26 May 2019

Patryk Ẇalczak
Group 2

Project 10

Finding a root of the complex polynomial p(x) using the King method and Horner scheme, where

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

1 Method description

The King method is iterative method of finding root of the funciotn $f(x)$, more precisely in the task finding root of polynomial $p(x)$. Constructing a sequence x_k such that $x_k \approx \alpha$, where $f(\alpha) = 0$, using formulas:

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

$$x_{k+1} = y_k - \frac{f(y_k)}{f'(y_{k)}} - \frac{f(x_k)}{f'(x_k)}\left(\frac{f(y_k)}{f(x_k)}\right)^3$$

Starting from a given initial guess x_0 we construct a sequence x_k of approximations, such that $x_k \to \alpha$ as $k \to \infty$.

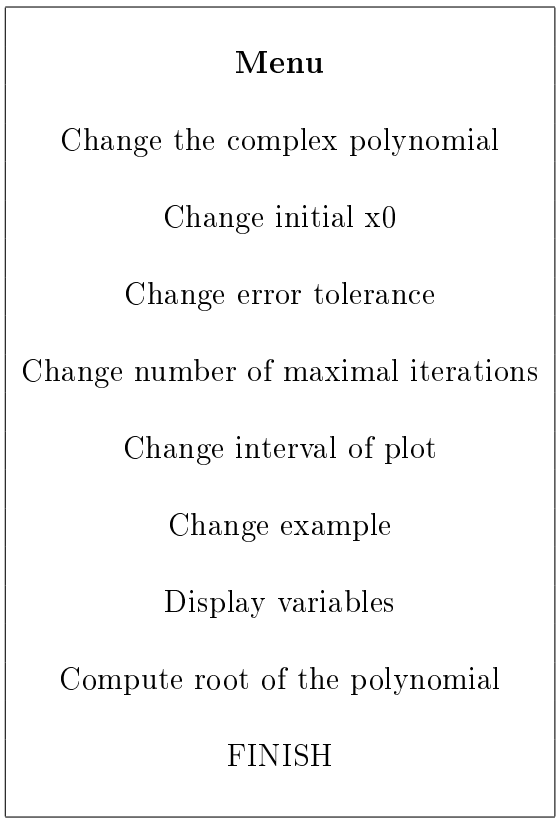
Horner sheme is method of evalution polynomial value for received data. This sheme allows evaluation of a polynomial of degree n with only n multiplications and n additions. This is optimal, since there are polynomials of degree n that cannot be evaluated with fewer arithmetic operations. Horner's method formula for polynimials looks like

$$p(x) = a_0+a_1x+a_2x^2+a_3x^3+\cdots+a_nx^n = a_0+x\Big(a_1+x\Big(a_2+x\big(a_3+\cdots+x(a_{n-1}+x\,a_n)\cdots\big)\Big)\Big).$$

In the project, Horner scheme is used for every evaluation of polynomial value and value of polynomial derivative in the King method.

2 Description of Matlab program

After run the program, the Menu appears:



After pushing:

- Change the complex polynomial - user will be able to input their own roots of polynomial. The program compute coefficients of polynomial using matlab function 'poly([set of user's roots])', that is the easer way to input polynomial, but that is also possibility to input coefficients of polynomial.
- Change initial x0 - user will be able to input their own initial value of x_0.
- Change error tolerance - user will be able to input their own tolerance of error.
- Change number of maximal iterations - user will be able to input their own limit of iterations.
- Change example - change the default variables.
- Display variables - user will be able to see all variables.

- Compute root of the polynomial - program computes root of polynomial using the King Method and Horner scheme, then prints score on screen and value of the polynomial of the root to be sure that score is correct.
- FINISH - program will finish.

MATLAB functions:

1. Menu.m - script for graphic interface of menu, which uses rest of functions.
2. King.m - function strictly for computing root of polynomial with the King method, using initial x_0 iterate to obtain root (with $\pm$ error), or obtain maximal amount of iterations, then return root and number of iteration
3. Horner.m - fuction evaluates value of polynomial for input polynomial and value of x .

   (**Note.** *All source codes can be found in section 5.*)

3 Numerical tests

All the numerical tests are included in the Menu.m file, while user pushes button 'Change example', the example data are changed. Program computes the value of root (if it obtains before number of maximal iteration), and return amount of iterations. Then plot of real value of the polynomial appears. In every test the variables are different, for checking the correctnes of program there is error also.

Tests:
Polynomial is obtained by input roots and evaluate coefficents by matlab function "poly([set of roots])". In every example is choosed initial value of x_0 from that program will start compute root.

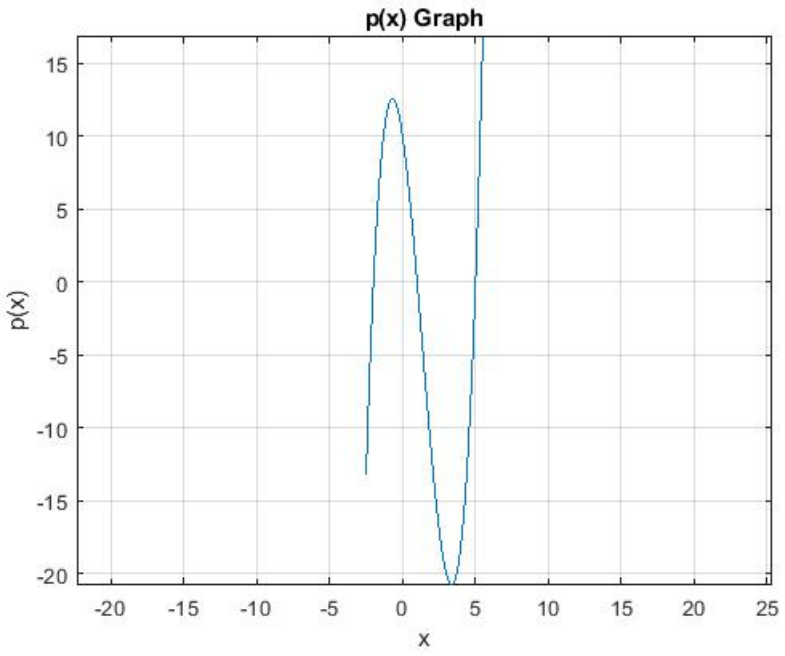
1. Initial roots are $a = [1, -2, 5]$ from the set obtained polynomial is

   $$p(x) = 1x^3 - 4x^2 - 7x + 10,$$

   where initial value of x_0 is equal 3. The tolerance error is equal $1e^{-10}$
   and number of maximal iteration is equal 100.
   Moreover interval of plot is $[-2.5, 5.5]$ to see all roots.

   Program returns:

   King method after 1 iterations returns value root = -2
   and value of the polynomial for root = 0



From graph is known that program computes root correctly.

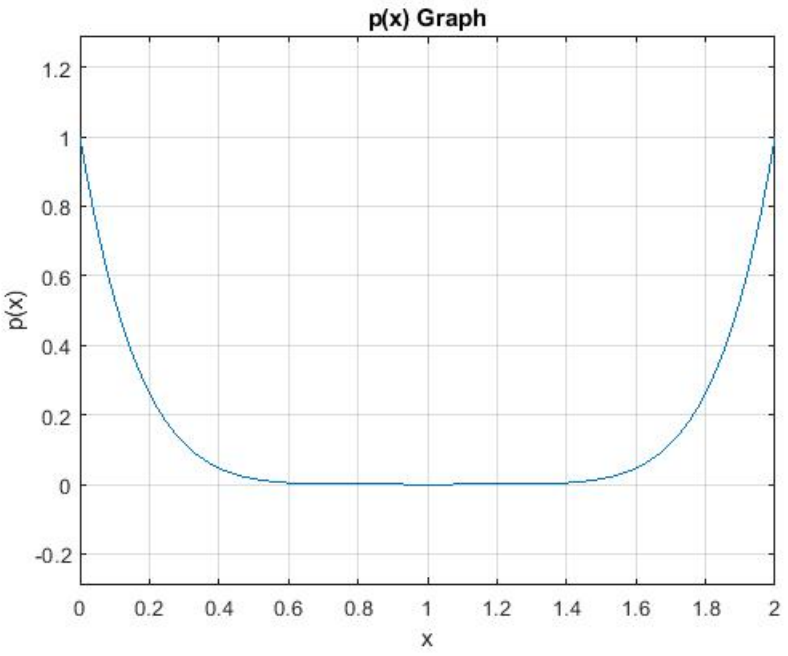
2. Initial roots are $a = ones(1, 6)$ from the set obtained polynomial is

$$p(x) = 1x^6 - 6x^5 + 15x^4 - 20x^3 + 15x^2 - 6x + 1,$$

where initial value of x_0 is equal 3. The tolerance error is equal $1e^{-10}$ and number of maximal iteration is equal 100.
Moreover interval of plot is $[0, 2]$ to see all roots.

Program returns:

King method after 13 iterations returns value root = 1.015531e+00
and value of the polynomial for root = 1.403233e-11



From graph is known that program compute root correctly. Value of the polynomial is close to 0 ($1.403233e^{-11} = 0.00000000001403233$) and thats is less then error.

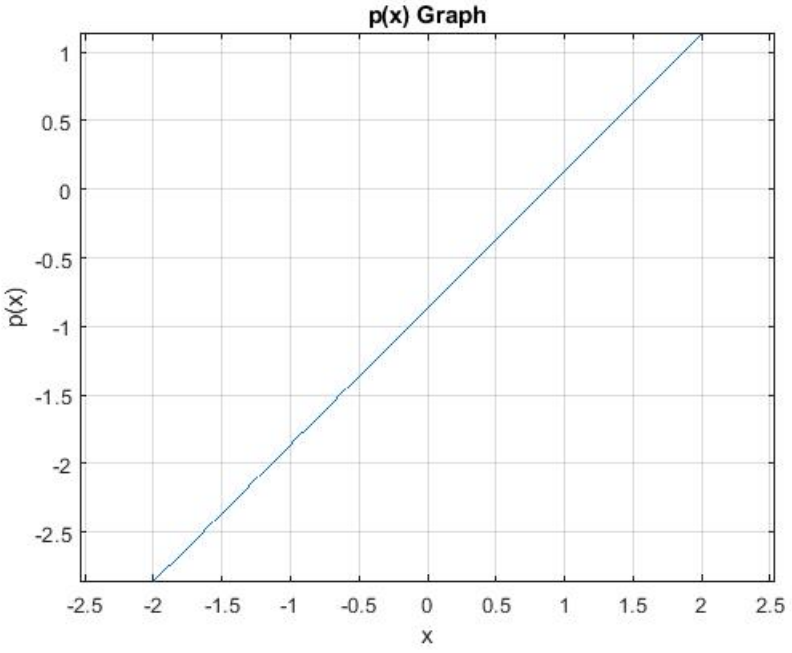
3. In the example, roots are random, and they are complex numbers to check correctners of the program for comples numbers. Initial roots are $a = randn(1)+2i$ from the set obtained polynomial is

$$p(x) = p(x) = (1.0000 + 0.0000i)x - 0.8622 - 2.0000i$$

where initial value of x_0 is equal -1. The tolerance error is equal $1e^{-10}$
and number of maximal iteration is equal 100.
Moreover interval of plot is $[-2, 2]$ to see all roots.

Program returns:

King method after 1 iterations returns value root = 8.621733e-01
and value of the polynomial for root = 0



From graph is known that program computes root correctly even for complex numbers. But because graph is for real numbers there is no imaginaru part of polynomial value.

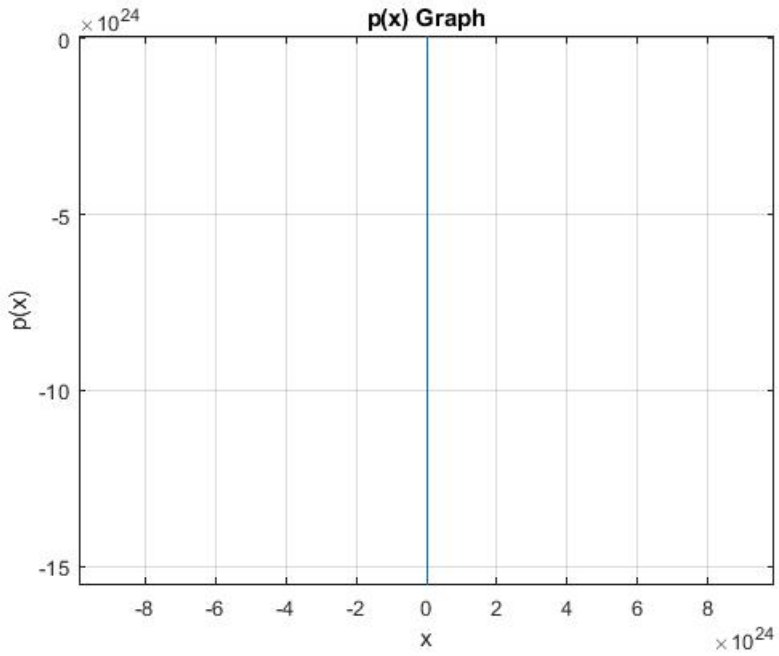
4. This is the example with the error, to be sure that program can not compute root for bad choosed number. Initial roots are $a = [1 : 25]$ from the set obtained polynomial is

$$p(x) = -0.0003x^{10} + 0.0020x^9 - 0.0100x^8 + 0.0414x^7 - 0.1375x^6 + 0.3577x^5 - 0.7087x^4 + 1.0234x^3 - 1.0048x^2 + 0.5919x - 0.1551$$

where initial value of x_0 is equal 1000. The tolerance error is equal $1e^{-10}$ and number of maximal iteration is equal 100.
Moreover interval of plot is $[0, 25]$ to see all roots.

Program return:

King method after 101 iterations return value root = -1.339470e+07 and value of the polynomial for root = -1.490524e+178



Here program obtains 100 iterations and returns no correct root. For too big initial value of x_0 100 iteration is too small amount of iterations.

4 Conclusion

First what is mentioned in the raport Horner scheme is efficient way of computing value of polynomial. Secondly, the succes of the finding root of polynomial using the King method is dependent of the initial value of x_0. Both algorithms works for complex numbers what is proved above. Moreover the King method coincides fast to the value of the root. Although it computes value of root with really low amount of iterations,that always finds only one root, what is usefull if we search any root, but that is no such usefull when we need all roots.

5 Source Codes

Is_Hessenberg.m :

```
function [x,k] = King(p,x0,tol,max_iter)
% [x,k] = King(p,x0,tol,max_iter)
% Finding a root of polynomial  w(x)=p(1) x^n+...+ p(n) x + p(n+1)
% by the King method

dx=tol+1;
k=0; x=0;
```

```
dp=polyder(p); % dp derivative of the polynomial p
while abs(dx)> tol && k<= max_iter
    w=Horner(p,x0); %value of p(x0)
    if abs(w)<=tol
        x=x0; return;
    end

    dw=Horner(dp,x0);%value of p'(x0)
    if dw==0
        disp(' You divide by zero! ');
        return;
    end
    dx=w/dw;
    y=x0-dx;
    pyk=Horner(p,y);%p(yk)
    pxk=Horner(p,x0);%p(xk)
    dpxk=Horner(dp,x0);%p'(x0)
    dpyk=Horner(dp,y);%p'(yk)
    if dpyk==0 || dpxk==0 || pxk==0
        disp(' You divide by zero! ');
        return;
    end
    c=(pyk/pxk)^3; %(p(yk)/p(xk))^3
    b=pxk/dpxk; % p(xk)/p'(xk)
    a=pyk/dpyk;   % p(yk)/p'(yk)
    x=y-a-(b*c); %next value of x
    k=k+1;
    x0=x;
end
end
```

Jacobi_Method.m :

```
function y = Horner(a,x)
% y = a(1)*x^(n-1) + .... + a(n-1)*x + a(n)
% Horner's scheme compute the value the polynomial with value of x

    n = max(size(a));
    y = 0;
    for k = 1 : n
            y = a(k) + x .* y;
    end
end
```

Menu.m:

```
% MENU
clear
clc
finish=9;
control=1;

%default data %
a=[1,-2,5];
p=poly(a);
tol=1e-10;
max_iter=100;
n=-2.5;
m=5.5;
interval=linspace(n,m);
x0=3;
%example counter
ex=0;

while control~=finish

    control=menu('Menu', 'Change the complex polynomial',
    'Change initial x0','Change error tolerance',
    'Change number of maximal iterations','Change interval of plot',
    'Change example','Display variables','Compute root of the polynomial',
    'FINISH');

    switch control
        case 1
            a=input('p(x)= ');
            p=poly(a);

        case 2
            x0=input('x0= ');
        case 3
            tol=input('tolerance= ');

        case 4
            max_iter=input('max_iter= ');

        case 5
            n=input('start= ');
            m=input('end= ');
            interval=linspace(n,m);
```

```
case 6
    if ex==0
        a=ones(1,6);
        p=poly(a);
        tol=1e-10;
        max_iter=100;
        n=0;
        m=2;
        interval=linspace(n,m);
        x0=3;
    end
    if ex==1
        a=randn(1)+2i;
        p=poly(a);
        tol=1e-10;
        max_iter=100;
        n=-2;
        m=2;
        interval=linspace(n,m);
        x0=-1;
    end
    if ex==2
        a=[1:25];
        p=poly(a);
        tol=1e-10;
        max_iter=100;
        n=0;
        m=25;
        interval=linspace(n,m);
        x0=1000;

    end
    if ex==3
        a=[1,-2,5];
        p=poly(a);
        tol=1e-10;
        max_iter=100;
        n=-2.5;
        m=5.5;
        interval=linspace(n,m);
        x0=3;

    end
    ex=ex+1;
    ex=mod(ex,4);
```

```
        case 7
            disp('p(x)= '); disp(p)
            disp('x0= '); disp(x0)
            disp('max_iter= '); disp(max_iter)
            disp('tolerance= '); disp(tol)
            fprintf('interval of plot = [%d,%d]\n',n,m);

        case 8
            [root,k]= King(p,x0,tol,max_iter);
            fprintf('King method after %d iterations returns
            value root = %d\n',k,root);
            fprintf('and value of the polynomial for root = %d\n',
            Horner(p,root));
            f=@(s) Horner( p , s );
            plot(interval, f(interval)), xlabel('x'), ylabel('p(x)'),
            title('p(x) Graph'),
            grid on, axis equal

        case 9
            disp('FINISH')
    end
end
```