Numerical methods I (2017)
PROJECT II

MACIEJ KORZENIEWSKI

1 Task description

Write a computer program to implement the Chebyshev method for finding a root of the complex polynomial

$$p(x) = \sum_{k=0}^{n} a_k x^k.$$

Use Horner's scheme.

2 Chebyshev method

Chebyshev method is a root-finding algorithm, that is it is an algorithm, for finding values x such that $f(x) = 0$, for a given continuous function f from the real numbers to real numbers or from the complex numbers to the complex numbers. Such an x is called a root or zero of the function f. As, generally, the roots may not be described exactly, they are approximated as floating point numbers, or isolated in small intervals (or disks for complex roots), an interval or disk output being equivalent to an approximate output together with an error bound.

Given an intial approximation x_0 the Chebyshev method allows us to find a root of a polynomial.

The Chebyshev method for solving the nonlinear equation $f(x) = 0$ is expressed with the following formula:

$$y_k = x_k - \frac{f(x_k)}{f'(x_k)},$$

$$x_{k+1} = y_k - \frac{f''(x_k)(y_k - x_k)^2}{2f'(x_k)}.$$

3 Horner's scheme

Horner's scheme is an polynomial evaluation technic. It has the advantage that a polynomial of degree n is evaluated with n additions and n multiplications. With this scheme, you never have to explicitly raise the varible, x, to a power.

To explain Horner's scheme, I will use the polynomial

$$p(x) = 1x^3 - 2x^2 - 4^x + 3.$$

Horner's scheme rewrites the poynomial as a set of nested linear terms:

$$p(x) = ((1x - 2)x - 4)x + 3.$$

To evaluate the polynomial, simply evaluate each linear term. The partial answer at each step of the iteration is used as the "coefficient" for the next linear evaluation.

4 Chebyshev method program

```
1  function [x, k] = Chebyshev(polynomialCoeffs, rootApprox, tolerance,
       maxIterations)
2
3  % [x, k] = Chebyshev(polynomialCoeffs, rootApprox, tolerance, maxIterations)
4  %
5  % polynomialCoeffs — Coefficient of the given polynomial
6  % rootApprox — approximation of the root to find
7  % tolerance — algorithm's tolerance
8  % maxIterations — maximum number of iterations
9  %
10 % Finding a root of polynomial w(x) = p(1)x^n + ... + p(n)x + p(n + 1)
11 % by the Chebyszev method
12
13 dx = tolerance + 1;
14 k = 0;
15
16 while abs(dx) > tolerance && k <= maxIterations
17
18     [w, dw, dww] = Horner(polynomialCoeffs, rootApprox);
19
20     if dw == 0
21         error('Cannot divide by zero!');
22     end
23
24     dx = w / dw;
25     y = rootApprox — dx;
26
```

```
    x = y - w^2*dww/(2*(dw)^3);
    k = k + 1;
    rootApprox = x;

end
end
```

5 Horner algorithm implementation

```
function [val, valp, valpp] = Horner( a, x )
%  y = a(1)*x^(n-1) + .... + a(n-1)*x + a(n)
%  Horner's scheme

    a = fliplr(a);

    n = max(size(a));
    w = a(n); p = w; r = p;

    for k = n-1:-1:3
        w = x * w + a(k);
        p = x * p + w;
        r = x * r + p;
    end

    w = x * w + a(2);
    p = x * p + w;
    w = x * w + a(1);

    val = w;
    valp = p;
    valpp = 2*r;

end
```

6 Tests

I Consider polynomial with the following form:

$$f(x) = x^3 + -2x^2 + 5x + 11.$$

We will use Horner's scheme to calculate polynomial value at, let's say $x_0 = 1$. Moreover, we will calculate value of $f'(x_0)$ and $f''(x_0)$.

Hence,

$$f(x_0) = 15, f'(x_0) = 4, f''(x_0) = 2.$$

Those values will be substituted inside the Chebysev algorithm, inside the following formula:

$$y_k = x_k - \frac{f(x_k)}{f'(x_k)},$$

$$x_{k+1} = y_k - \frac{f''(x_k)(y_k - x_k)^2}{2f'(x_k)}.$$

The output of the Chebyshev program is as follows:

$$x = -1.2275,$$

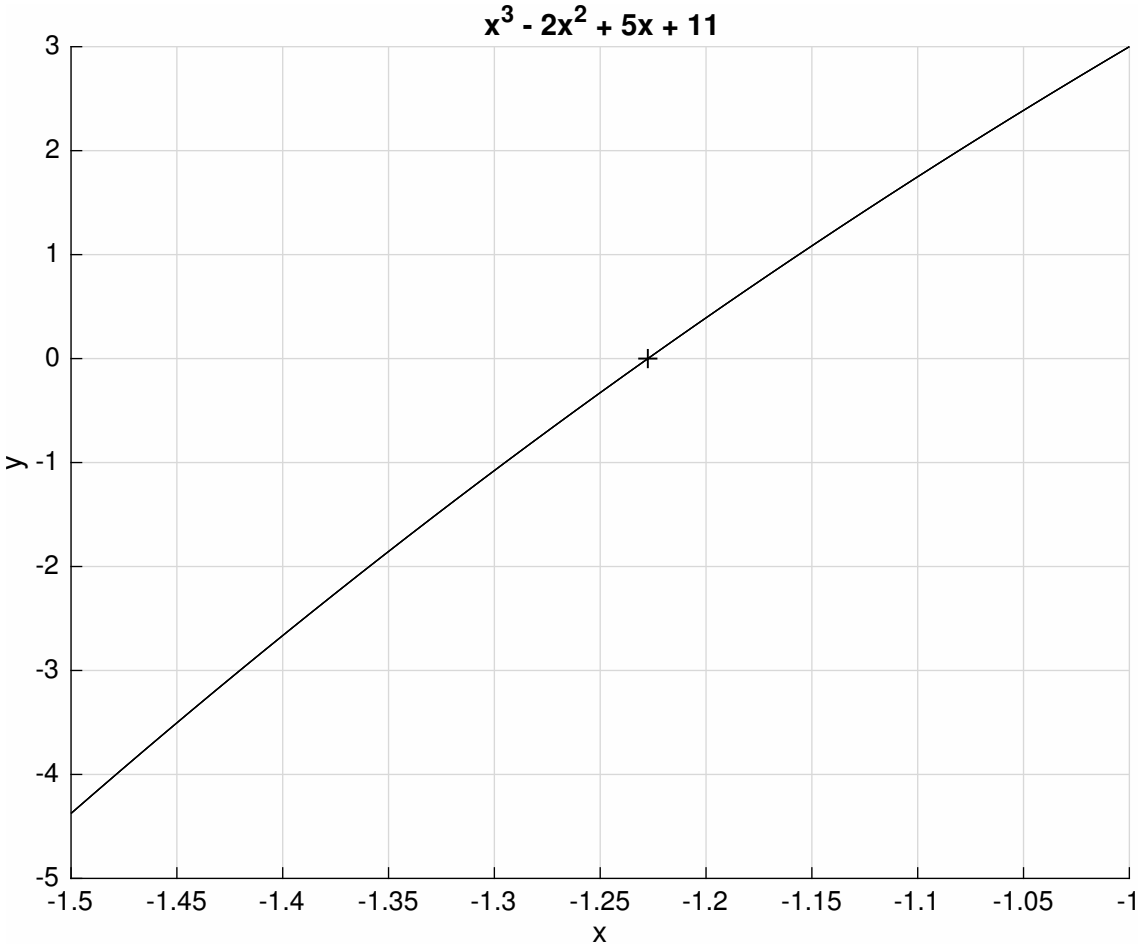
$$k = 6,$$

where k indicates number of iterations.

Now, we will compare our result with Matlab's build in function ***roots***, which is resposible for finding roots of the polynomial. Comparison resulted in an error equal to:

$$error = 2.2204 * 10^{(-16)},$$

which indicates very good result.


x^3 - 2x^2 + 5x + 11
3
2
1
0
-1
-2
-3
-4
-5
y
-1.5
-1.45
-1.4
-1.35
-1.3
-1.25
-1.2
-1.15
-1.1
-1.05
-1
x