**BISECTION**

**Variable Key:**

Y1 - The function to be evaluated

L - Lower bound

U - Upper bound

O - Stopping criteria Option

E - Tolerance, Epsilon (Either y-axis magnitude or percent error)

D - Tolerance, Delta x (Difference between upper and lower bound)

M - Maximum number of iterations

G - The function evaluated at the Lower bound, L

H - The function evaluated at the Upper bound, I

I - Current number of Iterations

X - The current root approximation

F - The function evaluated at the root approximation

Z- The old X value, used in the percent error criteria

**Program: Comments: (not in actual program)**

**//Inputs**

Input "EQUATION:",Y1 //Input equation

Input "LOWER BOUND:",L //Input lower bound

Input "UPPER BOUND:",U //Input upper bound

Repeat O=1 or O=2 or O=3 //Repeat until O equals 1, 2, or 3

Disp "1=F(X) TOLERANCE" //Display options for stopping criteria

Disp "2=DX TOLERANCE"

Disp "3=PERCENT ERROR"

Prompt O //Prompt for stopping criteria option

End

If 0=1 //If option 1 is selected

Then

Input "F(X) TOLERANCE:",E //Input y-axis magnitude tolerance, epsilon

End

If O=2 //If option 2 is selected

Then

Input "F(X) TOLERANCE:",E //Input y-axis magnitude tolerance, epsilon

Input "DX TOLERANCE:",D //Input change in x tolerance, delta x

End

If O=3

Then

Input "PERCENT ERROR:", E //Input percent error tolerance, epsilon

End

Input "MAX ITERATIONS:",M //Input the maximum iterations to run

pgrmEQUATION //Call the Equation sub-program, sets Y1 to the equation specified in the Equation sub-program

Y1(L)→G //Calculate the value of the function at the lower bound

Y1(u)→H //Calculate the value of the function at the upper bound

**//Begin iterative numerical process**

For(I,1,M,1) //Repeat the following steps at least M times

(1/2)\*(L+U)→X //Calculate the approximation, the midpoint between the upper and lower bounds

Y1(X)→F //Calculate the value of the function at the approximation

Disp "X IS:", X //Display the approximation for the iteration

Disp "ITERS IS:",I //Display the number of iterations

Pause

prgmSTPCRTRA //Call the STPCRTRA sub-program, determines if s topping criteria are met

X→Z //Store the x value for the percent error calculation

G\*F→T //Find the product of the function evaluated at the lower bound and approximation

If T>0 //If product is greater than 0, the root is in the upper

Then half of the current interval

F→G //Set the value of the function at the approximation and

X→L the approximation to the values for the lower bound

Else //If the product is less than 0, the root is in the lower half of the current interval

F→H //Set the value of the function at the approximation and

X→U the approximation to the values for the upper bound

End //End if statement

End //End for loop

Disp "NO ROOT" //If an accurate approximation is not found in the gi number of iterations, M, display no root was found

Disp "MAX ITERS",M //Display the maximum number of iterations

Pause

Stop

**False Position**

**Variable Key:**

Y1 - The function to be evaluated

L - Lower bound

U - Upper bound

O - Stopping criteria Option

E - Tolerance, Epsilon (Either y-axis magnitude or percent error)

D - Tolerance, Delta x (Difference between upper and lower bound)

M - Maximum number of iterations

G - The function evaluated at the Lower bound, L

H - The function evaluated at the Upper bound, I

I - Current number of Iterations

X - The current root approximation

F - The function evaluated at the root approximation

**Program: Comments: (not in actual program)**

**//Inputs**

Input "EQUATION:",Y1 //Input equation

Input "LOWER BOUND:",L //Input lower bound

Input "UPPER BOUND:",U //Input upper bound

Repeat O=1 or O=2 or O=3 //Repeat until O equals 1, 2, or 3

Disp "1=F(X) TOLERANCE" //Display options for stopping criteria

Disp "2=DX TOLERANCE"

Disp "3=PERCENT ERROR"

Prompt O //Prompt for stopping criteria option

End

If 0=1 //If option 1 is selected

Then

Input "F(X) TOLERANCE:",E //Input y-axis magnitude tolerance, epsilon

End

If O=2 //If option 2 is selected

Then

Input "F(X) TOLERANCE:",E //Input y-axis magnitude tolerance, epsilon

Input "DX TOLERANCE:",D //Input change in x tolerance, delta x

End

If O=3

Then

Input "PERCENT ERROR:", E //Input percent error tolerance, epsilon

End

Input "MAX ITERATIONS:",M //Input the maximum iterations to run

pgrmEQUATION //Call the Equation sub-program, sets Y1 to the equation specified in the Equation sub-program

Y1(L)→G //Calculate the value of the function at the lower bound

Y1(u)→H //Calculate the value of the function at the upper bound

**//Begin iterative numerical process**

For(I,1,M,1) //Repeat the following steps at least M times

Y1(L)\*U - Y1(U)\*L/ //Calculate the approximation, use the False Position Y1(L)- Y1(U)→X root approximation equation

Y1(X)→F //Calculate the value of the function at the approximation

Disp "X IS:", X //Display the approximation for the iteration

Disp "ITERS IS:",I //Display the number of iterations

Pause

prgmSTPCRTRA //Call the STPCRTRA sub-program, determines if s topping criteria are met

X→Z //Store the x value for the percent error calculation

G\*F→T //Find the product of the function evaluated at the lower bound and approximation

If T>0 //If product is greater than 0, the root is in the upper

Then half of the current interval

F→G //Set the value of the function at the approximation and

X→L the approximation to the values for the lower bound

Else //If the product is less than 0, the root is in the lower half of the current interval

F→H //Set the value of the function at the approximation and

X→U the approximation to the values for the upper bound

End //End if statement

End //End for loop

Disp "NO ROOT" //If an accurate approximation is not found in the gi number of iterations, M, display no root was found

Disp "MAX ITERS",M //Display the maximum number of iterations

Pause

Stop

**Newton-Raphson**

**Variable Key:**

Y1 - The function to be evaluated

Y2 - The analytical derivative of Y1

L - The initial guess

O - Stopping criteria Option

E - Tolerance, Epsilon (Either y-axis magnitude or percent error)

D - Tolerance, Delta x (Difference between upper and lower bound)

M - Maximum number of iterations

G - The function evaluated at the Lower bound, L

H - The function evaluated at the Upper bound, I

I - Current number of Iterations

X - The current root approximation

F - The function evaluated at the root approximation

**Program: Comments: (not in actual program)**

**//Inputs**

Input "EQUATION:",Y1 //Input equation

Input "EQUATION:",Y2 //Input analytical derivative

Input "GUESS:",L //Input initial guess

Repeat O=1 or O=3 //Repeat until O equals 1 or 2

Disp "1=F(X) TOLERANCE" //Display options for stopping criteria

Disp "3=PERCENT ERROR"

Prompt O //Prompt for stopping criteria option

End //End repeat

If 0=1 //If option 1 is selected

Then

Input "F(X) TOLERANCE:",E //Input y-axis magnitude tolerance, epsilon

End

If O=3 //If option 2 is selected

Then

Input "PERCENT ERROR:", E //Input percent error tolerance, epsilon

End

Input "MAX ITERATIONS:",M //Input the maximum iterations to run

pgrmEQUATION //Call the Equation sub-program, sets Y1 and Y2to the equations specified in the Equation sub-program

**//Begin iterative numerical process**

For(I,1,M,1) //Repeat the following steps at least M times

L-(Y1(L)/Y2(L))→X //Calculate the approximation, use the Newton- Raphson root approximation equation

Y1(X)→F //Calculate the value of the function at the approximation

Disp "X IS:", X //Display the approximation for the iteration

Disp "ITERS IS:",I //Display the number of iterations

Pause

prgmSTPCRTRA //Call the STPCRTRA sub-program, determines if the stopping criteria are met

X→Z //Store the x value for the percent error calculation

X→L //Set the root approximation to the next guess

End //End for loop

Disp "NO ROOT" //If an accurate approximation is not found in the gi number of iterations, M, display no root was found

Disp "MAX ITERS",M //Display the maximum number of iterations

Pause

Stop

**Secant**

**Variable Key:**

Y1 - The function to be evaluated

L - Guess on, xn

U - Guess two,xn-1

O - Stopping criteria Option

E - Tolerance, Epsilon (Either y-axis magnitude or percent error)

D - Tolerance, Delta x (Difference between upper and lower bound)

M - Maximum number of iterations

G - The function evaluated at the Lower bound, L

H - The function evaluated at the Upper bound, I

I - Current number of Iterations

X - The current root approximation

F - The function evaluated at the root approximation

**Program: Comments: (not in actual program)**

**//Inputs**

Input "EQUATION:",Y1 //Input equation

Input "LOWER BOUND:",L //Input guess one

Input "UPPER BOUND:",U //Input guess two

Repeat O=1 or O=3 //Repeat until O equals 1, 2, or 3

Disp "1=F(X) TOLERANCE" //Display options for stopping criteria

Disp "3=PERCENT ERROR"

Prompt O //Prompt for stopping criteria option

End

If 0=1 //If option 1 is selected

Then

Input "F(X) TOLERANCE:",E //Input y-axis magnitude tolerance, epsilon

End

If O=3

Then

Input "PERCENT ERROR:", E //Input percent error tolerance, epsilon

End

Input "MAX ITERATIONS:",M //Input the maximum iterations to run

pgrmEQUATION //Call the Equation sub-program, sets Y1 to the equation specified in the Equation sub-program

**//Begin iterative numerical process**

For(I,1,M,1) //Repeat the following steps at least M times

L-(Y1(L)\*(U-L))/(Y1(U) - Y1(L))→X //Calculate the approximation, use the Secant root approximation equation

Y1(X)→F //Calculate the value of the function at the approximation

Disp "X IS:", X //Display the approximation for the iteration

Disp "ITERS IS:",I //Display the number of iterations

Pause

prgmSTPCRTRA //Call the STPCRTRA sub-program, determines if s topping criteria are met

L→U //Set xn to xn-1

X→L //Set xn+1 to xn

End //End for loop

Disp "NO ROOT" //If an accurate approximation is not found in the gi number of iterations, M, display no root was found

Disp "MAX ITERS",M //Display the maximum number of iterations

Pause

Stop

**Equation (subroutine)**

**Variable Key:**

Y1 – Function

Y2 – Analytical Derivative

**Program: Comments: (not in actual code)**

“”→Y1 //Assign the function to Y1, Enter between the quotes

“”→Y2 //Assign the derivative to Y2,Enter between the brackets

Return //Returns

**STPCRTRA (subroutine)**

**Variable Key:**

O – Stopping criteria Option

F – The function evaluated at the root approximation

E - Tolerance, Epsilon (Either y-axis magnitude or percent error)

X – The current root approximation

I – Current number of Iterations

U – Upper bound

L – Lower bound

Z – The root approximation from the previous iteration

**Program: Comments: (not in actual code)**

If O=1 and abs(F)<E //If the absolute value of the function at X is less than

//epsilon

Then

Disp “ROOT IS:”,X //Display the approximation for the iteration

Disp “ITERS IS:”,I //Display the number of iterations

Pause //Pause execution

Stop //Stop the program

End

If O=2 and abs(F)<E and (U-L)<D //If the absolute value of the function is less than

//epsilon and the range between the upper and

//lower bounds is less than delta

Then

Disp “ROOT IS:”,X //Display the approximation for the iteration

Disp “ITERS IS:”,I //Display the number of iterations

Pause //Pause execution

Stop //Stop the program

End

If O=3 and abs(X-Z)/X)\*100<E //If the percent change between the current and

//the previous root is less than epsilon

Then

Disp “ROOT IS:”,X //Display the approximation for the iteration

Disp “ITERS IS:”,I //Display the number of iterations

Pause //Pause execution

Stop //Stop the program

End //End the if statement

Return //Return to the previous routine