Scan Range Method for Matlab

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 a Matlab 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
```

Matlab Listing

Here is the Matlab listing for the Scan Range Method. The scanroots() function implements this method:

```
function [r, numpts] = scanroots(sfx, a, b, stepSize, toler, fxtoler)
% Scan a range and uses newton's method to find a root when the sign of the
% function changes. The function displays the results in a formatted table
% and also returns these results so you can further work with them.
%
% Copyright (c) 2012 Namir Clement Shammas
% Version 1.0.0
% Date 7/16/2012
%
```

```
% UPDATES
% =======
%
%
% INPUT
% =====
% sfx - string containing the expression for f(x)
% a, b - the range [a,b] to scan
% stepSize - the step size for the scan
% toler - tolerance for the roots or minima/maxima
% fxtoler - toleranc eused for the second derivative
%
% OUTPUT
% ======
% r = an array of structures containing the following fields:
%     x - value of x at point
%     fx - function value at x
%     type - name of point (root, minima, etc.)
% numpts - number of results
%
%
% Example 1
% ----------
%
%  >> [~, ~] = scanroots('exp(x)-3*x^2', -1, 4, .1, 1e-8, 1e-4)
%
%  Scanning function y = exp(x)-3*x^2 from -1 to 4
%
%       x                 f(x)            Type
% --------------------------------------------------
% -4.58962268e-01   +3.74367204e-13    Root
% +2.04481511e-01   +1.10145071e+00    Maximum
% +9.10007572e-01   -5.85664850e-12    Root
% +2.83314411e+00   -7.08129358e+00    Minimum
% +3.73307903e+00   +5.68576297e-10    Root
%
%
% Example 2
% ----------
%
% >> [~, ~] = scanroots('sin(x)+1', -1, 20, .1, 1e-8, 1e-4)
%
%
% Scanning function y = sin(x)+1 from -1 to 20
%
%       x                 f(x)            Type
% --------------------------------------------------
% +1.57079633e+00   +2.00000000e+00    Maximum
% +4.71238898e+00   +0.00000000e+00    Root/Minimum
% +7.85398163e+00   +2.00000000e+00    Maximum
% +1.09955743e+01   +0.00000000e+00    Root/Minimum
% +1.41371669e+01   +2.00000000e+00    Maximum
% +1.72787596e+01   +0.00000000e+00    Root/Minimum
%
%
% Example 3
% ----------
```

```
%
% >> [~, ~] = scanroots('(x-5.5)*(x-1.5)*(x+2.5)*(x+4.5)', -20, 20, .1, 1e-8,
1e-4)
%
%
% Scanning function y = (x-5.5)*(x-1.5)*(x+2.5)*(x+4.5) from -20 to 20
%
%       x                f(x)            Type
% --------------------------------------------------
% -4.50000000e+00   +0.00000000e+00    Root
% -3.64842447e+00   -4.60623683e+01    Minimum
% -2.50000000e+00   +0.00000000e+00    Root
% -3.59070941e-01   +9.65661210e+01    Maximum
% +1.50000000e+00   +0.00000000e+00    Root
% +4.00749549e+00   -2.07191253e+02    Minimum
% +5.50000000e+00   +0.00000000e+00    Root
%
%

  global gExpression; % stores string for expression

  if nargin < 6, fxtoler = 1e-4; end;
  if nargin < 5, toler = 1e-8; end;
  if nargin < 4, stepSize = (b-a)/100; end;

  gExpression= sfx;
  xa = a;
  fa = myfx(xa);
  da = myslope(xa,fa);
  sfa = sign(fa);
  sda = sign(da);
  numpts = 0;
  numsteps = 0;
  errcount = 0;
  max_error = 5;
  xp=[xa];
  yp=[fa];
  ymin = 1e+9;
  ymax = -1e+9;
  if fa < ymin, ymin=fa; end
  if fa > ymax, ymax=fa; end
  formatStr = '%+12.8e   %+12.8e    %s\n';

  fprintf('\n\n');
  fprintf('Scanning function y = %s from %g to %g\n\n',  sayMyFx(), a, b);
  fprintf('      x                f(x)            Type\n');
  fprintf('--------------------------------------------------\n');

  % ------------------ start main loop ------------------
  while xa < b
 try
    numsteps = numsteps + 1;
    bMoveOneExtraStep = false;
    xb = a + numsteps * stepSize;
    fb = myfx(xb);
    db = myslope(xb,fb);
    sfb = sign(fb);
```

```
sdb = sign(db);
% store xb and fb in arrays x and yp
xp=[xp xb];
yp=[yp fb];
% update minimum an dmaximum for y
if fb < ymin, ymin=fb; end
if fb > ymax, ymax=fb; end

% landed on a root??
if abs(fb) == 0 % then
  numpts = numpts + 1;
  r(numpts).x = xb;
  r(numpts).fx = 0;
  r(numpts).type = 'Root';
  drv2 = mysecderiv(xb);
  if sda * sdb < 0 % then
    if abs(drv2) > fxtoler % then
      if drv2 > 0 % then
        r(numpts).type = 'Root & Minimum';
      else
        r(numpts).type = 'Root & Maximum';
      end % if
    else
      r(numpts).type = 'Root & Saddle point';
    end % if
  end % if
  %fprintf('Point number %g\n', numpts);
  %disp(r(numpts));
  fprintf(formatStr,  r(numpts).x, r(numpts).fx, r(numpts).type);
  bMoveOneExtraStep = true;
% found a range that contains a root?
elseif sfb * sfa < 0 % then
  x = newtonroot((xa + xb) / 2, toler);
  numpts = numpts + 1;
  fx = myfx(x);
  r(numpts).x = x;
  r(numpts).fx = fx;
  r(numpts).type = 'Root';
  %fprintf('Point number %g\n', numpts);
  %disp(r(numpts));
  fprintf(formatStr,  r(numpts).x, r(numpts).fx, r(numpts).type);
  bMoveOneExtraStep = true;
% located a range that has a minimum/maximum/root?
elseif (sfa * sfb > 0) && (sda * sdb < 0) % then
  x = newtonminmax((xa + xb) / 2, toler);
  drv2 = mysecderiv(x);
  % found a root?
  numpts = numpts + 1;
  r(numpts).x = x;
  fx = myfx(x);
  r(numpts).fx =fx;
  bMoveOneExtraStep = true;
  if abs(fx) < fxtoler % then
    r(numpts).type = 'Root/';
  else
     r(numpts).type = '';
  end % if
```

```
      if drv2 > 0 % then
        r(numpts).type = strcat(r(numpts).type, 'Minimum');
      else
        r(numpts).type = strcat(r(numpts).type, 'Maximum') ;
      end % if
      %fprintf('Point number %g\n', numpts);
      %disp(r(numpts));
      fprintf(formatStr,  r(numpts).x, r(numpts).fx, r(numpts).type);
    end % if

    if bMoveOneExtraStep % then
      numsteps = numsteps + 1;
      xa = a + numsteps * stepSize;
      fa = myfx(xa);
      da = myslope(xa,fa);
      sfa = sign(fa);
      sda = sign(da);
    else
      xa = xb;
      fa = fb;
      da = db;
      sfa = sfb;
      sda = sdb;
    end % if

 catch me
  errcount = errcount + 1;
  if errcount > max_error % then
    disp('Reached maximum limit or runtime errors!');
    return
  else
    numsteps = numsteps + 1;
    xa = a + numsteps * stepSize;
    fa = myfx(xa);
    da = myslope(xa,fa);
    sfa = sign(fa);
    sda = sign(da);
  end % if
 end % catch
  end % while  ------------------ end main loop ------------------
  fprintf('\n\n');
  % plot data and show funcation name in the title
  plot(xp,yp);
  %axis([a b ymin umax]);
  line([a b],[0 0]);
  line([0 0],[1.2*ymin 1.2*ymax]);
  title(strcat('y=',gExpression));
end % function scan

function s = sayMyFx()
  global gExpression;
  s = gExpression;
end

function y = myfx(x)
  y = eval(sayMyFx());
```

```
end

function y = myslope(x,fx)
  h = 0.001 * (1 + abs(x));
  y = (myfx(x + h) - fx) / h;
end

function y = mysecderiv(x)
  h = 0.001 * (1 + abs(x));
  fx = myfx(x);
  fp = myfx(x + h);
  fm = myfx(x - h);
  y = (fp - 2 * fx + fm) / h / h;
end

function y = newtonroot( x, toler)
  diff = 10*toler;
  while abs(diff) >= toler
    h = 0.001 * (1 + abs(x));
    fx = myfx(x);
    diff = h * fx / (myfx(x + h) - fx);
    x = x - diff;
  end
  y = x;
end

function y = newtonminmax( x, toler)
  diff = 10*toler;
  while abs(diff) >= toler
    h = 0.001 * (1 + abs(x));
    fx = myfx(x);
    fp = myfx(x + h);
    fm = myfx(x - h);
    drv1 = (fp - fm) / 2 / h;
    drv2 = (fp - 2 * fx + fm) / h / h;
    diff = drv1 / drv2;
    x = x - diff;
  end
  y = x;
end
```

Sample Sessions

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-8.
- The function tolerance value of 1e-4

```
>> [~, ~] = scanroots('exp(x)-3*x^2', -1, 4, 0.1, 1e-8, 1e-4)
```



```
 Scanning function y = exp(x)-3*x^2 from -1 to 4

      x                 f(x)            Type
-------------------------------------------------
-4.58962268e-01   +3.74367204e-13    Root
+2.04481511e-01   +1.10145071e+00    Maximum
+9.10007572e-01   -5.85664850e-12    Root
+2.83314411e+00   -7.08129358e+00    Minimum
+3.73307903e+00   +5.68576297e-10    Root
```

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

f(x) = sin(x)+1

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

- The range of [-1, 20].
- The search step size of 0.1.
- The tolerance value of 1e-8.
- The function tolerance value of 1e-4

```
>> [~, ~] = scanroots('sin(x)+1', -1, 20, .1, 1e-8, 1e-4)


   Scanning function y = sin(x)+1 from -1 to 20

      x                 f(x)            Type
-------------------------------------------------
+1.57079633e+00   +2.00000000e+00    Maximum
+4.71238898e+00   +0.00000000e+00    Root/Minimum
+7.85398163e+00   +2.00000000e+00    Maximum
+1.09955743e+01   +0.00000000e+00    Root/Minimum
+1.41371669e+01   +2.00000000e+00    Maximum
+1.72787596e+01   +0.00000000e+00    Root/Minimum
```

