CP315-Portfolio-Yiqian-Kang

Yiqian Kang

February 1 2024

4 Equation Solving

4.1 Bisection method

The Matlab code for the Bisection method is:

```
function [c, e, iter] = bisction(f, a, b, tol)
    c = f(a);
    d = f(b);
    if c * d > 0
        error('Error');
    end
    iter = 0;
    e = (b - a) / 2;
    while e > tol
        iter = iter + 1;
        c = (a + b) / 2;
        j = f(c);
        if j == 0
            break;
        end
        if c * j <= 0
            b = c;
        else
            a = c;
        end
        e = (b - a) / 2;
    end
    c = (a + b) / 2;
end
```

This is the testing code for this Bisection method:

```
format long;
f1 = @(x) x^3 + x - 1;
[c1, e1, iter1] = bisc(f1, 0, 1, 0.5e-6);
f2 = @(x) cos(x) - x;
[c2, e2, iter2] = bisc(f2, 0, 1, 0.5e-6);
```

For the function $f(x) = x^3 + x - 1$:
For 20 iteration, The numerical approximation is:

$$0.6823277476$$

For the function $f(x) = \cos(x) - x$:

For 20 iteration, The numerical approximation is:

$$0.7390847204$$

The function 1 given is

$$f(x) = x^3 - 44x^2 + 564x - 1728.$$

By evaluating this function at integer values from 4 to 25, the following values are obtained:

$$\begin{aligned} f(4) &= -112.000000 \\ f(5) &= 117.000000 \\ &\vdots \\ f(18) &= 0.000000, \\ &\vdots \\ f(21) &= -27.000000 \\ f(22) &= 32.000000 \end{aligned}$$

From these evaluations, potential roots are identified in the intervals of [4, 5], [18, 19], and [21, 22]. Applying the bisection method to these intervals yields the following approximations for the roots:

- Using the interval [4, 5], the root found is approximately 4.455996.
- Using the interval [18, 19], the function value at 18 is exactly 0, indicating a root at 18.
- Using the interval [21, 22], the root found is approximately 21.544003.

4.2 Fixed-point iteration method (FPI)

The Matlab code for the FPI is:

```
function [ans, iter] = fpi(f, x0, t, n)
    x = x0;
    iter = 0;
    for i = 1:n
        x1 = f(x);
        fprintf('x%d = %.15f\n', i, x1);
        iter = iter + 1;
        if abs(x1 - x) < t
            x = x1;
            break;
        end
        x = x1;
    end
    ans = x;
end
```

This is the testing code for the FPI:

```
format long;
g = @(x) (x + 2/x) / 2;
[sqrt2, iterations_sqrt2] = fpi(g, 1.4, 1e-15, 1000);

f = @(x) ((2*x + 2)^(1/3));
[solution, iterations_solution] = fpi(f, 1, 1e-8, 1000);
```

For the function $g(x) = \frac{x+\frac{2}{x}}{2}$:
For 4 iteration, The numerical approximation is:

$$1.414213562373095$$

For the function $f(x) = \sqrt[3]{2x+2}$:
For 13 iteration, The numerical approximation is:

$$1.769292352609585$$

4.3 Newton's method

The Matlab code for the Newton's method is:

```
function [root, iter] = newton(f, df, x0, tol)
    x = x0;
    iter = 0;
    while true
        iter = iter + 1;
        dfx = df(x);
        if dfx == 0
            error('No convergence.');
        end
        x_new = x - f(x)/dfx;
        if abs(f(x_new)) < tol
            break;
        end
        x = x_new;
    end
    root = x_new;
end
```

This is the testing code for the Newton's method:

```
format long;
f = @(x) x^3 + x - 1;
df = @(x) 3*x^2 + 1;
[root, iterations] = newton(f, df, 0.1, 1e-8)

f = @(x) x^2 + 1;
df = @(x) 2*x;
[root_i, iterations_i] = newton(f, df, 1+1i, 1e-8)
[root_neg_i, iterations_neg_i] = newton(f, df, -1-1i, 1e-8)
[root, iterations] = newton(f, df, 0.1, 1e-8);
```

For the function $f(x) = x^3 + x - 1$ with initial guess $x_0 = 0.1$ and a tolerance of 10^{-8}, Newton's method yields: For 5 iteration, The root is:

$$0.682327803862471$$

For the function $f(x) = x^2 + 1$ with initial guess $x_0 = 1 + i$ and a tolerance of 10^{-8}, Newton's method converges to: For 5 iteration, The root is:

$$-0.000000000010029 + 0.999999999991561i$$

Similarly, with initial guess $x_0 = -1 - i$, the method converges to: For 5 iteration, The root is:

$$-0.000000000010029 - 0.999999999991561i$$

Newton's method successfully found an approximation to the root of the equation $f(x) = x^3 + x - 1$. The approximated root, with a tolerance 10^{-8} The root is: 0.682327803862471 of this approximation, was achieved in 3 iterations.
We can not calculate the real root because there is no intersection between the function and the x-axis. However, by trying 100 distinct numbers, we can find that when the initial point is close to 0, the final result is random. When the initial point is far from 0, it will converge to 0.

The Newton's method was applied to the function

$$f(x) = x^3 - 44x^2 + 564x - 1728$$

with its derivative

$$f'(x) = 3x^2 - 88x + 564.$$

Given initial guesses were:

- $x_{0,1} = 1$
- $x_{0,2} = 10$
- $x_{0,3} = 20$

The tolerance for the accuracy was set to 10^{-10}. Applying Newton's method yielded the following results:

- For the initial guess $x_{0,1} = 1$, the root found was 4.4560 after 5 iterations.
- For the initial guess $x_{0,2} = 10$, the root found was 21.5440 after 10 iterations.
- For the initial guess $x_{0,3} = 20$, the root found was 21.5440 after 9 iterations.

The previously applied Newton's method with initial guesses $x_{0,1} = 1$, $x_{0,2} = 10$, and $x_{0,3} = 20$ led to the discovery of only two distinct roots. This is due to the sensitivity of Newton's method to the initial guess. The method typically converges to the nearest root, which may cause some roots to be missed if the initial guess is not sufficiently close.
To address this, a new initial guess $x_{0,4} = 17$ was chosen, closer to the expected root near 18. Applying Newton's method with this new initial guess yielded the following result:

- For the initial guess $x_{0,4} = 17$, the root found was 17.999999999999339 after 4 iterations.

4.4 Newton's Method Basins of Attraction

The function `Nm` implements Newton's method for the given polynomial equation. It takes an initial guess z_0, a tolerance level `tol`, and a maximum number of iterations `iter` as inputs.

```
function root = Nm(z0, tol, iter)
    for k = 1:iter
        f = z0^3 - 44*z0^2 + 564*z0 - 1728; % given function (1)
        df = 3*z0^2 - 88*z0 + 564;          % derivative of the function
        zn = z0 - f / df;                   % Newton's method update
        if abs(zn - z0) < tol
            root = zn;
            return;
        end
        z0 = zn;                            % update the guess
    end
    root = "Error";                         % return "Error" if convergence
end
```

The tolerance and maximum iterations are set as follows:

```
tol = 1e-6;\\iter = 100;
```

To visualize the basins of attraction, the complex plane is divided into a grid within a specified region, and Newton's method is applied to each grid point.

```
x_cor = -30:1:30;
y_cor = -30:1:30;
basin = zeros(length(y_cor), length(x_cor));
```