# Numerical Methods I

## Roots of Polynomials: The Birge Vieta Method

**program** main

**implicit** **none**

**integer** :: degreeOfPolynomial

**real**, **dimension**(:), **allocatable** :: a

**real**, **dimension**(:), **allocatable** :: roots

**real**, **parameter** :: tolerance = 1.0e-6

**real** :: xGuess

**integer** :: counter

**write**(\*,\*) "The Birge Vieta Method"

**write**(\*,\*)

! Illustration

! P(x) = x^5 - 7x^4 - 3x^3 + 79x^2 - 46x - 120 = 0

! Roots: -3, -1, 2, 4, 5

degreeOfPolynomial = 5

**allocate**(a(degreeOfPolynomial + 1))

**allocate**(roots(degreeOfPolynomial))

a(1) = 1

a(2) = -7

a(3) = -3

a(4) = 79

a(5) = -46

a(6) = -120

xGuess = 1000.0

**call** birgeVieta(xGuess, a, degreeOfPolynomial, tolerance, roots)

**write**(\*,\*) "Roots:"

**do** counter = 1, degreeOfPolynomial

**write**(\*,10) roots(counter)

**end** **do**

10 **format**(f7.2)

**end** **program** main

**subroutine** birgeVieta(xGuess, aIn, degreeOfPolynomial, tolerance, roots)

**implicit** **none**

**real**, **intent**(in) :: xGuess

**integer**, **intent**(in) :: degreeOfPolynomial

**real**, **intent**(in), **dimension**(degreeOfPolynomial + 1) :: aIn

**real**, **intent**(in) :: tolerance

**real**, **intent**(out), **dimension**(degreeOfPolynomial) :: roots

**real**, **dimension**(:), **allocatable** :: a

**real**, **dimension**(:), **allocatable** :: b

**real**, **dimension**(:), **allocatable** :: c

**real** :: x, xPrevious

**real** :: error

**integer** :: numberOfUnfoundRoots

**integer** :: iteration = 0

**integer** :: counter

**allocate**(a(degreeOfPolynomial + 1))

**allocate**(b(degreeOfPolynomial + 1))

**allocate**(c(degreeOfPolynomial))

a = aIn

b = 0

c = 0

numberOfUnfoundRoots = degreeOfPolynomial

**do** **while**(numberOfUnfoundRoots > 0)

**write**(\*,20) "Finding root #", ((degreeOfPolynomial - numberOfUnfoundRoots) + 1),&

"(Roots not found yet: ", numberOfUnfoundRoots, ")"

**write**(\*,\*) "Coefficients of P(x):"

**do** counter = 1, *size*(a)

**write**(\*,40, advance="no") a(counter), "|"

**end** **do**

**write**(\*,\*)

iteration = 0

x = xGuess

**do** **while**((error > tolerance) .**or**. (iteration <= 2))

iteration = iteration + 1

b(1) = a(1)

**do** counter = 2, *size*(b)

b(counter) = a(counter) + (x \* b(counter - 1))

**end** **do**

c(1) = b(1)

**do** counter = 2, *size*(c)

c(counter) = b(counter) + (x \* c(counter - 1))

**end** **do**

**if**(c(*size*(c)) == 0) **then**

**stop** "Error-- derivative of f(x) in the Newton-Raphson method is zero."

**end** **if**

x = x - (b(*size*(b))) / (c(*size*(c)))

error = *abs*(x - xPrevious)

xPrevious = x

**end** **do**

numberOfUnfoundRoots = numberOfUnfoundRoots - 1

roots(degreeOfPolynomial - numberOfUnfoundRoots) = x

**write**(\*,30) "Root #", (degreeOfPolynomial - numberOfUnfoundRoots), ": ", x

**write**(\*,\*)

**if**(numberOfUnfoundRoots == 1) **then**

**write**(\*,\*) "Q(x) is now linear. Evaluating the last root directly..."

roots(degreeOfPolynomial) = -(b(2) / b(1))

**write**(\*,30) "Root #", degreeOfPolynomial, ": ", roots(degreeOfPolynomial)

**write**(\*,\*)

**return**

**end** **if**

**deallocate**(a)

**allocate**(a(numberOfUnfoundRoots + 1))

**do** counter = 1, (numberOfUnfoundRoots + 1)

a(counter) = b(counter)

**end** **do**

**write**(\*,\*) "---"

**deallocate**(b)

**deallocate**(c)

**allocate**(b(numberOfUnfoundRoots + 1))

**allocate**(c(numberOfUnfoundRoots))

b = 0

c = 0

**end** **do**

20 **format**(a14, i2, a23, i2, a1)

30 **format**(a7, i2, a2, f7.2)

40 **format**(f7.2, a3)

**end** **subroutine** birgeVieta

### Output

The Birge Vieta Method

Finding root # 1 (Roots not found yet: 5)

Coefficients of P(x):

1.00 | -7.00 | -3.00 | 79.00 | -46.00 |-120.00 |

Root # 1: 5.00

---

Finding root # 2 (Roots not found yet: 4)

Coefficients of P(x):

1.00 | -2.00 | -13.00 | 14.00 | 24.00 |

Root # 2: 4.00

---

Finding root # 3 (Roots not found yet: 3)

Coefficients of P(x):

1.00 | 2.00 | -5.00 | -6.00 |

Root # 3: 2.00

---

Finding root # 4 (Roots not found yet: 2)

Coefficients of P(x):

1.00 | 4.00 | 3.00 |

Root # 4: -1.00

Q(x) is linear. Evaluating the last root directly...

Root # 5: -3.00

Roots:

5.00

4.00

2.00

-1.00

-3.00