Paul Buttles

08/28/2020

PHYS 331, HW2

1b.) f1(x) roots:

x = 0.2607421875 at 10^-3 tolerance over 11 iterations

x = 0.26076221466064453 at 10^-6 tolerance over 21 iterations

x = 0.2607617224684873 at 10^-12 tolerance over 41 iterations

f2(x) roots:

x = 0.5 at 10^-3 tolerance over 2 iterations

x = 0.5 at 10^-6 tolerance over 2 iterations

x = 0.5 at 10^-12 tolerance over 2 iterations

f3(x) roots:

x = 0.3076171875 at 10^-3 tolerance over 11 iterations

x = 0.30831050872802734 at 10^-6 tolerance over 21 iterations

x = 0.30830988618345145 at 10^-6 tolerance over 41 iterations

f4(x) did not return any roots as it has an asymptote at x=1/2, thus giving a divide by zero error. Additionally, it asymptotically approaches 0 but never reaches it, within the [-1,1] range or otherwise.

The roots found for f1(x) and f2(x) are valid and make sense over the [-1,1] interval. The root found for f3(x) is valid but does not make sense as there are many roots within the [-1,1] interval, so it simply returned the rightmost root while ignoring the others. A solution for this is to pick a particular root that is desired, and readjust the interval to target in on that root specifically. No root was found for f4(x) as it is bound to fail for the reasons explained above. There is no solution for this as the problem lies with the function itself, although a separate piece of code could be added to identify asymptotes. This could be done by tracking the expansion of the value of the function evaluated at the new midpoints. If they begin increasing at a rate faster than some threshold rate, the function terminates and identifies an asymptote at the final midpoint.

4a.) The volume of a cube takes the form V=s^3 where V is volume and s is side length. In this instance, 390=x^3. Because we are root finding, this is equivalently 390-x^3. Thus f(x)=390-x^3.

4b.) Newton-Raphson functions take the form x_(n+1) = x_n – f(x)/f'(x). In this case, f(x) = 390-x^3 and f'(x)=-3x^2. So the iterative function is x_(n+1) = x_n – (390-(x_n)^3)/(-3(x_n)^2), which can be simplified to x_(n+1) = (2/3)x_n + 130/((x_n)^2). Using a four function calculator, this would be performed as:
2 * x_n / 3 + 130 / x_n / x_n. This takes a minimum of five operations, multiplication, division, addition, division, division.

4c.) The closest integer to the true answer is 7cm, as 390 lies between 7^3=343 and 8^3=512, closest to 343, and hence the true answer is closest to 7.

	x_n	V_n
X_1	7	343
X_2	7.319727891156462	392.17942889127505
X_3	7.306168768922235	390.00403469859157
X_4	7.3061435741496865	390.0000000139134

Because 390.0000000139134 is within 0.003 of 390, the approximation is complete at the fourth iteration, where x_n = 7.3061435741496865.

4d.) The equation sigma_v = 3(sigma_L)(L^2) is given. Sigma_v is given to be 0.003cm^3, and L was found to be 7.3061435741496865 cm. Solving for sigma_L using the calculator is done by:
sigma_L = sigma_v / 3 / L / L. Thus sigma_L = 1.8733701471510356e-05 cm.

4e.) Using the rf_bisect function from problem 1, it took 16 iterations to reach this same tolerance. This is four times the number of iterations, meaning it is far less efficient and would take significantly longer with a four figure calculator. N-R is four times as efficient as the bisection method in this instance, and in most cases is more computationally efficient.