HW 2

(MATH/CS 375)

Jacob Hurst

02/13/18

**Problem 1.** (35 points) Consider the scalar equation f(x) = 2x − 1 − sin x = 0.

1. (10 points) Find a closed interval [a, b] on which the equation has a root r, and use the Intermediate Value Theorem to prove that r exists.
2. (10 points) Prove that r from (a) is the only root of the equation (on all of R).

**Problem 1 (Continued).** (35 points) Consider the scalar equation f(x) = 2x − 1 − sin x = 0.

1. (15 points) Write a Matlab function, function r=bisect(f,a,b,p), that finds a root r of a function f in the interval [a, b] to p correct decimal places, if one exists, and prints an error message otherwise. Use your bisect function to approximate the root r of the function f(x) to p = 8 correct decimal places. Include your Matlab function, the calling script (i.e. the command that calls bisect function), the resulting final approximation, and the total number of iterations.

Code:

function root = bisect(f, a, b, p)

if(a > b)

error('Invalid interval.');

end

n = 0;

a0 = a;

b0 = b;

root = (a + b)/2;

check = (b - a)/2;

tol = 1\*10^(-p);

syms x;

f0 = subs(f, x, a);

f1 = subs(f, x, b);

% by IVT if f(a) < 0 & f(b) > 0 there must be a root in [a,b]

% if root exists, then this means f(a)\*f(b) < 0 since neg\*pos=neg.

if(f0\*f1 >= 0)

error('No roots exist for given function in interval.');

end

while abs(check) > tol

fa = subs(f, x, a);

fc = subs(f, x, root);

if(fa\*fc < 0)

b = root;

else

a = root;

end

n = n + 1;

root = (a + b)/2;

check = (b0 - a0)/2^(n + 1);

end

fprintf('Root = %.15f, found in n = %d steps.', root, n);

end

Output:

>> syms x

>> f = 2\*x - 1 - sin(x)

>> bisect(f, 0, 1, 8)

Root = 0.887862212955952, found in n = 26 steps.

**Problem 2.** (65 points) Consider the scalar equation f(x) = x − 4 sin(2x) − 3 = 0.

1. A close up of text on a white background

   Description generated with high confidence(10 points) Plot f(x). All the zero crossings should be in the plot. How many are there?

There are 5 zero crossings for the function; r0≈-0.89, r1≈-0.54, r2≈1.73, r3≈3.16, r4≈4.51.

**Problem 2 (Continued).** (65 points) Consider the scalar equation f(x) = x − 4 sin(2x) − 3 = 0.

1. (15 points) Write a Matlab program that computes the roots using the fixed point iteration method for finding the fixed points of g(x) = − sin(2 x) + 5 x/4 − 3/4: xi = − sin(2 xi−1) + 5 xi−1/4 − 3/4, i = 1, 2, 3, . . . . Use a stopping criterion that gives an answer with 10 correct digits. Include your code. Find empirically (by numerical experiments) which roots can be found with the above iteration. Give a theoretical explanation (without proof).

Code:

function root = fpi(initial, p, N)

syms x;

g = (-sin(2\*x))+((5\*x)/4)-(3/4);

xn = initial;

xn1 = 0;

n = 0;

tol = 1\*10^(-p);

error = (xn1 - xn) + 2\*tol;

while(abs(error) >= tol)

if(n >= N)

break; %error('Reached stopping criteria of n.');

end

xn1 = double(subs(g, x, xn));

error = (xn1 - xn);

n = n + 1;

xn = xn1;

end

root = xn1;

fprintf('Root = %.15f, found in n = %d steps.', root, n);

end

Output:

>> fpi(-1, 10, 100)

Root = -11691525728.540550000000000, found in n = 100 steps.

>> fpi(-.8, 10, 100)

Root = -0.544442400708549, found in n = 23 steps.

>> fpi(-.6, 10, 100)

Root = -0.544442400722548, found in n = 19 steps.

>> fpi(0, 10, 100)

Root = -0.544442400708311, found in n = 23 steps.

>> fpi(1, 10, 100)

Root = -0.544442400663504, found in n = 20 steps.

>> fpi(2, 10, 100)

Root = 3.161826486593999, found in n = 78 steps.

>> fpi(3, 10, 100)

Root = 3.161826486511945, found in n = 76 steps.

>> fpi(4, 10, 100)

Root = 3.161826486517765, found in n = 76 steps.

>> fpi(5, 10, 100)

Root = 10514022783.168577000000000, found in n = 100 steps.

Theoretical Explanation:

**Problem 2 (Continued).** (65 points) Consider the scalar equation f(x) = x − 4 sin(2x) − 3 = 0.

1. (10 points) Write a Matlab program that computes the roots with Newton’s method with 10 correct digits. Include your code.

Code:

function root = nm(initial, p, N)

syms x;

f = x-4\*sin(2\*x)-3;

df = -8\*cos(2\*x)+1;

xn = initial;

xn1 = 0;

n = 0;

tol = 1\*10^(-p);

error = (xn1 - xn) + 2\*tol;

while(abs(error) >= tol)

if(n >= N)

break; %error('Reached stopping criteria of n.');

end

xn1 = double(xn- (subs(f,x,xn))/((subs(df,x,xn))));

error = (xn1 - xn);

n = n + 1;

xn = xn1;

end

root = xn1;

fprintf('Root = %.15f, found in n = %d steps.', root, n);

end

Output:

>> nm(-.8, 10, 100)

Root = -0.898356581545585, found in n = 6 steps.

>> nm(-.6, 10, 100)

Root = -0.544442400680698, found in n = 5 steps.

>> nm(0, 10, 100)

Root = -0.544442400680698, found in n = 6 steps.

>> nm(1, 10, 100)

Root = -0.898356581545585, found in n = 10 steps.

>> nm(2, 10, 100)

Root = 1.732069502143588, found in n = 5 steps.

>> nm(3, 10, 100)

Root = 3.161826486551946, found in n = 4 steps.

>> nm(4, 10, 100)

Root = 3.161826486551946, found in n = 6 steps.

>> nm(5, 10, 100)

Root = 4.517789514180033, found in n = 5 steps.

**Problem 2 (Continued).** (65 points) Consider the scalar equation f(x) = x − 4 sin(2x) − 3 = 0.

1. (d) (10 points) Write a Matlab program that computes the roots with the secant method with 10 correct digits. Include your code.

Code:

function root = secant(guess1, guess2, p, N)

syms x;

x\_old = guess1;

x\_new = guess2;

f = x-4\*sin(2\*x)-3;

f\_old = double(subs(f, x, x\_old));

f\_new = double(subs(f, x, x\_new));

n = 0;

tol = 1\*10^(-p);

while abs(f\_new) >= tol

if(n >= N)

break; %error(‘Reached stopping criteria of n.’);

end

df = f\_new - f\_old;

dx = x\_new - x\_old;

x\_old = x\_new;

f\_old = f\_new;

x\_new = x\_new - f\_new\*dx/df;

f\_new = double(subs(f, x, x\_new));

n = n + 1;

end

root = x\_new;

fprintf('Root = %.15f, found in n = %d steps.', root, n);

end

Output:

>> secant(-.9, -.7, 10, 100)

Root = -0.898356581551056, found in n = 5 steps.

>> secant(-.6, -.4, 10, 100)

Root = -0.544442400680688, found in n = 6 steps.

>> secant(0, 2, 10, 100)

Root = 1.732069502143588, found in n = 6 steps.

>> secant(1, 2, 10, 100)

Root = 1.732069502143588, found in n = 5 steps.

>> secant(2, 4, 10, 100)

Root = 3.161826486551679, found in n = 6 steps.

>> secant(4, 5, 10, 100)

Root = 4.517789514180033, found in n = 6 steps.

**Problem 2 (Continued).** (65 points) Consider the scalar equation f(x) = x − 4 sin(2x) − 3 = 0.

1. (20 points) Now you will compare the convergence properties of the three methods in (b), (c), and (d). Choose a root (say r∗ ) that all methods can find and use the same initial guess x0 for all three methods. For each method plot the error Ei = |r∗ − xi | as a function of iterations i = 1, 2, . . . n. Note that the total number of iterations n may be different for different methods. Plot all three curves in the same figure. Use semilogy to get a logarithmic scale on the y-axis. Use different line styles/markers/colors for different curves. The plot should have labels on each axis and a legend to describe each curve. Use large fonts. From your figure you should be able to see how much faster Newton’s method converges. NOTE: Since the exact value of r∗ is not available, you can compute and use a very accurate approximation, say with 15 correct digits, instead of r∗ . You have to compute this very accurate approximation before you can compute the error. Use the same r∗ for all three methods

A close up of a device

Description generated with high confidence