

## I. CMPE 360 HOMEWORK II ANSWERS

## A. Part I

In order to generate random coefficients, you can use the `rand` or `randn` functions. Note that `rand` generates uniformly distributed random numbers, whereas `randn` generates random values drawn from a normal distribution with mean zero and standard deviation. A vector of length  $N$  created as such corresponds to a random polynomial of order  $N - 1$ . Then you can use the `polyval` function to find the polynomial's value for a given  $x$ , and the `polyder` function to generate the derivative of the polynomial. These functions allow us to efficiently code Newton's method as follows:

```
polys = (rand(3,6)-0.5) * 10; % Three polynomials of order 5
% randn may generate values outside of this range
epsilon = 1e-5; % Convergence threshold = 10^-5
initial_guess = 0;

for p = 1:3
    Xo = initial_guess;
    X = Xo - polyval(polys(p,:), Xo) / polyval(polyder(polys(p,:)), Xo);
    convergence = abs(X - Xo);
    while convergence > epsilon
        Xo = X;
        X = Xo - polyval(polys(p,:), Xo) / polyval(polyder(polys(p,:)), Xo);
        convergence = abs(X - Xo);
    end
    root(p) = X;
end
```

Assuming that the initial guess is around the vicinity of a real root, this algorithm gives us that root. In order to devise a complete algorithm, we need to check for divergence as well as other roots. We can update the algorithm as follows:

```
polys = (rand(3,6)-0.5) * 10; % Three polynomials of order 5.
% randn may generate values outside of this range
epsilon = 1e-5; % Convergence threshold = 10^-5
initial_guess = 0;
for p = 1:3
    Xo = initial_guess;
    X = Xo - polyval(polys(p,:), Xo) / polyval(polyder(polys(p,:)), Xo);
    convergence = abs(X - Xo);
    while convergence > epsilon
        Xo = X;
        X = Xo - polyval(polys(p,:), Xo) / polyval(polyder(polys(p,:)), Xo);
        convergence = abs(X - Xo);
        % Code inserted here
        while convergence > 10 % Means that the algorithm diverges instead
            if initial_guess <= 0
                initial_guess = abs(initial_guess) + 1;
            else
                initial_guess = -1 * initial_guess;
            end
            Xo = initial_guess; % Start from the new guess
            X = Xo - polyval(polys(p,:), Xo) / polyval(polyder(polys(p,:)), Xo);
            convergence = abs(X - Xo);
        end
    end
    % Done
    root(p) = X;
end
```

This new section detects divergence and moves the initial guess in both directions, trying each integer value in the real line. Since a polynomial of order 5 must have at least one real root, the algorithm eventually stops. One can update the algorithm to ensure that it finds all possible roots. This requires restricting the range of initial guess to a certain range, such as  $[-100, 100]$ .

As the coefficients are already bounded, the roots are expected to be in a bounded region. This region is actually computable. For a formal proof and methodology do a search on rational root theorem on the internet.

### B. Part II

Second part of the question is essentially the same. This time instead of polynomials a more general function is used, and therefore functions related to polynomial computations are useless. Instead, the derivative should be explicitly stated and the value of the functions should be calculated using corresponding MATLAB functions. This corresponds to simply replacing the line:

```
X = Xo - polyval(polys(p,:), Xo) / polyval(polyder(polys(p,:)), Xo);
```

with

```
X = Xo - (sin(2*log(Xo)) + cos(x/5)/2) / (2*cos(2*log(x))/Xo + sin(Xo/5)/10);
```

in the code given in part I. Also plotting the function reveals that the first roots are very close. To find the first roots, the initial guesses must be chosen very carefully.

### C. Important notes

- `roots` function cannot be used for functions other than polynomials.
- `plot` function does not directly handle polynomials. `plot(polys(p,:))` plots the coefficients instead.
- In order to visually detect the roots, you need to turn the grid on using the `grid on` command after issuing the `plot` command.