Name:_____

ChBE 2120, Numerical Methods, Paravastu Section, Fall 2015

Exam 2: 100 points possible, time: 70 min

Problem 1 (40 points total):

(a) (15 points) Have you ever wondered how computers calculate the square roots of positive real numbers? The calculation could described as a root finding problem. Write the Matlab function CalculateSqrt declared below, without using the Matlab sqrt function or the ^ operator. Your function must use the bisection method function declared below to perform the calculation and output the numerical value of the square root of the input variable *x*. Assume that we want the positive square root and that $x \geq 0$. This function should work with numerical (not symbolic) input. Assume that the function BisectionMethod works as implied by the declaration with negligible numerical error (code is not shown). Your function's choice of initial bracket must be appropriate for any $x \geq 0$.

Hints: i) If we want to calculate $y = \sqrt{x}$, then the desired value of y is the root of the equation $y^2 - x = 0$.
ii) If $x > 1$, then $x > \sqrt{x} > 0$.
iii) If $x < 1$, then $x < \sqrt{x} < 0$.

function [root] = BisectionMethod(f, xGuessLow, xGuessHigh)
%The variable f must represent an anonymous function of one variable. The guesses xGuessLow and xGuessHigh must be above and below the desired root of f, respectively. The output, root, will contain only a single number representing the value of x at which f(x) is equal to 0.
%----------code not shown, but assume the function works----------

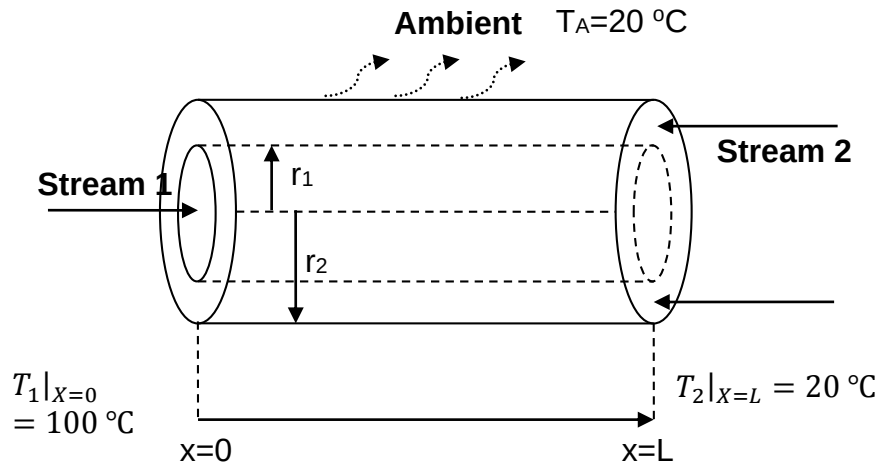
function [sqrtx] = CalculateSqrt(x)
%outputs the numerical value of the square root of x.

end

(b) (10 points) Write a Matlab function that calculates the numerical Hessian matrix for a function. The inputs should be an anonymous function of two variables and numerical values of those two variables. The function should output a matrix of numbers.

(c) (15 points) Perform one iteration of the steepest descent algorithm for optimization of the function, $f(x,y,z) = (1-x)^2 + (1-y)^2 + 0.1[(z-y^2)^2 + (y-x^2)^2]$. Use the initial guess $x = y = z = 0$. Show the equation you would need to solve in order to calculate the optimal step size for this iteration. Without solving for the optimal step size, indicate how you would calculate the next guess.

Problem 2 (40 points total): Consider counter current core-shell heat exchanger depicted below. The hot fluid of Stream 1 flows in the core channel and is cooled by the coolant flowing in the shell (annular) channel (Stream 2). The diameters of the core and shell are $r_1 = 1$ and $r_2 = 2$, respectively. Heat transfer between Streams 1 and 2 is represented by a heat transfer coefficient $h_1 = 1.6$. In addition, there is heat loss from Stream 2 to the surrounding air, characterized by a heat transfer coefficient $h_2 = 0.1$. Streams 1 and 2 enter the heat exchanger at $T_1|_{X=0} = 100$ ℃ and $T_2|_{X=L} = 20$ ℃. The length of the heat exchanger is 8 and the ambient temperature is 20℃.



a) (10 points) Formulate a system of differential equations for temperatures T_1 and T_2 as functions of x at steady state. Assume that the temperatures depend only on x.

b) (20 points) Applying the Finite Difference method, setup a matrix equation (linear system) that could be used to solve the differential equations you derived in (a). If you were unable to do part (a), use the alternative system: $\frac{dT_2}{dx} = 4(T_1 - T_2) + 7T_2$; $\frac{dT_1}{dx} = 3(T_1 - T_2) + 4T_2$. Choose $\Delta x = 2$. Approximate derivatives using the simplest (least accurate) forward finite difference formula.

c) (10 points) Using the functions defined in the headers below (assume they work), write Matlab code to solve the same problem using the Shooting Method.

Function Header 1: function [Tprime] = InitialValueODE(x, T)
Function Header 2: function [tSolution, Ysolution] = ODERungeKutta4(Yprime, tRange, Y0, h)
%Y = [T2; T1]
Function Header 3: function [xRoot] = SecantMethod(x0, x1, f, EaMax)

Problem 3 (20 points total):

a) (10 points) Determine the equations and count the number of unknowns that would be established for the Boundary Value Problem below, if it were solved using the Finite Differences method with an increment $\Delta x = 0.25$. You need not put the linear system into matrix problem.

$\frac{d^2y}{dx^2} + \frac{dy}{dx} = x^2, y|_{x=0} = 1, \frac{dy}{dx}\Big|_{x=1} = 0$

Approximate derivatives using the following finite difference formulas:

$\frac{dy}{dx} \cong \frac{y_{i+1} - y_{i-1}}{2\,\Delta x}, \frac{d^2y}{dx^2} \cong \frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta x^2}$

b) (10 points) While using the Finite Difference Method to solve a different Boundary Value Problem, you derive the formulas below. Write Matlab code to setup and solve this linear system for an arbitrary step size, h. A, B, C, and D are known constants; assume that corresponding Matlab variables are already assigned with the correct numbers. Note that your unknowns are $Y_{-1}, Y_0, Y_1, Y_2, \ldots, Y_n$, and Y_{n+1}.

$\left(\frac{A}{h^2} + \frac{B}{2h}\right) Y_{i-1} - \frac{A}{h^2} Y_i + \frac{B}{2h} Y_{i+1} = 0$, for $i = 0$ through n in steps of 1.

$\frac{A}{h^2} Y_{-1} - BCY_0 - \frac{B}{2h} Y_1 = D$

$Y_{n-1} - Y_{n+1} = 0$