Assignment

Note : Each question carries 4 marks, making a total of 20 marks for this assignment. You have to submit within 2 weeks, that is, by September 7.

Q1a) Write your own program to generate an array of random numbers between -1000 and +1000. Pick some random number generation formula.

b) Write your own program to sort (arrange in the increasing order) an array (vector) of integers of any arbitrary length. The array elements should be chosen as random numbers between -1000 and +1000.

c) Find the largest and smallest element, the mean, median and standard deviation of this array of elements.

Q2a) Write a program to obtain the first three energy eigen values of a particle in a finitely-deep potential well, by finding the first three roots of appropriate functions as discussed in the class. Use Newton-Raphson method.

b) Keep increasing the depth of the well (increase β) and make a table of these eigenvalues and see that they approach the eigenvalues of the particle in an infinite well.

Or

Write a program to calculate the time taken by a body to come back to the ground, when thrown up into the air. Include damping force for air resistance. Choose the damping constant to be such that the maximum damping force varies between 0 t0 20% of the body's weight. See how the time taken varies with the changing value of the damping constant. Choose the initial velocity appropriately.

Q3a) Write a function that will do a polynomial fit to a set of data points. The degree of the polynomial should be a variable to be input to the function. You may use the MATLAB built-in function for matrix inversion. Note that this problem was discussed in the class.

b) As a particular case, generate your data points to be within an error bar of 10% of the values of the function $y = x^2$ and fit these data with a quadratic polynomial. The values of the error at each data point should be a random number between $\pm 10\%$ of the actual value on the parabola.

c) Do a similar fit for the points generated about the curve (about means, you have to add error as described above)

$$y = x^3/(e^x - 1)$$

Here you have to change the degree of the polynomial to find a good fit.

Q4) Consider quadratic spline interpolation, where two successive data points (x_i, y_i) & (x_{i+1}, y_{i+1}) are joined by a second-degree polynomial

$$P_i(x) = a_i(x - x_i)^2 + b_i(x - x_i) + c_i \quad i = 1, 2, \ldots\ldots, N-1$$

a) With the condition that the first derivative of one polynomial matches with the first derivative of the subsequent polynomial at the common boundary, derive the following equations

i) $a_i = (y_{i+1} - y_i)/h^2 - b_i/h \qquad i = 1, 2, ..., N-1$

ii) $b_i + b_{i+1} = 2(y_{i+1} - y_i)/h \qquad i = 1, 2, ..., N-2$

b) As the additional condition, put $P_1''(x_1) = 0.$ ... Hence, show that

$$b_1 = (y_2 - y_1)/h$$

c) With b_1 known, it is straight forward to determine all the other b_i (i = 2, 3, ..., N-2) from eqs (ii) and from them, all a_i (i = 1, 2, ..., N-1) from eqs. (i). Write a MATLAB function to encode the above procedure and test it on some known functions and compare it with cubic-spline interpolation through graphs.

Q5a) Given a set of N data points $\{(x_i, y_i)\}_{i=1,.....,\ N}$ with equal spacing of the x-coordinates (spacing is h), write a function to compute the Newton's forward differences. The function should take as input, the coordinates of the points and should return the forward differences of all orders.

b) Write a program to carry out a $(N-1)^{th}$-degree polynomial interpolation by using the Newton's form of the interpolating polynomial. Take as your data points the sampled values of known functions and compare the graphs of the interpolating polynomials as well as the actual functions.

****