**Numerical Analysis: Roots of a polynomial equation**

(Bairstow + Polynomial Deflation + Root Formula)

1. Basic Theoretical Background

* 1. Polynomial Deflation
* 2nd order polynomial equation with two real roots  

 🡪 

* 2nd order polynomial equation with a complex pair  

 🡪 

* polynomial deflation



 

* 1. Nonlinear residual function for the Bairstow method



* 1. Newton Raphson method to resolve 



* 1. Roots of the 2nd order polynomial 

If  and , one solution: 

Else

 where 

1. Program Structure

**Main** **polynomial\_roots\_Bairstow**

iter\_root\_max = n/2+1;

if n < 3 ; **function root\_formula** (a,epsilon\_eisilon); return

for iter1 = 1:1:iter\_root\_max

**function** **polynomial\_root2\_Bairstow**

**function polynomial\_deflation2**(n,a,q1,q0)

aa(1,1) = q0

aa(2,1) = q1

aa(3,1) = 1.0

**function** **root\_formula**(aa,epsilon\_eisilon);

n = n – 2

if n < 3

if n == 1; a(3,1) = 0.0; end

**function root\_formula**(a,epsilon\_eisilon);

return

end if

end

1. Function Description
2. **polynomial\_roots\_Bairstow**

%------------------------------------------------------------------------------------------

% Finding all roots of an n-th order polynomial using Bairstow's method (Polynomial Deflation)

% y = a(1,1)+a(2,1)\*x + a(3,1)\*x^2 + .... + a(n+1,1)\*x^n

%------------------------------------------------------------------------------------------

% Input

% n : the order of polynomial

% a(n+1,1) : polynomial coefficients

% root1\_real0 : real part of the initial guess of one root

% root1\_imag0 : imaginary part of the initial guess of one root

% root2\_real0 : real part of the initial guess of the other root

% root2\_imag0 : imaginary part of the initial guess of the other root

% epsilon : tolerance

% alpha : under relaxation factor ((0,1])

% iter\_max : allowd maximum number of iteration

%------------------------------------------------------------------------------------------

% Output

% nroot : number of roots found

% root\_real(nroot,1) : real parts of roots

% root\_imag(nroot,1) : imaginary parts of roots

% nresid : number of the 1st order residual function

% residual(nresid,2) : coefficients of the 1st order residual function

% iter\_root(nresid,1): iteration number for each polynomial-deflation routine

%-------------------------------------------------------------------------------------------

function [nroot,root\_real,root\_imag,nresid,residual,iter\_root]=polynomial\_roots\_Bairstow (n,a,root1\_real0,root1\_imag0,root2\_real0,root2\_imag0,epsilon,alpha,iter\_max)

%-----------------------------------------------------------------------------------------------------------------

% (1) intialize parameter

%-----------------------------------------------------------------------------------------------------------------

epsilon\_eisilon = 0.1^16;

nroot = 0 ; nresid = 0; residual(1,1:2) = 0.0; iter\_root(1,1) = 0;

%-----------------------------------------------------------------------------------------------------------------

% (2) define maximum number of root search using function (polynomial\_root2\_Bairstow)

%-----------------------------------------------------------------------------------------------------------------

iter\_root\_max = n/2 + 1 ; % maximum number of root search using (polynomial\_root2\_Bairstow)

%-----------------------------------------------------------------------------------------------------------------

% (3) Check the polynomial order

%-----------------------------------------------------------------------------------------------------------------

if n < 3

[nroot,root\_real,root\_imag] = root\_formula(a,epsilon\_eisilon);

return

end

%-----------------------------------------------------------------------------------------------------------------

% (4) iterative root-finding

%-----------------------------------------------------------------------------------------------------------------

for iter1 = 1:1:iter\_root\_max

[b,r,r\_real,r\_imag,iter]=polynomial\_root2\_Bairstow(n,a,root1\_real0,root1\_imag0,root2\_real0,root2\_imag0,epsilon,alpha,iter\_max)

n1 = nroot + 1;

n2 = nroot + 2;

root\_real(n1:n2,1) = r\_real(1:2,1);

root\_imag(n1:n2,1) = r\_imag(1:2,1);

%

nroot = nroot + 2 ;

nresid = nresid+ 1 ;

residual(nresid,1:2) = r(1:2,1) ;

iter\_root(nresid,1) = iter ;

%-----------------------------------------------------------------------------------------------------------------

% (5) Preparation of next iteration

%-----------------------------------------------------------------------------------------------------------------

n = n - 2 ; a = 0.0 ; a(1:n+1,1) = b(1:n+1,1); b = 0.0 ;

%

root1\_real0= r\_real(1,1); root1\_imag0= r\_imag(1,1);

root2\_real0= r\_real(2,1); root2\_imag0= r\_imag(2,1);

% Check the polynomial order

if n < 3

if n == 1

a(3,1) = 0.0;

end

%

[nroot1,r\_real,r\_imag] = root\_formula(a,epsilon\_eisilon);

n1 = nroot + 1 ;

n2 = nroot + nroot1 ;

root\_real(n1:n2,1) = r\_real(1:nroot1,1); root\_imag(n1:n2,1) = r\_imag(1:nroot1,1);

%

nroot = nroot + nroot1 ;

nresid = nresid + 1 ;

residual(nresid,1:2) = 0.0 ;

iter\_root(nresid,1) = 0 ;

%

return

end

end

1. **root\_formula**

%------------------------------------------------------------------------------------------

% Finding roots for the 1st order and the 2nd order polynomial equations

% y = a(1,1)+a(2,1)\*x + a(3,1)\*x^2

%------------------------------------------------------------------------------------------

% Input

% a(3,1) : polynomial coefficients

% epsilon\_eisilon : very very small number (0.1^16)

%------------------------------------------------------------------------------------------

% Output

% m : number of roots

% root\_real : real parts of roots

% root\_imag : imaginary parts of roots

%-------------------------------------------------------------------------

function [m,root\_real,root\_imag] = **root\_formula**(a,epsilon\_eisilon)

%-------------------------------------------------------------------------

1. **polynomial\_root2\_Bairstow**

%------------------------------------------------------------------------------------------

% Finding two roots of an n-th order polynomial using Bairstow's method (Polynomial Deflation)

% y = a(1,1)+a(2,1)\*x + a(3,1)\*x^2 + .... + a(n+1,1)\*x^n

% = (x^2+q1\*x+q0)\*[b(1,1)+b(2,1)\*x + b(3,1)\*x^2 +....+ b(n-1,1)\*x^(n-2)]

% roots = 0.5\*[-q1+-sqrt(q1^2-4\*q0)]

%------------------------------------------------------------------------------------------

% Input

% n : the order of polynomial

% a(n+1,1) : polynomial coefficients

% root1\_real0: real part of the initial guess of one root

% root1\_imag0: imaginary part of the initial guess of one root

% root2\_real0: real part of the initial guess of the other root

% root2\_imag0: imaginary part of the initial guess of the other root

% epsilon : tolerance

% alpha : under relaxation factor ((0,1])

% iter\_max : allowd maximum number of iteration

%------------------------------------------------------------------------------------------

% Output

% b : coeficients of the remained quotient polynomial (order = n-2)

% r : coefficient of the residual polynomial (order= 1:r1\*x + r0)

% root : two roots in complex form

%-------------------------------------------------------------------------

function [b,r,root\_real,root\_imag,iter]= **polynomial\_root2\_Bairstow** (n,a,root1\_real0,root1\_imag0,root2\_real0,root2\_imag0,epsilon,alpha,iter\_max)

1. **polynomial\_deflation2**

%-------------------------------------------------------------------------

% Polynomial Deflation for the n-th order polynomial

% y = a(1,1)+a(2,1)\*x + a(3,1)\*x^2 + .... + a(n+1,1)\*x^n

% = (x^2+q1\*x+q0)\*[b(1,1)+b(2,1)\*x + b(3,1)\*x^2 +....+ b(n-1,1)\*x^(n-2)]

%-------------------------------------------------------------------------

function [b,r]=**polynomial\_deflation2**(n,a,q1,q0)

% =======================================================================

% Example Test #1

% (a) MATLAB

% >> n=8; p=[1 2 3 4 5 6 7 8 9]; rmat = roots(p)

% (b) BAIRSTOW

% >> n = 8; a=[9;8;7;6;5;4;3;2;1];

% >> root1\_real0 = 0.0; root1\_imag0 = 0.0;

% >> root2\_real0 = 0.0; root2\_imag0 = 0.0;

% >> epsilon = 0.1^8;

% >> alpha = 1.0; iter\_max = 1000;

%

%>> [nroot,root\_real,root\_imag,nresid,residual,iter\_root]

% = polynomial\_roots\_Bairstow(n,a,root1\_real0,root1\_imag0,

% root2\_real0,root2\_imag0,epsilon,alpha,iter\_max)

% =======================================================================

%>> rmat = roots(p)

%>> nroot

%>> nresid

%>> root1 = root\_real+i\*root\_imag

%>> residual

%>> iter\_root

%>> figure(1),p1=plot(rmat,'ob');grid;hold on

%>> p2=plot(root1,'sr');xlabel('real part');ylabel('imaginary part');

%>> legend([p1,p2],'MATLAB','BAIRSTOW')

% =======================================================================



% =======================================================================

% Example Test #2

% (a) MATLAB

% >> n=9; p=[1 2 3 4 5 6 7 8 9 10]; rmat = roots(p)

% (b) BAIRSTOW

% >> n = 9; a=[10;9;8;7;6;5;4;3;2;1];

% >> root1\_real0 = 0.0; root1\_imag0 = 0.0;

% >> root2\_real0 = 0.0; root2\_imag0 = 0.0;

% >> epsilon = 0.1^8;

% >> alpha = 1.0; iter\_max = 1000;

%

%>> [nroot,root\_real,root\_imag,nresid,residual,iter\_root]

% = polynomial\_roots\_Bairstow(n,a,root1\_real0,root1\_imag0,

% root2\_real0,root2\_imag0,epsilon,alpha,iter\_max)

% =======================================================================

%>> rmat = roots(p)

%>> nroot

%>> nresid

%>> root1 = root\_real+i\*root\_imag

%>> residual

%>> iter\_root

%>> figure(1),p1=plot(rmat,'ob');grid;hold on

%>> p2=plot(root1,'sr');xlabel('real part');ylabel('imaginary part');

%>> legend([p1,p2],'MATLAB','BAIRSTOW')

% =======================================================================



**Matlab commands**

**Commands: function, fprintf, fopen, disp, plot, return, break**

**Logical : == (equal), ~= (not equal), <= (less or equal), > (greater than)**