Q:
给定一个六次多项式，系数均为实数时。
（1）假设所有根均为实数，试编写一套求根算法找到所有的6个实根。说明你的算法为什么是有效的。
（2）假设有一个复数根在一个已知的实数附近（例如，复根在实数3附近），试编写算法实现寻找该复根的方法。
（3）请查找相关文献，尽可能实现寻找所有6个复根的算法，并谈谈算法是否可以延拓到任意整数情形。

A:
(1)
idea:
The core idea of the algorithm is to remove the known roots of the polynomial.
It is easy to find one or a few roots of a polynomial of degree 6 using Newton's method (depending on the starting guess chosen), but it is not practical to find all six roots because we cannot predict the roots and therefore cannot choose all the starting guesses properly. It means we must find all six roots with only one starting guess, which requires that the algorithm cannot iterate to the same root. So, once we find a root, we get rid of it.
In particular, the method we're using is polynomial division. The original polynomial divided by , where is the root that we have already found, so that we won't find that root again in the next operation.

code:
main.m:

```
p = poly([1,2,3,4,5,6]);        % a sixth degree polynomial with these root
n=6;                            % Sixth degree
x = zeros(1,n);                 % the array that stores the roots
for i = n:-1:1                  % get rid of one root per cycle
    dp = zeros(1,i);            % dp: Derivative of the polynomial
    for j=1:i                   % calculate the derivative
        dp(j) = (i+1-j)*p(j);
    end
    x(n+1-i) = Newton(p,dp,0);  % calculate a new root
    p = newpoly(p,i,x(n+1-i));  % generate the polynomial with the new root removed
end
x                               % show the roots
```

Newton.m:

```
function x=Newton(f,df,a)
b=a-polyval(f,a)/polyval(df,a);
while abs(a-b)>10^(-9)              % answer is accurate enough
    a=b;
    b=b-polyval(f,b)/polyval(df,b);
end
x=b;
```

newpoly.m:

```
function newp=newpoly(p,n,x0)
newp = zeros(1,n);
newp(1) = p(1);
for i = 2:n
    newp(i) = p(i)+x0*newp(i-1);    % polynomial division formulas
end
```

results:

```
x =

列 1 至 3

0.999999999999998   2.000000000000015   3.000000000000002

列 4 至 6

3.999999999999837   5.000000000000293   5.999999999999854
```

discussion:

This method will produce errors. For example, for polynomials , both of its roots are 1. Suppose that the first root we find using Newton's method is 0.9, the polynomial with this root removed will become (x-1.1). However, . By contrast, there is an error 0.01.

(2)

Newton's method can find real and complex roots. In order to find a complex root near 3, we choose the starting guess with a imaginary part.

code:

```
p = poly([1,2,4,5,3.2+0.5i,3.2-0.5i]);  % a sixth degree polynomial with these root
n=6;                                    % Sixth degree
x = zeros(1,n);                         % the array that stores the roots
for i = n:-1:1                          % get rid of one root per cycle
    dp = zeros(1,i);                    % dp: Derivative of the polynomial
    for j=1:i                           % calculate the derivative
        dp(j) = (i+1-j)*p(j);
    end
    x(n+1-i) = Newton(p,dp,3+1i);       % calculate a new root
    p = newpoly(p,i,x(n+1-i));          % generate the polynomial with the new root removed
end
x                                       % show the roots
```

results:

```
x =

列 1

3.199999999999600 + 0.500000000000208i

列 2

3.199999999999725 - 0.500000000000232i

列 3

1.999999999999975 - 0.000000000000025i

列 4

4.000000000001094 + 0.000000000000047i

列 5

4.999999999999599 - 0.000000000000000i

列 6

1.000000000000005 + 0.000000000000002i
```

(3)

The algorithm successfully find all the complex roots.

code:

```
p = poly([1+2i,1-2i,4+5i,5+1i,3.2+0.5i,3.2-0.5i]);  % a sixth degree polynomial with these root
n=6;                                        % Sixth degree
x = zeros(1,n);                             % the array that stores the roots
for i = n:-1:1                              % get rid of one root per cycle
    dp = zeros(1,i);                        % dp: Derivative of the polynomial
    for j=1:i                               % calculate the derivative
        dp(j) = (i+1-j)*p(j);
    end
    x(n+1-i) = Newton(p,dp,3+1i);           % calculate a new root
    p = newpoly(p,i,x(n+1-i));              % generate the polynomial with the new root removed
end
x                                           % show the roots
```

results:

```
x =

列 1

3.199999999999989 + 0.499999999999973i

列 2

3.200000000000009 - 0.499999999999985i

列 3

3.999999999999996 + 5.000000000000006i

列 4

1.000000000000001 - 2.000000000000000i

列 5

5.000000000000004 + 1.000000000000005i

列 6

1.000000000000000 + 2.000000000000001i
```