CH2120

Class 17

# mainProgram.f08

**program** mainProgram

**implicit** **none**

**call** mainNewtonRaphson()

**end** **program** mainProgram

# mainNewtonraphson.f08

**subroutine** mainNewtonRaphson

**implicit** **none**

! Variables: User Input

**real** xGuess

**real** tolerance

! Variables: Console Output

**real** root, error

**integer** iterations

**character**(*len*=25) methodName

! Variables: Internal

**real** fOfX, dFByDx

! Get user input for the (1) initial guess and (2) tolerance.

**write**(\*,\*) "Enter the initial guess for the root:"

**read**(\*,\*) xGuess

**write**(\*,\*) "Enter the tolerance:"

**read**(\*,\*) tolerance

! Evaluate function and derivative at the initial guess.

**call** rootFindingFunctionAndDerivative(xGuess, fOfX, dfByDx)

! Display the column headers of the output table.

**write**(\*,10) "|", "Method", "|", "Root", "|", "Error", "|", "Iterations", "|"

! Check: Is the initial guess a root?

**if**(fOfX == 0) **then**

root = xGuess

error = 0.0

iterations = 0

methodName = "None"

**call** displayRoot(methodName, root, error, iterations)

**return**

**end** **if**

! Get and display the root as estimated by the Newton-Raphson method.

methodName = "Newton-Raphson"

**call** newtonRaphsonRoot(xGuess, tolerance, root, error, iterations)

**call** displayRoot(methodName, root, error, iterations)

10 **format**(a3, a25, a3, a10, a3, a10, a3, a12, a3)

**end** **subroutine** mainNewtonRaphson

# newtonRaphsonRoot.f08

**subroutine** newtonRaphsonRoot(xGuess, tolerance, root, error, iterations)

**implicit** **none**

! Variables: Input Arguments

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

! Variables: Output Arguments

**real**, **intent**(out) :: root, error

**integer**, **intent**(out) :: iterations

! Variables: Internal: function and derivative

**real** x, xPrevious

**real** fOfX, dfByDx

! Initialize internal variables.

x = xGuess

! Initialize output arguments (except error).

root = x

iterations = 0

! Iterations

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

iterations = iterations + 1

**call** rootFindingFunctionAndDerivative(x, fOfX, dfByDx)

**if**(dFByDx == 0) **then**

**stop** "Error: The Newton-Raphson method will not converge. Slope of the tangent is zero."

**end** **if**

x = x - (fOfX / dfByDx)

error = *abs*(x - xPrevious)

xPrevious = x

**end** **do**

root = x

**end** **subroutine** newtonRaphsonRoot

# rootFindingFunctionAndDerivative.f08

**subroutine** rootFindingFunctionAndDerivative(x, fOfX, dfByDx)

**implicit** **none**

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

**real**, **intent**(out) :: fOfX, dfByDx

fOfX = (3 \* x) + *sin*(x) - *exp*(x)

dfByDx = (3) + *cos*(x) - *exp*(x)

**end** **subroutine** rootFindingFunctionAndDerivative

# displayRoot.f08

**subroutine** displayRoot(methodName, root, error, iterations)

**implicit** **none**

**character**(*len*=25), **intent**(in) :: methodName

**real**, **intent**(in) :: root, error

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

**write**(\*,10) "|", methodName, "|", root, "|", error, "|", iterations, "|"

10 **format**(a3, a25, a3, f10.4, a3, f10.4, a3, i12, a3)

**end** **subroutine** displayRoot

# Output

## Output 1 [(3 \* x) + *sin*(x) - *exp*(x)]

Enter the initial guess for the root:

5

Enter the tolerance:

1e-6

| Method | Root | Error | Iterations |

|Newton-Raphson | 1.8900 | 0.0000 | 9 |

## Output 2: f(x) is highly nonlinear [x\*\*11 - 1]

Enter the initial guess for the root:

5

Enter the tolerance:

1e-6

| Method | Root | Error | Iterations |

|Newton-Raphson | 1.0000 | 0.0000 | 21 |