**Lecture Note-Numerical Analysis (5): Roots of the Polynomial Equation**

1. **Definition of n-th order polynomials and their computation**

* **N-th order polynomials**



* **Computation of n-th order polynomial**

1. 

Number of multiplications = 1+3+4+5+….. +(n+1) = (n+1)(n+2)/2 = O(n2)

Number of additions = n

1. 

Number of multiplications = 1+1+1+1+….. +1 = n = O(n)

Number of additions = n

**Function PolyVal(n, a, x, p)**

**! Pseudo code for calculating n-th order polynomial**

**! input: n(order), a(1:n+1)(coefficients), x(independent variables)**

**! outout: p(polynomial value)**

**!-----------------------------------------------------------------------------------------------------------------------**

**p = a(n)**

**do j=n, 1, -1**

**p = a(j-1) + x\*p**

**end do**

**!-----------------------------------------------------------------------------------------------------------------------**

**End PolyVal**

1. **Polynomial Deflation: Removal of roots from a polynomial**

* **Removal of one root**  **from f(x) to get (n-1)-th order polynomial g(x)**



If ,  with 

If ,



**Therefore, we can get the following relations and pseudo code**

 **🡪**  **🡪**

**If**  **is not the root (not quotient), then there exists a constant residual such as**



* **Removal of two roots**   **from f(x) to get (n-2)-th order polynomial g(x) when the order of polynomial n is greater than 2 (n=3,5,…).**

 **,where** 

 (1)

 (2)

**Therefore, we can get the following relations and pseudo code**

 **🡪**  **(3)**

**🡪** (4)

**If**  **is not the root (not quotient), then there exists a 1st order residual**

 (5)

**On the other hand, if  with , the equation becomes**



**Therefore, we can obtain two solutions by solving  such as**

****

**And, the following equation is left for the next roots corresponding to the following equation**

 (6)

**Here, Eq (6) can be expressed when **

 🡪

**Therefore, we have an important question how to make  by adjusting**  and .

* **As shown by Eq (3) and (4), all coefficients of ,  are the functions of . And,  can be satisfied by the following relations**

**🡪 **

**which is the nonlinear algebraic equation and can be solved using the Newton-Raphson method such as**

****

* **Examples of Polynomial Deflation**

**(a) Formula**



**(b) Example 1: 3rd order polynomial**

**🡪**

 **🡪** **🡪**

**(c) Example 2: 4th order polynomial**

**🡪**

**🡪**



**i)** **🡪**



**ii)  ,which are not solution since  is a complex number**

**iii)**  **,which are not solution since  is a complex number**

**(c) Example 3: 5th order polynomial**

**🡪**



**🡪**

**Highly complex, use a numerical method such as the Newton-Raphson method**

1. **Muller’s Method to find one real root: Local quadratic approximation of function**

* **Local quadratic approximation of f(x) with given 3-point data such as**

**where** 

 🡪 

 🡪 

 🡪 

**Using the last two equations, we can calculate coefficients a and b as follows**

Let

 

Then,

 🡪

* **Approximated roots of local quadratic approximation of f(x)**



🡪

**Choose the nearest x to as the approximated solution of the root**

**Then, repeat the above procedure after the following shifting in three points**

 🡪 

 🡪 

 🡪 

1. **Bairstow’s Method: Newton like method to find 1st or 2nd order quotient of the given polynomial.**

**(If we find a 2nd order quotient, we can calculate complex pair of roots or two real roots.)**

* **Background Rationale**

- Assume an approximated 2nd order quotient as the form

 with two roots of  (reals or complex pair)

- Since approximates the quotient, there exists the residual such as

 ,where the residual R(x) is the 1st order and can be

calculate using a method similar to Polynomial Deflation.



- If , then  is the true quotient and we can find two roots by solving 

**(Question) How to define**  **in**  **to meet** **for all x.**

Answer: By solving for 



However, are the functions of  , Therefore, we should solve the nonlinear system

of equations



**(Question) How to solve above nonlinear algebraic equation? Answer: Newton Raphson Method**

* **Newton Raphson Method revisited**

- Definition of the system of nonlinear equations

 , which has n unknowns and n nonlinear equations

- **Newton Raphson Method**





* **Bairstow’s Method**



 

Using the Newton-Raphson method, 



For RHS of the equation, use 

**(Question) How to estimate** **?**

**Answer: use the central difference formula**

1. **Pseudo code for Bairstow’s Method to fine two roots for the polynomial f(x) = 0 and the quotient polynomial after the polynomial deflation using the quadratic quotient q(x) = x\*x – r\*x –s.**

**Function Bairstow(n, a, IT\_max, epsilon, b, rr, ir, res)**

**!-----------------------------------------------------------------------------------------------------------------------**

**!n: (input) order of the polynomial (n>2)**

**!a(0:n): (input) coefficient of the polynomial**

**!IT\_max: (input) maximum allowed iteration number**

**!epsilon: (input) tolerance in function residual**

**!b(0:n-1): (output) coefficient of quotient polynomial after Polynomial Deflation**

**!rr(1:2): (input/output) estimation of real part of two roots**

**!ir(1:2) ): (input/output) estimation of imaginary part of two roots**

**!res(1:2): (output) residual polynomial coefficient as the form R(x)=r(2)x+r(1)**

**!**

**!res0(1:2): (local) residual due to zero perturbation**

**!resp(1:2): (local) residual due to positive perturbation**

**!resm(1:2): (local) residual due to negative perturbation**

**!grad(1:2,1:2) (local) gradient estimation using central difference**

**!-----------------------------------------------------------------------------------------------------------------------**

**if a(n)=0, exit with notice of “polynomial order is less than n”**

**if (imag\_root(1)+ imag\_root(2)) != 0, exit with notice of “roots are not complex pair”**

**! define quadratic quotient of the form** 

**r = rr(1) + rr(2)**

**s = - rr(1)\*rr(2) + ir(1)\*ir(2)**

**!define small perturbation for central difference formula**

**dr= 0.01;**

**ds= 0.01;**

**!iteration of Newton-Raphson to find the** **which reduces the residual near to zero.**

**do iter =1, IT\_max**

**call FUNCTION Poly\_Defl\_two(n, a, -r,-s, b, res0)**

**!central difference formula to calculate gradient for the residual function**

**rp = r+dr; call FUNCTION Poly\_Defl\_two(n, a, -rp,-s, b, resp)**

**rm = r-dr; call FUNCTION Poly\_Defl\_two(n, a, -rm,-s, b, resm)**

**grad(1:2,1) = 0.5\*(resp(1:2) - resm(1:2))/dr**

**sp = s+ds; call FUNCTION Poly\_Defl\_two(n, a, -r,-sp, b, resp)**

**sm = s-ds; call FUNCTION Poly\_Defl\_two(n, a, -r,-sm, b, resm)**

**grad(1:2,2) = 0.5\*(resp(1:2) - resm(1:2))/ds**

**!update the quotient polynomial**

**(r0;s0)🡨 (r;s)**

**(r;s) 🡨 (r;s) – inv(grad)\*(res0(1), res0(2))**

**!termination condition**

**if norm(res0) < epsilon, exit**

**if sqrt((r-r0)\*(r-r0)+ (s-s0)\*(s-s0)) < epsilon, exit**

**!**

**end do**

**!**

**call FUNCTION Poly\_Defl\_two(n, a, -r,-s, b, res)**

**call Function Quadroot(1,-r,-s,r1,r2,i1,i2,nr)**

**rr(1) = r1; rr(2)= r2;ir(1)= i1, ir(2)=i2;**

**!-----------------------------------------------------------------------------------------------------------------------**

**End Bairstow**

1. **Pseudo code to find N-roots of f(x) = 0 using Bairstow’s Method**

**i) check N ( N=1 or N=2)**

**if N=1, return after finding one real root**

**if N=2, returen after finding two roots**

**root\_real(1) 🡨 real part of one root**

**root\_imag(1) 🡨 imaginary part of one root**

**root\_real(2) 🡨 real part of the other root**

**root\_imag(2) 🡨 imaginary part of the other root**

**ii) set M= int(N/2)+1, k=0, NR=0**

**iii) repeat util k=M**

**k=k+1**

**(a) calculate the quotients of** 





**(b) calculate two roots of** 

**root\_real(NR+1) 🡨 real part of one root**

**root\_imag(NR+1) 🡨 imaginary part of one root**

**root\_real(NR+2) 🡨 real part of the other root**

**root\_imag(NR+2) 🡨 imaginary part of the other root**

**NR=NR+2 🡨 number of roots found**

**(c) Check the order of** 

**NR\_remained = N-NR**

**check NR\_remained NR\_remained=1 or NR\_remained=2)**

**find a root when NR\_remained=1**

**find two roots when NR\_remained=2**

**return**

**end if**

**(d) redefine** 



**(e) repeat step iii)**

1. **Library and Packages for root location**

**- Excel**

**- Matlab**

**- IMSL**

**- Matlib Libraries**