Problem 1.

In this problem we will approximate the value of $\sqrt[3]{10}$ in a manner that can be carried out by hand (only using addition, subtraction, multiplication, and division of integers). The allowed error is $\epsilon = 10^{-3}$. You may use a pocket calculator to do the calculations and verify your results. (Matlab is not considered a pocket calculator.)

- Find a function $f(x)$ that receives $\sqrt[3]{10}$ as a root.
- Use bisection method to approximate this root of f. You will need to provide the initial interval, the number of iterations to be performed, and the approximate value of the root.
- Use Newton's method to approximate the root of f. You will need to provide the initial estimate x_0, the iteration formula, and the stopping condition (does Newton's method give a strict convergence formula for the error bound?).

Solution

One such f is

$$f(x) = x^3 - 10$$

We could also choose $f(x) = 10 - x^3$, or $f(x) = \frac{x^3}{10} - 1$ as our objective functions.

Bisection Method

As $2^3 = 8 < 10$ and $3^3 = 27 > 10$, then $f(2) < 0$ and $f(3) > 0$ so we can chose our initial interval as $[a_0, b_0] = [2, 3]$. We will need to perform

$$n \geq \log_2\left(\frac{b-a}{\epsilon}\right) - 1 = \log_2\left(\frac{1}{10^{-3}}\right) - 1 = \frac{\log(10^3)}{\log(2)} - 1 \approx 8.96578428466208\ldots$$

so $n \geq 9$ will suffice.

We can perform $n = 9$ iterations (omitted here for clarity, you need to show this) and find that the estimate should be $x_9 = 2.155273437500000 \approx \sqrt[3]{10}$.

Newton's method

We use the iteration scheme

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^3 - 10}{3x_n^2} = x_n - \frac{x_n}{3} + \frac{10}{3x_n^2} = \frac{2x_n}{3} + \frac{10}{3x_n^2}$$

We would like to approximate $\sqrt[3]{10}$ to within ϵ. To do so, we will choose a stopping condition that is better than the prescribed error tolerance to find where the change between iterations is very small. 2 additional orders of magnitude is more than sufficient, so we stop iterating our method when $|x_i - x_{i-1}| < 10^{-5}$. Our starting value (initial guess) is $x_0 = 2$. After performing 4 iterations, we see that our stopping criteria is met at $x = 2.154434690031884$. The sequence of iterates is

$$2.000000000000000, \quad 2.166666666666667, \quad 2.154503616042077, \quad 2.154434692236913, \quad 2.154434690031884$$

You can check how good this approximation is with a calculator or Matlab (it's really good).

Problem 2.

Consider a sequence defined recursively as follows.

$$x_{n+1} = \frac{15x_n^2 + 13}{4x_n} - 6, \quad x_0 = 2$$

1. Guess the limit of this sequence. Then use the recursive formula to verify that this value is a limit of x_n.

2. Find the order of convergence. If it is linear (order of convergence = 1), find the linear rate of convergence (a positive real number) C.

Consider the sequence

$$x_{n+1} = \frac{x_n}{2} + \frac{1}{x_n}, \quad x_0 = 1$$

Repeat the above 2 tasks for this sequence.

Solution

Fist Iteration

First consider

$$x_{n+1} = \frac{15x_n^2 + 13}{4x_n} - 6, \quad x_0 = 2$$

The sequence does not appear to converge, so we say it diverges. To be specific, as this sequence is monotone increasing we can more descriptively say that this sequence diverges to positive infinity.

Second Iteration

Consider

$$x_{n+1} = \frac{x_n}{2} + \frac{1}{x_n}, \quad x_0 = 1$$

We can iterate this with a pocket calculator and find 1.414213562373095 is the approximate location for the root. This happens to be a good approximation for $\sqrt{2}$ (check this on your calculator with the sqrt function). Seeing why this is true requires a bit of algebraic manipulation (factor out a $1/2x_n$ term) before we apply our limit operator.

$$x_{n+1} = \frac{x_n}{2} + \frac{1}{x_n} = \frac{1}{2x_n}\left(x_n^2 + 2\right)$$

Then we can apply the limit operator

$$\lim_{n\to\infty} x_{n+1} = \lim_{n\to\infty} \frac{1}{2x_n}\left(x_n^2 + 2\right) \implies a = \frac{1}{2a}(a^2 + 2)$$

$$\implies 2a^2 = a^2 + 2 \implies a^2 = 2 \implies a = \pm\sqrt{2}$$

And so we see that there are 2 possible equilibrium values, $a = \pm\sqrt{2}$, which includes the value we found above with our calculator.

We next consider the convergence of the sequence above towards $\sqrt{2}$. Factorization does us good here,

$$|x_{n+1} - \sqrt{2}| = \left|\frac{x_n}{2} + \frac{1}{x_n} - \sqrt{2}\right| = \left|\frac{x_n^2 + 2 - 2x_n\sqrt{2}}{2x_n}\right| = \left|\frac{x_n^2 - 2x_n\sqrt{2} + 2}{2x_n}\right| = \left|\frac{(x_n - \sqrt{2})^2}{2x_n}\right|$$

As $x_n \approx \sqrt{2}$ when n is large, we have

$$|x_{n+1} - \sqrt{2}| \approx \frac{1}{\sqrt{8}}|x_n - \sqrt{2}|^2$$

And we find the convergence is quadratic (the order of convergence is 2).

Problem 3.

Consider the function $f(x) = x^3 - 3x^2 + 1$. We will use Newton's method to approximate all roots of the function.

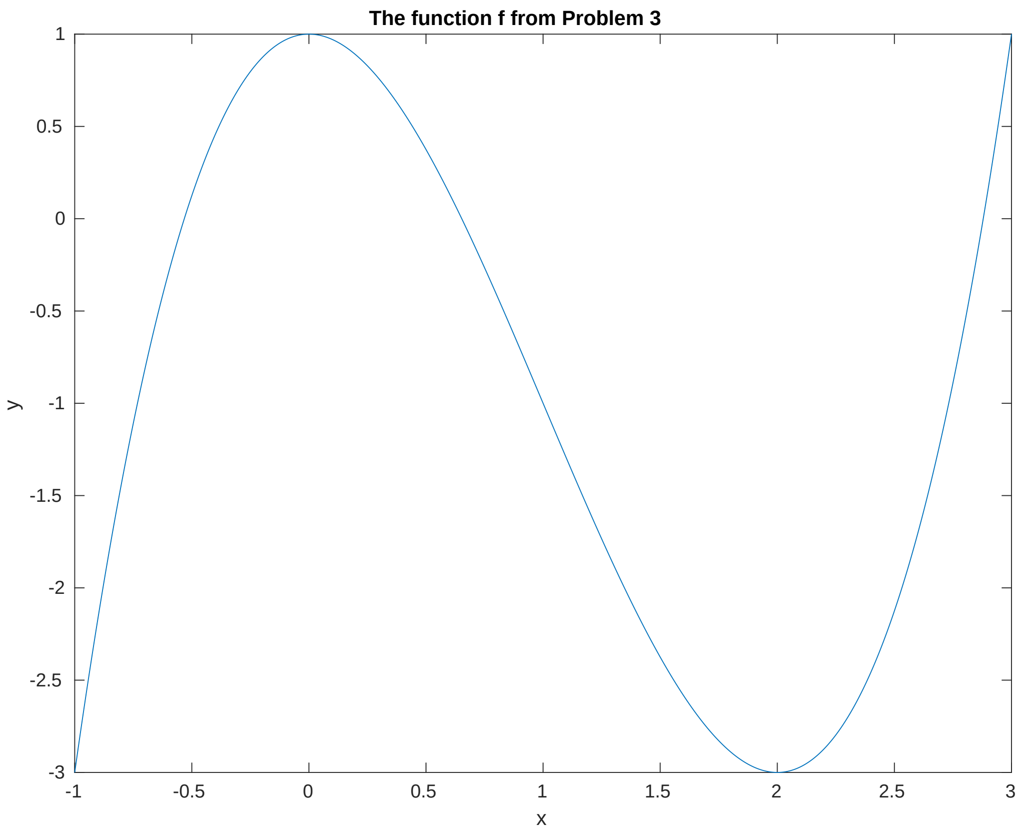
1. Use Matlab to plot this function on the interval $[-1, 3]$. How many roots does f have on this interval? Can f have any roots outside of this interval?
2. Write the iteration formula for Newton's method.
3. Label the 3 roots of f as $r_1 < r_2 < r_3$. What is the range of values x_0 that will guarantee that x_n.

Solution

```
f = @(x) x.^3 - 3.*x.^2 +1;
x_values = linspace(-1,3,800);
y_values = f(x_values);

plot(x_values, y_values)
hold on
title('The function f from Problem 3')
ylabel('y')
xlabel('x')
hold off
saveas(gcf, 'HW4P3A', 'epsc')
```

Which produces the following plot,



This polynomial appears to have 3 roots in the interval $[-1, 3]$. This polynomial is of degree 3 so it has exactly 3 (possibly complex) roots. We see 3 real roots in the interval, so there cannot be any additional roots outside of $[-1, 3]$.

To construct Newton's method iteration, we first pre-compute f' as $f'(x) = 3x^2 - 6x$, then

$$x - \frac{f(x)}{f'(x)} = x - \frac{x^3 - 3x^2 + 1}{3x^2 - 6x}$$

So the iteration formula is given by

$$x_{n+1} = x_n - \frac{x_n^3 - 3x_n^2 + 1}{3x_n^2 - 6x_n}$$

Proving that a range of values converges to a specific value is a difficult task (this is a well studied problem in dynamical systems theory known as computing an α-set of a system for each element in an ω-set) beyond the scope of the class. By numerical experimentation, we see that

- If $x_0 < 0$ then $x_n \to r_1$.
- If $0 < x_0 < 0.1$ (approximately), then $x_n \to r_3$.
- If $0.1 < x_0 < 1.6$ (approximately), then $x_n \to r_2$.
- If $1.6 < x_0 < 2$, then $x_n \to r_1$.
- If $x_0 > 2$, then $x_n \to r_3$.

Starting near an root does not necessarily imply that the sequence will converge to that root.

Problem 4.

In this problem we will use Newton's method to find approximate solutions of the system

$$\begin{cases} x^2 + y = xy \\ y^2 + x = y + 1 \end{cases}$$

1. Write an iteration formula of the Newton's method.
2. Write a matlab program that allows you to adjust the initial point $\mathbf{x_0} = (x_0, y_0)^T$ and the number of steps n. Then experiment with $(x_0, y_0)^T = (1, -1)^T$ and $n = 5$.
3. Find an approximate solution (x, y) such that $x, y < 0$.
4. Use experiments to search for positive valued solution $(x, y > 0)$. What do you observe?

Solution

We begin by manipulating the algebraic form,

$$\begin{cases} x^2 + y - xy = 0 \\ y^2 + x - y - 1 = 0 \end{cases} \implies \begin{cases} f(x, y) = x^2 + y - xy \\ g(x, y) = y^2 + x - y - 1 \end{cases}$$

which gives us a root-finding problem. Define $F((x, y)) = (f(x, y), g(x, y))^T$. Let $J(\mathbf{x})$ be the Jacobian matrix, computed as

$$J(x, y) = \begin{bmatrix} 2x - y & 1 - x \\ 1 & 2y - 1 \end{bmatrix} \implies J^{-1}(x, y) = \frac{1}{x(4y - 1) - 2y^2 + y - 1} \begin{bmatrix} 2y - 1 & x - 1 \\ -1 & 2x - y \end{bmatrix}$$

When $x(4y-1)-2y^2+y-1 \neq 0$, the Jacobian is invertible. In practice, we will check that the matrix is invertible (or use a pseudo-inverse) and then use the built-in numerical matrix inversion subroutine (which is really really fast on 2x2 matrices) to compute J^{-1}.

Then
$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} - \frac{1}{x_n(4y_n-1)-2y_n^2+y_n-1} \begin{bmatrix} 2y_n-1 & x_n-1 \\ -1 & 2x_n-y_n \end{bmatrix} \begin{bmatrix} f(x_n,y_n) \\ g(x_n,y_n) \end{bmatrix}$$
where f and g are as defined above.

A Matlab implementation is as follows: (your code must allow you to *easily* adjust (x_0, y_0) and n. Implementing this as a function rather than a script is recommended.)

```
n = 5;
initial_guess = [1,-1]';
%prediction = objective(initial_guess);

prediction = NewtonLoop(initial_guess , n);
disp(prediction)

%prediction = Multi_Newton(initial_guess);
%disp(prediction)

function image = objective(input_value)

    %For readability
    x = input_value(1);
    y = input_value(2);
    image = [x^2 + y - x*y , y^2 + x - y - 1];
end

function new_value = Multi_Newton(input_value)
    % Performs a single update process

    %For readability
    x = input_value(1);
    y = input_value(2);

    new_value =input_value -  [[2*x - y, 1-x];[1, 2*y-1]] \ objective(
        input_value)';
end


function prediction = NewtonLoop(initial_guess , n)
    % performs the update loop for n cycles
    disp(size(initial_guess))
    for ii=1:n
        new_value = Multi_Newton(initial_guess);
        initial_guess = new_value;
    end
    prediction = new_value;  % Copy the value out
end
```

This produces (after 5 iterations) $(x_5, y_5) = (0.458984212397020, -0.389390683334934)$.

To find a root with both values negative, we change the starting coordinate to $(-1,-1)$. After $n = 7$ iterations, we converge to a root at $(-1.949787524078606, -1.288794992188486)$, which satisfies our requirement.

We can check many initial values, but we will eventually conclude by exhaustion that there are no strictly positive roots of this system of nonlinear equations. Many trajectories diverge as well, rather than converging to a root. The system possibly has no positive roots that can be found by Newton's method.