# Scan Range Method for the HP-41C

By Namir Shammas

## 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 HP-41C 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
```

## HP-41C Listing

Here is the HP-41C listing (and associated information) for the Scan Range Method:

### Memory Map

R00 = X
R01 = Fx
R02 = A
R03 = B
R04 = Step
R05 = Toler

R06 = FxToler
R07 = NumSteps
R08 = Xa
R09 = Fa
R10 = Da
R11 = SFa
R12 = SDa
R13 = Xb
R14 = Fb
R15 = Db
R16 = SFb
R17 = SDb
R18 = not used
R19 = Drv1
R20 = Drv2
R21 = X
R22 = h
R23 = Fx
R24 = Fp
R25 = Fm

## Flags

F00 = Move One Extra Step (MOES)

## HP-41C Listing

Here is the listing for the Scan Range Method. Please note that the code for executing the targeted mathematical function appears after LBL 00. To change the mathematical function (currently coded as f(x)=exp(x)-3*x^2) edit the code after LBL 00. To execute the program for the first time, perform [XEQ][Alpha]SCAN[Alpha]. For subsequent runs, simply press the [A] key when User mode is activated.

```
LBL SCAN
GTO A

LBL E         # Calculate f(x)
LBL 00
EXP
LASTX
X^2
3
*
-
RTN

LBL 05        # Calculate & store h = 0.001*(1 + ABS(X))
ABS
1
+
.001
*
STO 22
```

```
RTN

LBL 01         # Calculate f'(x) given x and f(x) in X and Y registers
STO 21
X<>Y
STO 23
X<>Y
XEQ 05         # Calculate h
RCL 21
+              # Calculate x+h
XEQ 00         # Calculate f(x+h)
RCL 23
-
RCL 22
/              # Calculate (f(x+h) - f(x))/ h
RTN

LBL 02         # Calculate f''(x) given x
STO 21
XEQ 05         # Calculate h
RCL 21
XEQ 00         # Calculate f(x)
STO 23
RCL 21
RCL 22
+
XEQ 00         # Calculate f(x+h)
STO 24
RCL 21
RCL 22
-
XEQ 00         # Calculate f(x-h)
STO 25
RCL 24
+
RCL 23
STO+ X
-
RCL 22
X^2
/              # Calculate (f(x+h) - 2*f(x) + f(x-h))/h^2
RTN

LBL 03         # Calculate the root of f(x) using Newton's method
STO 21
LBL 17         # ---------------- start of itearions loop
RCL 21
XEQ 05         # Calculate h
RCL 21
XEQ 00         # Calculate f(x)
STO 23
RCL 21
RCL 22
+
XEQ 00        # Calculate f(x+h)
RCL 23
-
RCL 23
X<>Y
/
RCL 22
*             # Calculate diff = h * f(x)/(f(x+h) - f(x))
STO- 21       # Calculate X = X - diff
ABS
RCL 05
X<Y?
GTO 17        # --------------- end of iterations loop
RCL 21
RTN

LBL 04        # Calculate the root of f'(x) using Newton's method
STO 21
LBL 18        # ----------------- start of itearions loop
RCL 21
XEQ 05        # Calculate h
RCL 21
XEQ 00        # Calculate Fx
STO 23
RCL 21
RCL 22
+
XEQ 00        # Calculate f(x+h)
STO 24
RCL 21
RCL 22
-
XEQ 00        # Calculate f(x-h)
STO 25
RCL 24
+
RCL 23
STO+ X
-
RCL 22
X^2
STO 20        # Calculate and store second derivative
RCL 24
RCL 25
-
2
/
RCL 22
/             # Calculate first derivative
RCL 20
```

```
/             # Calculate diff = f'(x) / f''(x)
STO- 21       # Calculate X = X - diff
ABS
RCL 05
X<Y?
GTO 18        # --------------- end of iterations loop
RCL 21
RTN

LBL A         # Start implementation of SCAN function
"A^B?"
PROMPT
STO 03
X<>Y
STO 02
"STEP?"
0.1
PROMPT
STO 04
"TOLER?"
1E-6
PROMPT
STO 05
"FXTOLER?"
1E-4
PROMPT
STO 06
0
STO 07        # Initialize NumSteps
RCL 02
STO 08        # Xa = A
XEQ 00        # Calculate Fa
STO 09
RCL 08
XEQ 01        # Calculate Da
STO 10
SIGN          # Calculate sign(Da)
ST0 12
RCL 09
SIGN          # Calculate sign(Fa)
STO 11
LBL 06        # ------------------------------ MAIN LOOP
CF 00         # Clear MOES flag
1
STO+ 07       # Increment NumSteps
RCL 02
RCL 04
RCL 07
*
+             # Calculate Xb
VIEW X
STO 13
```

```
XEQ 00        # Calculate Fb
STO 14
RCL 13
XEQ 01        # Calculate Db
STO 15
SIGN          # Calculate sign(Db)
ST0 17
RCL 14
SIGN          # Calculate sign(Fb)
STO 16

RCL 14
X#0?          # F(Xb) <> 0?
GTO 07
SF 00         # Set MOEs flag
"X="          # Xb is an exact root
ARCL 13
PROMPT
"FX=0"
PROMPT

RCL 12
RCL 17
*
X>0?          # SignDa * SignDb > 0?
GTO 09
RCL 13
XEQ 02        # Calculate f''(Xb)
STO 20
ABS
RCL 06
X>Y?          # FxToler>|f''(x)|?
GTO 12
RCL 20
X>0?
GTO 14
"ROOT/MAX"
PROMPT
GTO 09
LBL 14
"ROOT/MIN"
PROMPT
GTO 09
LBL 12
"ROOT/SADDLE"
PROMPT
GTO 09

LBL 07        # Xb is not a root!
RCL 11
RCL 16
*
```

```
X>0?
GTO 08
SF 00         # Set MOEs flag
RCL 08
RCL 13
+
2
/             # Calculate (Xa + Xb) / 2 and use it as initial guess
for a root
XEQ 03        # Calculate root using Newton's method
STO 00
"X="
ARCL 00
PROMPT
RCL 00
XEQ 00        # Calculate Fx
"FX="
ARCL X
PROMPT
GTO 09

LBL 08
RCL 12
RCL 17
*
X<0?
GTO 09
RCL 12
RCL 17
*
X>0?
GTO 09
SF 00         # Set MOES flag
RCL 08
RCL 13
+
2
/             # Calculate X = (Xa + Xb) / 2 and use it as
              # initial guess for a root
XEQ 04        # Calculate root of f'(X) using Newton's method
STO 00
"X="
ARCL X
PROMPT
RCL 00
XEQ 00        # Calculate f(x)
“FX=”
ARCL X
PROMPT        # Display f(x)
CLA           # Clear alpha register
RCL 00
XEQ 00        # Calculate f(x)
```

```
ABS
RCL 06
X>Y?
|-"ROOT/"     # Append "root/" to alpha register
              # and prepare for either min or max
RCL 00
XEQ 02        # Calculate f''(x)
STO 20
X>0?
GTO 16
|-"MAX"       # Append "max" to alpha register
PROMPT
GTO 09
LBL 16
|-"MIN"       # Append "min" to alpha register
PROMPT

LBL 09
FC?C 00       # Is MOES flag clear?
GTO 10
1             # Move search by one extra step
STO+ 07
RCL 02
RCL 04
RCL 07
*
+             # Calculate Xa
STO 08
XEQ 00        # Calculate Fa
STO 09
RCL 08
XEQ 01        # Calculate Da
STO 10
SIGN          # Calculate sign(Da)
ST0 12
RCL 09
SIGN          # Calculate sign(Fa)
STO 11
GTO 11

LBL 10        # Perform a regular step forward
13.017
ENTER
8.012
LBL 20        # Loop to copy register bloc
RCL IND Y
STO IND Y
RDN
ISG Y
STO X
ISG X
GTO 20
```

```
LBL 11
RCL 03
RCL 08       # Xa < B?
X<Y?
GTO 06
"DONE"
PROMPT
RTN
```

## Sample Session

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

f(x) = exp(x) – 3*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

```
                  [XEQ]  "SCAN”
A^B?
         -1.00000  ENTER
          4.00000    RUN
STEP?
          0.1        RUN
TOLER?
          1E-6       RUN
FXTOLER?
          1E-4       RUN
X=-0.45896
                     RUN
FX=-3.00000E-10
                     RUN
X=0.20448
                     RUN
FX=1.10145
                     RUN
MAX
                     RUN
X=0.91001
                     RUN
FX=0.00000
                     RUN
X=2.83314
                     RUN
FX=-7.08129
                     RUN
```

```
MIN
                     RUN
X=3.73308
                     RUN
FX=0.00000
                     RUN
DONE
```

The function has three roots, one maximum point, and one minimum point.