For problems 1 though 6, use the function $f(x) = x^5 - 3x^2 + 1$.

Problem 1.

Use the intermediate value theorem to show that f has a root in each of the intervals: $(-1, 0), (0, 1), (1, 2)$. Label these roots r_1, r_2, r_3.

Solution

The Intermediate Value Theorem (Theorem A.1, page 491 in course text) states:

Theorem 0.1. *Let $f(x)$ be a continuous function on the interval $a \leq x \leq b$. Let*

$$M = \max_{a \leq x \leq b} f(x) \quad m = \min_{a \leq x \leq b} f(x)$$

Then for every real number y satisfying $m \leq y \leq M$, there is at least one point c in the interval $[a, b]$ such that $f(c) = y$.

The function $f(x)$ defined above is a polynomial, so it is a continuous function. Observe that

$$\begin{aligned} f(-1) &= -1 - 3 + 1 = -3 < 0, \\ f(0) &= +1 > 0, \\ f(1) &= 1 - 3 + 1 = -1 < 0, \\ f(2) &= 2^5 - 3(2^2) + 1 = 21 > 0. \end{aligned}$$

We have eliminated the endpoints $-1, 0, 1, 2$ as potential roots.

Root $\mathbf{r_1}$: On the interval $[-1, 0]$, we can conclude that

$$m \leq -3 < 1 \leq M$$

The interval $[m, M]$ must contain 0. By the intermediate value theorem, there must be some point r_1 in $[-1, 0]$ which satisfies $f(r_1) = 0$. Lastly, we have already checked the endpoints of the interval and found that $x = -1, 0$ are not roots. So the root must be in $(-1, 0)$.

Root $\mathbf{r_2}$: On the interval $[0, 1]$, we can conclude that

$$m \leq -1 < 1 \leq M$$

The interval $[m, M]$ must contain 0. By the intermediate value theorem, there must be some point r_2 in $[0, 1]$ which satisfies $f(r_2) = 0$. Lastly, we have already checked the endpoints of the interval and found that $x = 0, 1$ are not roots. So the root must be in $(0, 1)$.

Root $\mathbf{r_3}$: On the interval $[1, 2]$, we can conclude that

$$m \leq -1 < 21 \leq M$$

The interval $[m, M]$ must contain 0. By the intermediate value theorem, there must be some point r_3 in $[1, 21]$ which satisfies $f(r_3) = 0$. Lastly, we have already checked the endpoints of the interval and found that $x = 1, 2$ are not roots. So the root must be in $(1, 21)$.

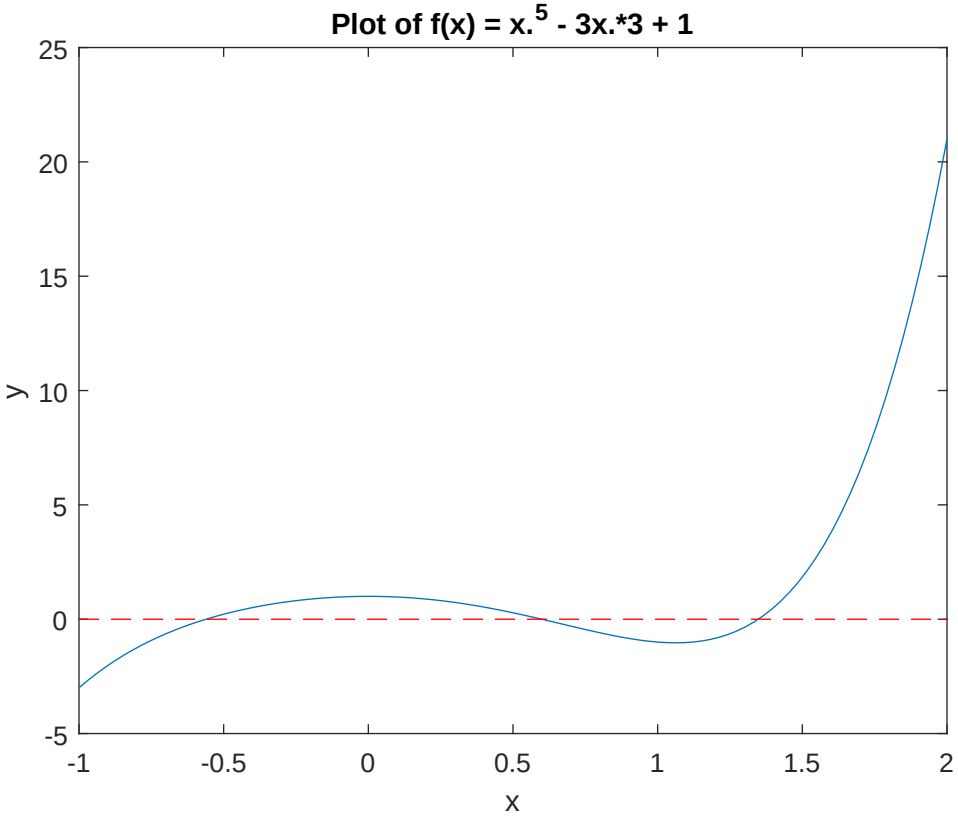
Problem 2.

Use Matlab to plot the graph of f on the interval $(-1, 2)$.

Solution

```
x = linspace(-1,2,100);  % Extra points for a smoother line
y = x.^5 - 3*x.^2 + 1;  % Compute values of y = f(x) for each x
plot(x,y)

% Visualize the x-axis
hold on
xzeros = zeros(1,100);
plot(x,xzeros, 'r--')
xlabel('x')
ylabel('y')
title('Plot of f(x) = x.^5 - 3x.*3 + 1')
hold off
```



(Note: you can insert Matlab code with the editor markdown using the *matlab-prettifier* latex package).

Problem 3.

With the help of your pocket calculator, use the bisection method to compute approximately root r_1 with the initial interval $(a_0, b_0) = (-1, 0)$.

Solution

a	b	$sign(f((a+b)/2))$
-1	0	$+1$
-1	$-\frac{1}{2}$	-1
$-\frac{3}{4}$	$-\frac{2}{4}$	-1
$-\frac{3}{4}$	$-\frac{5}{8}$	-1
$-\frac{9}{16}$	$-\frac{5}{8}$	

Our sequence of estimates is $-1/2, -3/4, -5/8, -9/16$. Our estimate after 4 iterations is $r_1 \approx -9/16 = -0.5625$.

Problem 4.

How many iterates are needed to compute r_1 using the bisection method (see prior problem) to within an error of 10^{-6}?

Solution

(See lecture notes for a more complete derivation) In lecture, we derived the relation

$$|b_n - a_n| = \frac{1}{2^n}|b_0 - a_0|$$

which describes the interval width after $n+1$ iterations. Let ϵ be the true error. As $\epsilon = |c_n - x^\star| \leq \frac{|b_n - a_n|}{2}$, we can combine the results to obtain

$$\epsilon = |c_n - x^\star| \leq \frac{||b_n| - a_n|}{2} = \frac{1}{2^{n+1}}|b_0 - a_0|$$

To find the number of iterations needed to be within an error bound of 10^{-6}, we need to find some n (preferably the smallest one) which satisfies

$$\epsilon < \frac{|b_0 - a_0|}{2^{n+1}}$$

Solve by isolating n,

$$2^{n+1} = \frac{|b_0 - a_0|}{\epsilon} = \frac{1}{\epsilon} \implies (n+1)\log_{10}(2) = (6\log_{10}(10)) \implies (n+1) = \frac{6}{log_{10}(2)} \approx 19.931$$

So $n = 21$ is the smallest n which is within the desired error bound.

Problem 5.

Write a program to compute r_1, r_2, r_3 by Newton's method. This program should contain a 'while' loop and stops when $|x_{n+1} = x_n| \leq 10^{-6}$. The initial point is of your choice.

Solution

We really only need to implement an algorithm to compute roots given a starting points, and then loop over different starting points.

```
format long %does not change machine arithmetic
maxstep = 10^(-6); %maximum step size where we can stop iteration
guess = [-0.5, 0.5, 1.5];  % Obtained by inspection of the plot in prob 2
results = zeros(1,length(guess));  % pre-allocate a vector to store our
   results
for i = 1:length(guess)
        xp = guess(i); %xp for x previous estimate
        %compute an initial estimate by Newton's method
        xn = N(xp);  % xn for x next estimate
        while abs(xp-xn)> maxstep
                xp = xn;  % store previous value
                xn = N(xp);  % compute new value
        end
        results(i) = xn;
end
disp(results)  % return values to console

%Note: these functions could be anonymized or built into the above code.
function out = N(x)  % Newton's method step
        out = x - f(x)./fp(x);
end

function out = fp(x) %fp for f prime
        out = x.*(5.*x^3 - 6);
end

function out = f(x)
        out = x.^5 - 3*x.^2 + 1;
end
```

This returns

$$-0.561070007170282 \quad 0.599241027965686 \quad 1.348046941291339$$

in the results vector which are approximately r_1, r_2, and r_3.

Problem 6.

With each initial point $x_0 = 0.16, 0.17, 0.18, 0.19$, which root does your program give?

Solution

We modify the values entered into the guess vector.

```
format long %does not change machine arithmetic
maxstep = 10^(-6); %maximum step size where we can stop iteration
guess = [0.16, 0.17, 0.18, 0.19];  % Obtained by inspection of the plot in
    prob 2
results = zeros(1,length(guess));  % pre-allocate a vector to store our
   results
for i = 1:length(guess)
        xp = guess(i); %xp for x previous estimate
        %compute an initial estimate by Newton's method
        xn = N(xp);  % xn for x next estimate
        while abs(xp-xn)> maxstep
```

```
                xp = xn;  % store previous value
                xn = N(xp);  % compute new value
        end
results(i) = xn;
end
disp(results)  % return values to console

%Note: these functions could be anonymized or built into the above code.
function out = N(x)  % Newton's method step
        out = x - f(x)./fp(x);
end

function out = fp(x) %fp for f prime
        out = x.*(5.*x^3 - 6);
end

function out = f(x)
        out = x.^5 - 3*x.^2 + 1;
end
```

This returns the values

$$1.348046941291339 \quad 1.348046941291352 \quad -0.561070007170282 \quad 0.599241027965686$$

The Geogebra visualization offers an explanation as to why this occurs. Newton's Method may not necessarily converge to the root closest to the initial guess.

Problem 7.

Let $f(x) = \sqrt[3]{x}$. We know that $x = 0$ is the only root of f. Nevertheless, we want to test if Newton's method is able to give us this root.

a) Plot the function f on the interval $[-5, 5]$.

b) Write the iterative formula of the Newton's method.

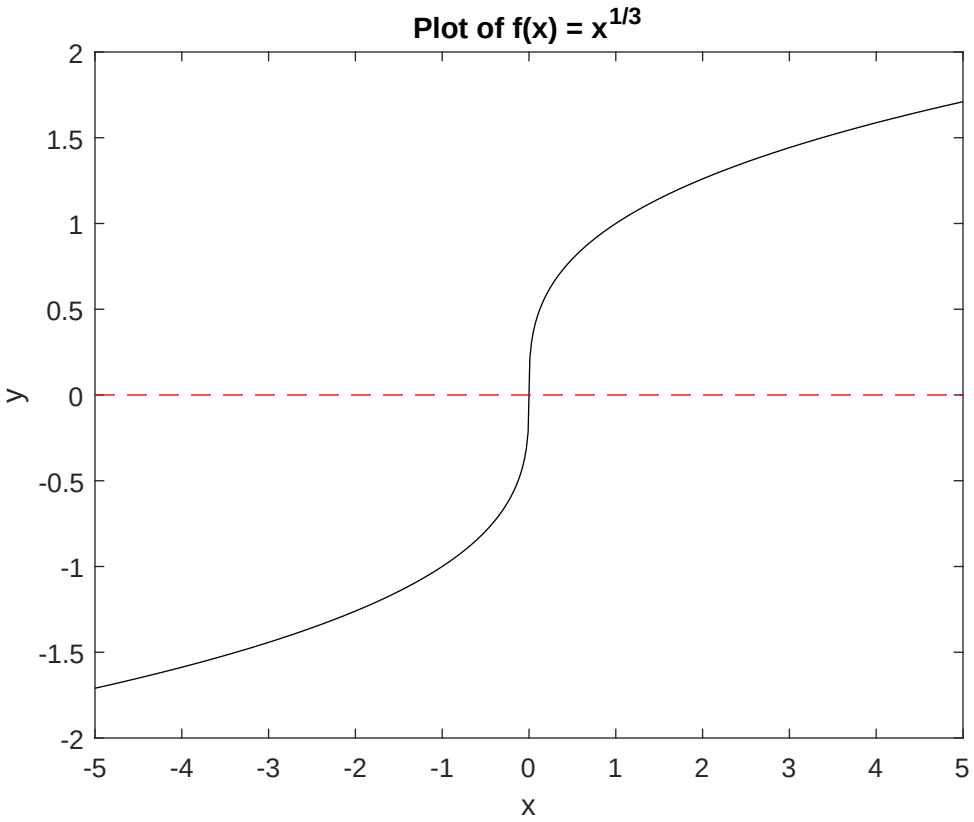
c) Express x_n as a function of n and x_0.

d) For what x_0 does x_n converge? Is newton's Method a good method to find the root of f?

Solution

a) This is doable in Matlab,

```
x = linspace(-5,5,500);
y = nthroot(x,3);
plot(x,y)
hold on
% Visualize the x-axis
xzeros = zeros(1,500);
plot(x,xzeros, 'r--')
xlabel('x')
ylabel('y')
title('Plot of f(x) = x^{1/3}')
hold off
```



and we see that f has a root at 0.

b) Newton's method defines an iterative scheme as

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

which leaves the fraction to compute,

$$\frac{f(x)}{f'(x)} = x^{1/3}\frac{1}{\frac{1}{3x^{2/3}}} = x^{1/3}(3x^{2/3}) = 3x^{3/3} = 3x$$

$$x_{n+1} = x_n - (3x_n) = -2x_n$$

is the iteration scheme defined by Newton's method.

c) This is a geometric relationship, we can use induction to observe that

$$x_n = -2x_{n-1} = -2(-2x_{n-2}) \implies x_n = (-2)^n x_0$$

We can create a function g which takes a real number x_0 (our initial guess at the root) and a whole number n (natural numbers and zero) as $x_n = g(x_0, n) = (-2)^n x_0$.

d) As a consequence of our iteration scheme, the error of the n-th iterate (call this quantity ϵ_n) is given by

$$\epsilon_n = |x_n - 0| = |x_n| = |-2x_{n-1}| = |2||x_{n-1} - 0| = 2\epsilon_{n-1} \implies \epsilon_n = 2^n \epsilon_0$$

If the initial choice x_0 is not zero, then $\epsilon_0 > 0$, so the size of the error at every successive step is monotonically increasing, so it does not converge (to anything, including the root). The only way around this problem is to set $x_0 = 0$, but that's the root we are trying to find! So the sequence of iterates converges only when $x_0 = 0$ (and nowhere else). This is not a good method for the function f.