Introduction
Ever since HP launched programmable calculators, like the HP-65, HP-67, and the HP-25, it included root-seeking programs in its manuals, standard applications, and math applications. When HP released the HP-34C in 1978, it offered a built-in root Solver, for the first time. The Solver found a single root for a nonlinear function, given two guesses for (and near) a root. HP refined this Solver in later machines to accept an initial guess for and near a root.

This article presents the Scan Range Method and an Excel VBA listing for a multi-root method. The method scans a user-specified range and using user-defined step sizes to examine multiple small intervals. The Scan Range Method finds roots and inflection points (minima, maxima, and saddle points) in the given range. The method relies on two basic algorithms:

- A root-seeking algorithm that locates a root in a sub-interval where the function values at the ends of that sub-interval change signs.
- An optimum-seeking algorithm that locates minima, maxima, and saddle points in sub-intervals where the slopes change signs at the ends of the sub-interval. These points can also double up as roots.

The Algorithm
Here is the pseudo-code for the Scan Range Method:

Initialize mechanism or structure used to report the results
Xa=A
Fa = f(Xa)
Da = d1(Xa)
SFa = sign of Fa
SDa = sign of Da
NumSteps = 0
Repeat
  Increment NumSteps
  Xb = A + NumSteps*StepSize
  Fb=f(Xb)
  Db=d1(Xb)
  SignFb = sign of Fb
  SignDb = sign of Db
  // Xb landed on a root?
  If Fb=0 Then
    Report and/or store Xb, Fb
    If SignDb and SignDa have opposite signs Then
      // Second derivative
      Drv2 = d2(Xb)
      If |Drv2|>=FxToler Then
        If Drv2 < 0 Then
          Report a maximum point
Else
  Report a minimum point
End
Else
  Report a saddle point
End
End
Else If SignFb and SignFa have opposite signs Then
  X = calculated root in [Xa, Xb] with tolerance Toler
  Report and/or store X, f(x)
Else If (SignFa and SignFb have same values) AND
  (SignDa and SignDb do not have the same values) THEN
  X = calculated minima/maxima in [Xa, Xb] with tolerance Toler
  Report and/or store X, f(X)
  Drv2 = d2(X)
  If Drv2 < 0 Then
    Report a maximum point
  Else
    Report a minimum point
  End
End
If found a root, minimum, or maximum Then
  Increment NumSteps
  Xa = A + NumSteps*StepSize
  Fa = f(Xa)
  Da = d1(Xa)
  SignFa = sign of Fa
  SignDa = sign of Da
Else
  Xa = Xb
  Fa = Fb
  Da = Db
  SignFa = SignFb
  SignDa = SignDb
End
Until Xa >= B
Return accumulated information

VBA Listing
Here is the VBA listing (and associated information) for the Scan Range Method: To run the Scan Method
macro, invoke the macro \texttt{ScanRoot}.

Option Explicit

Function MyFx(ByVal sfX As String, ByVal X As Double) As Double
  sfX = Replace(sfX, "$X", "(" & CStr(X) & ")")
  MyFx = Evaluate(sfX)
End Function

Function MySlope(ByVal sfX As String, ByVal X As Double, ByVal Fx As Double) As Double
  Dim h As Double
  MySlope = \text{Evaluate sanitizes the function before returning it as a Double.}
End Function
Function MySecDeriv(ByVal sfX As String, ByVal X As Double) As Double
Dim h As Double, Fp As Double, Fx As Double, Fm As Double
h = 0.001 * (1 + Abs(X))
Fx = MyFx(sfX, X)
Fp = MyFx(sfX, X + h)
Fm = MyFx(sfX, X - h)
MySecDeriv = (Fp - 2 * Fx + Fm) / h / h
End Function

Function NewtonRoot(ByVal sfX As String, ByVal X As Double, ByVal Toler As Double) As Double
Dim h As Double, Fx As Double, Diff As Double
Do
h = 0.001 * (1 + Abs(X))
Fx = MyFx(sfX, X)
Diff = h * Fx / (MyFx(sfX, X + h) - Fx)
X = X - Diff
Loop Until Abs(Diff) < Toler
NewtonRoot = X
End Function

Function NewtonMinMax(ByVal sfX As String, ByVal X As Double, ByVal Toler As Double) As Double
Dim h As Double, Fx As Double, Diff As Double
Dim Fp As Double, Fm As Double, Drv1 As Double, Drv2 As Double
Do
h = 0.001 * (1 + Abs(X))
Fx = MyFx(sfX, X)
Fp = MyFx(sfX, X + h)
Fm = MyFx(sfX, X - h)
Drv1 = (Fp - Fm) / 2 / h
Drv2 = (Fp - 2 * Fx + Fm) / h / h
Diff = Drv1 / Drv2
X = X - Diff
Loop Until Abs(Diff) < Toler
NewtonMinMax = X
End Function

Sub ScanRoot()
'Scan a range and uses Newton's method to find a root when the sign of the function changes.
Const MAX_ERROR = 5
Dim sfX As String
Dim R As Integer, NumSteps As Long, ErrCount As Integer
Dim A As Double, B As Double, Stp As Double, Toler As Double, FxToler As Double
Dim Xa As Double, Fa As Double, SFa As Integer, Da As Double, SDb As Integer
Dim Xb As Double, Fb As Double, SFb As Integer, Db As Double, SDb As Integer
Dim Fx As Double, h As Double, Diff As Double, X As Double, Drv2 As Double
Dim bMoveOneExtraStep As Boolean

sfX = UCASE([A2].Value)
A = [A4].Value
B = [A6].Value
Stp = [A8].Value
Toler = [A10].Value
FxToler = [A12].Value
Range("B2:D1000").Value = ""
Xa = A
Fa = MyFx(sfX, Xa)
Da = MySlope(sfX, Xa, Fa)
SFa = Sgn(Fa)
SDa = Sgn(Da)
R = 2
NumSteps = 0
ErrCount = 0
On Error GoTo ErrHandler
Do
    NumSteps = NumSteps + 1
    bMoveOneExtraStep = False
    Xb = A + NumSteps * Stp
    Fb = MyFx(sfX, Xb)
    Db = MySlope(sfX, Xb, Fb)
    SFb = Sgn(Fb)
    SDb = Sgn(Db)
    ' landed on a root??
    If Abs(Fb) = 0 Then
        Cells(R, 2) = Xb
        Cells(R, 3) = 0
        Drv2 = MySecDeriv(sfX, Xb)
        If SDa * SDb < 0 Then
            If Abs(Drv2) > FxToler Then
                If Drv2 > 0 Then
                    Cells(R, 4) = "Root & Minimum"
                Else
                    Cells(R, 4) = "Root & Maximum"
                End If
            Else
                Cells(R, 4) = "Root & Saddle point"
            End If
        End If
        End If
        R = R + 1
        bMoveOneExtraStep = True
    ' Found a range that contains a root?
    ElseIf SFb * SFa < 0 Then
        X = NewtonRoot(sfX, (Xa + Xb) / 2, Toler)
        Cells(R, 2) = X
        Cells(R, 3) = MyFx(sfX, X)
        R = R + 1
        bMoveOneExtraStep = True
    ' located a range that has a minimum/maximum/root?
    ElseIf SFa * SFb > 0 And SDa * SDb < 0 Then
        X = NewtonMinMax(sfX, (Xa + Xb) / 2, Toler)
        Drv2 = MySecDeriv(sfX, X)
            ' found a root?
        Cells(R, 2) = X
        Cells(R, 3) = MyFx(sfX, X)
        bMoveOneExtraStep = True
        If Abs(Fx) < FxToler Then
            Cells(R, 4) = "Root & 
        Else
            Cells(R, 4) = ""
        End If
        End If
        If Drv2 > 0 Then
            Cells(R, 4) = Cells(R, 4) & "Minimum"
        Else
            Cells(R, 4) = Cells(R, 4) & "Maximum"
        End If
        R = R + 1
ResumeHere:
   If bMoveOneExtraStep Then
      NumSteps = NumSteps + 1
      Xa = A + NumSteps * Stp
      Fa = MyFx(sfX, Xa)
      Da = MySlope(sfX, Xa, Fa)
      SFa = Sgn(Fa)
      SDa = Sgn(Da)
   Else
      Xa = Xb
      Fa = Fb
      Da = Db
      SFa = SFb
      SDa = SDb
   End If
   Loop Until Xa >= B

ErrHandler:
   ErrCount = ErrCount + 1
   If ErrCount > MAX_ERROR Then
      MsgBox "Reached maximum limit or runtime errors!"
      Resume ExitProc
   Else
      bMoveOneExtraStep = True
      Resume ResumeHere
   End If
End Sub

Sample Session
Here is a sample session that finds the roots and inflection points for the function:

\[ f(x) = \exp(x) - 3\cdot x^2 \]

The sample session searches for the roots and inflection points using the following input data:

- The range of \([-1, 4]\).  
- The search step size of 0.1.  
- The tolerance value of 1e-6.  
- The function tolerance value of 1e-4

The following table represents a partial view of an Excel worksheet. The cells that require your input have their contents in red. Note that cell A2 has an expression for the mathematical function. The variable X appears in the expression as $X. Using the $ character allows the Excel VBA code to distinguish the variable X from the letter X that may be part of a function, such as \(\exp\). Column A is the input column, while columns B, C, and D show the output after executing the macro ScanRoots().
<table>
<thead>
<tr>
<th>Function</th>
<th>Roots</th>
<th>FX</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXP($X)-3*$X*$X</td>
<td>-0.458962268</td>
<td>3.74145E-13</td>
</tr>
<tr>
<td>A</td>
<td>0.204481511</td>
<td>1.101450707</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>-5.8562E-12</td>
</tr>
<tr>
<td>B</td>
<td>2.833144107</td>
<td>-7.081293582</td>
</tr>
<tr>
<td>Search Step</td>
<td>4</td>
<td>3.733079029</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.68612E-10</td>
</tr>
<tr>
<td>Tolerance</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.00E-08</td>
<td></td>
</tr>
<tr>
<td>Fx Tolerance</td>
<td>1.00E-10</td>
<td></td>
</tr>
</tbody>
</table>