time, the problem is expressed by the system of nonlinear differential equations

$$x_1'(t) = k_1x_1(t) - k_2x_1(t)x_2(t)$$

and

$$x_2'(t) = k_3x_1(t)x_2(t) - k_4x_2(t).$$

Solve this system for $t \in [0, 4]$ and assume that the initial population of the prey is 1000 and of the predators is 500 and that the constants are $k_1 = 3, k_2 = 0.002, k_3 = 0.0006$, and $k_4 = 0.5$. Make a plot of the solutions to this problem, plotting both populations with time, and describe the physical phenomena represented. Is there a stable solution to this population model? If so, for what values x_1 and x_2 is the solution stable? Write a script, PredatorPrey.m, to solve this problem taking advantage of ode45. Note that you will also need to write a function which evaluates both x_1' and x_2'.

Hand-in all of your code, the appropriate plots, and concisely written answers to the above questions.

(c) ***Well-Posedness***

Let $D = \{(t, y) | 0 \leq t \leq 2, -\infty < y < \infty\}$ and consider the IVP

$$\frac{dy}{dt} = y - t^2 + 1, 0 \leq t \leq 2, y(0) = 0.5.$$

Explain why this problem is well-posed and why this is a desirable property of IVP's. Turn-in your mathematical argument and explanation.