LMFnlsq – Nonlinear least squares

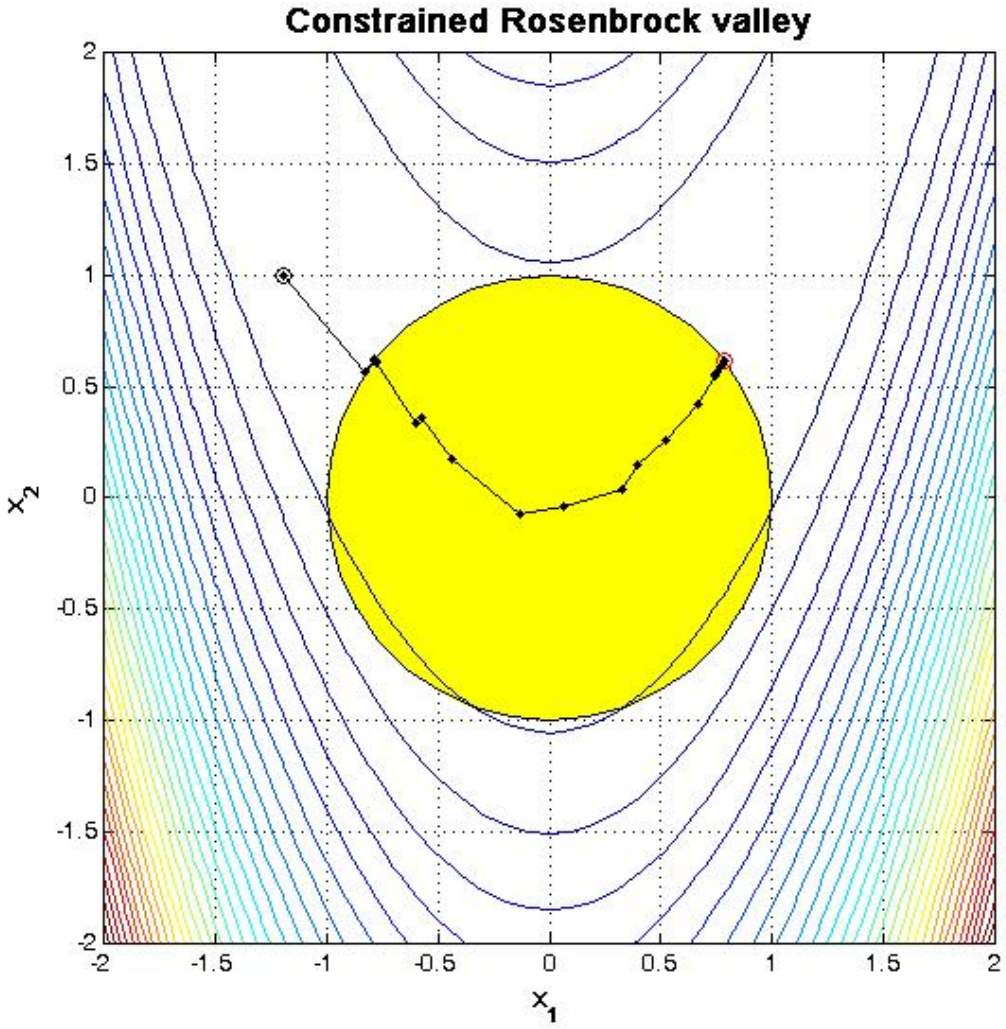
The Levenberg-Marquardt method in Fletcher's modification [1] for solution of nonlinear least squares problems has been implemented in simplified version under the name `LMFsolve` some time ago [2]. The convergence and stability of the function has been strongly influenced both by the simplification of the code and a bug in it. This has been a reason why a new function `LMFnlsq` has been implemented. It is almost unchanged transcription of the original Fletcher's FORTRAN code into MATLAB, but the initial part containing option settings. The new function is stable and efficient.

A script named `LMFtest` is provided for testing `LMFnlsq`. It covers both unconstrained and constrained minimization problem of Rosenbrock's function

$$f(\boldsymbol{x}) = (x_2 - x_1^2)^2 + (1 - x_1)^2 = 0 \tag{1}$$

as a sum of squares of residuals, $f(\boldsymbol{x}) = f_1^2(\boldsymbol{x}) + f_2^2(\boldsymbol{x})$, where $f_1(\boldsymbol{x}) = x_2 - x_1^2$ and $f_2(\boldsymbol{x}) = 1 - x_1$. An additional condition should be stated in case of a **constrained** problem. If the feasible domain were circular with its center at the origin of coordinates, the condition could be formulated as $f_3(\boldsymbol{x}) = 0$ inside the circle, and $f_3(\boldsymbol{x}) = g(d)$, where d is an outer distance of $\boldsymbol{x}$ from the border of the circle. A user may set a radius of a circular feasible domain as well as many other optional parameters. The output of `LMFtest` for the constrained problem with radius R=1 after accepting preset optional values follows:

```
########################## LMFtest   08-Nov-2007 ############################

          radius  = 1 =>
          x_0     = [-1.2; 1] =>
          iprint  = 5 =>
          dist    = lin =>
          weight  = 1000 =>
          ScaleD  = [] =>
          Lambda  = 0 =>
          Trace   = 1 =>

*****************************************************************************
  itr  nfn   SUM(r^2)        x           dx           l           lc
*****************************************************************************
   0    1  3.1592e+005 -1.2000e+000  0.0000e+000  0.0000e+000  1.0000e+000
                        1.0000e+000  0.0000e+000
   5    8  3.1601e+000 -7.7658e-001 -4.1079e-003  5.8606e-006  6.2400e-007
                        6.0931e-001  5.2744e-003
  10   14  1.0317e+000  6.1177e-002 -1.9378e-001  4.2412e-006  6.2400e-007
                       -3.5022e-002 -3.4913e-002
  15   19  6.8997e-002  7.4880e-001 -8.0971e-002  1.0603e-006  6.2400e-007
                        5.5302e-001 -1.2898e-001
  20   26  5.0613e-002  7.7522e-001 -5.5658e-003  1.3254e-005  6.2400e-007
                        6.0004e-001 -8.2279e-003
  25   34  4.5759e-002  7.8624e-001 -1.7343e-004  4.1418e-004  6.2400e-007
                        6.1736e-001 -2.4143e-004
  30   41  4.5680e-002  7.8644e-001 -5.3187e-005  1.2943e-003  6.2400e-007
                        6.1764e-001 -7.9228e-005

  35   49  4.5676e-002  7.8645e-001  3.5255e-007  6.0602e-002  6.2400e-007
                        6.1766e-001 -4.0767e-007

  Distance from the origin R = 1.000000,   R^2 =  1.000000

############################### t = 0.282 sec ################################
```



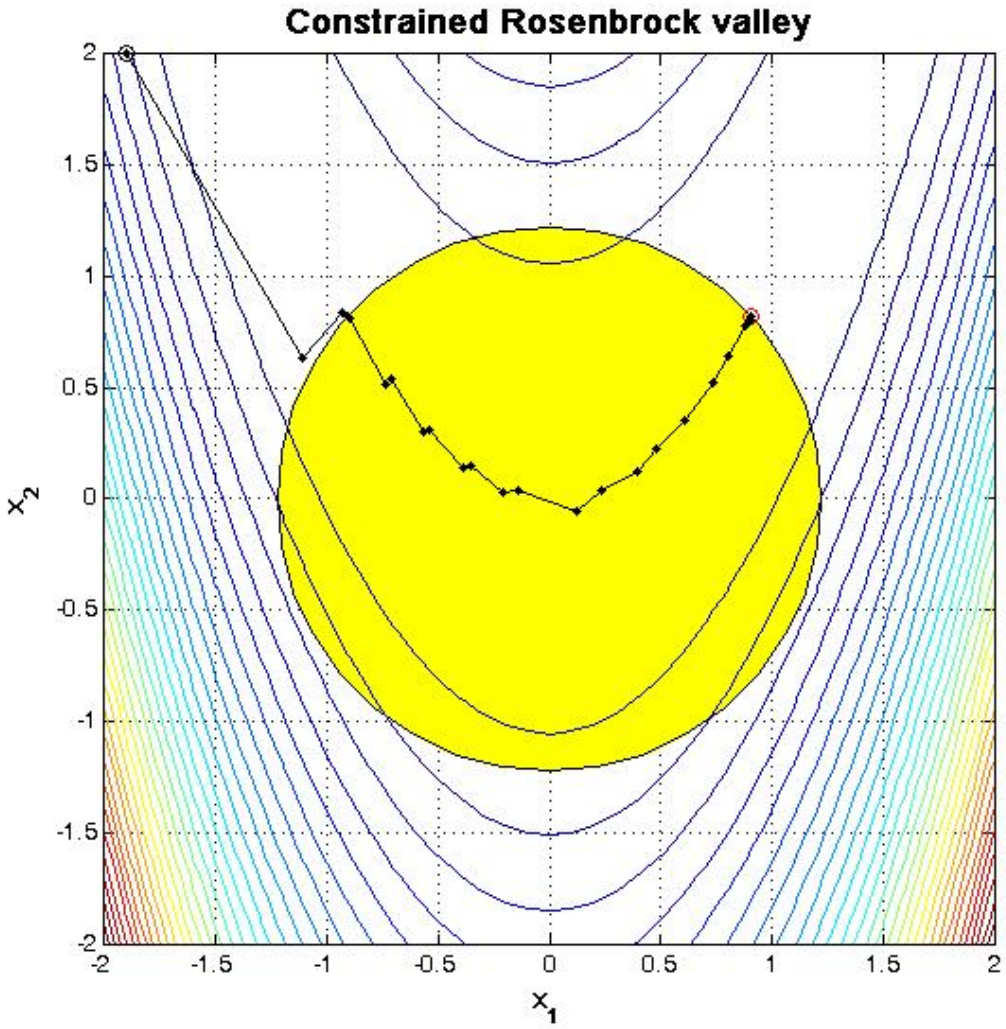
Another example comes from the Optimization Toolbox where the circular constraint is described by an additional inequality

$$x_1^2 + x_2^2 - 1.5 \leq 0\,, \tag{2}$$

which should be transformed for the use of `LMFnlsq` in a single change of input data, where instead default value of radius is given $\sqrt{1.5}$. The final solution corresponds to that from the Optimization Toolbox:

```
########################## LMFtest   13-Nov-2007 ############################

          radius  = 1 => sqrt(1.5)
          x_0     = [-1.2; 1] => [-1.9; 2]
          iprint  = 5 => 10
          dist    = lin =>
          weight  = 1000 =>
          ScaleD  = [] =>
          Lambda  = 0 =>
          Trace   = 1 =>

*************************************************************************
  itr  nfJ   SUM(r^2)        x           dx           l           lc
*************************************************************************
   0    1  2.3530e+006 -1.9000e+000  0.0000e+000  0.0000e+000  1.0000e+000
                       2.0000e+000  0.0000e+000
```

```
  10    15  2.4249e+000 -5.3983e-001 -2.3618e-002  8.0794e-006  4.5394e-007
                        3.1462e-001 -1.1960e-002
  20    27  1.0062e-001  7.3583e-001 -1.2713e-001  1.5532e-006  4.5394e-007
                        5.2389e-001 -1.7122e-001
  30    41  8.7529e-003  9.0653e-001 -5.9104e-004  3.6767e-005  4.7062e-007
                        8.2139e-001 -1.0392e-003
  40    57  8.6170e-003  9.0726e-001 -6.1870e-007  3.5906e-002  4.7062e-007
                        8.2272e-001 -1.0396e-006

  43    61  8.6162e-003  9.0726e-001 -4.9328e-007  8.9764e-002  4.7062e-007
                        8.2273e-001 -8.3338e-007

  Distance from the origin R = 1.224745,    R^2 =  1.500000

############################### t = 0.234 sec ################################
```



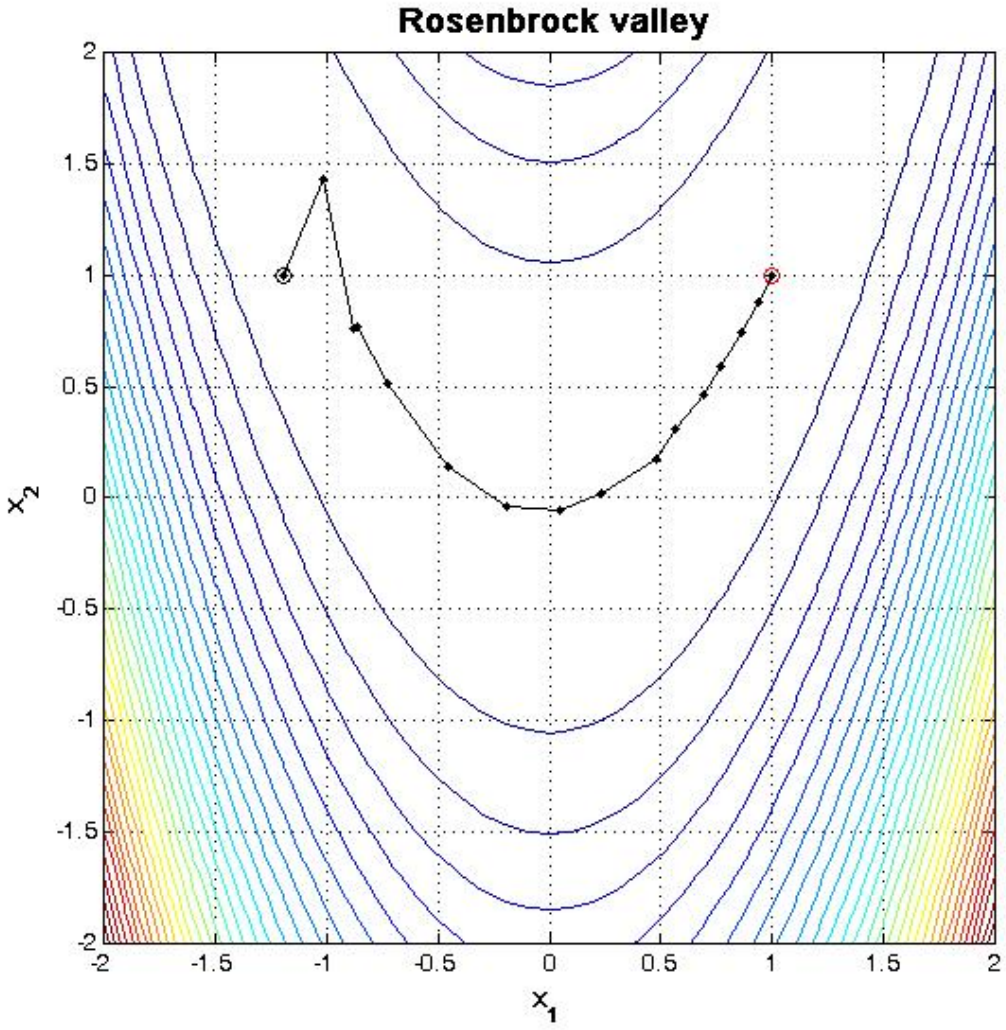
The script `LMFtest` utilizes several functions from the File Exchange collection of functions (FEX):

The function `inp` is employed for operative input of parameters from the keyboard. Usually, it has two arguments, the second one carrying a default value, which may be accepted or changed. It is good for debugging or parametric studies.

The function `fig` creates a figure window on required place of the screen. In case of `LMFtest` it has been in right lower corner.

The function `separator` generates a line created out of arbitrary characters on the screen. The line may be interrupted by a user text. Here, it was applied at beginning and end of the output for displaying the program name, date and elapsed time spent for finding a solution.

The `LMFnlsq` finds also a solution of **unconstrained** Rosenbrock's valey, which is in the point (1; 1):




```
########################## LMFtest   13-Nov-2007 ############################

          radius  = 1 => 0
          x_0     = [-1.2; 1] =>
          iprint  = 5 =>
          ScaleD  = [] =>
          Lambda  = 0 =>
          Trace   = 1 =>

*****************************************************************************
  itr  nfJ   SUM(r^2)        x           dx           l           lc
*****************************************************************************
   0    1  2.4200e+001 -1.2000e+000  0.0000e+000  0.0000e+000  1.0000e+000
                       1.0000e+000  0.0000e+000
   5    8  2.6137e+000 -4.5776e-001 -2.6926e-001  6.9532e-003  8.6655e-004
                       1.3964e-001  3.7337e-001
  10   13  1.9450e-001  5.6651e-001 -8.4431e-002  3.4766e-003  8.6655e-004
                       3.1282e-001 -1.4134e-001
```

```
  15   18  1.1799e-003  1.0000e+000 -5.8609e-002  0.0000e+000  8.6655e-004
                        9.9656e-001 -1.1612e-001

  17   20  0.0000e+000  1.0000e+000 -2.3235e-025  0.0000e+000  8.6655e-004
                        1.0000e+000  5.6217e-012

  Distance from the origin R = 1.414214,   R^2 =  2.000000

############################## t = 0.219 sec ################################
```

Appendix

```
%   LMFtest.m         (Constrained) Rosenbrock's valey
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   The script solves a testing problem of the Rosenbrock's function by
%   minimization of of a sum of squares of residuals.
%   Requirements:                                               FEX ID:
%       inp         function for keyboard input with default value  9033
%       fig         function for coded figure window placement      9035
%       separator   for separating displayed results               11725
%       LMGnlsq     function for nonlinear least squares

clear all close all separator([mfilename,'   ',date],'#',38)

r  = eval(inp('radius ','1'));
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%   r  = 0      no constraint
%      > 0      radius of a circular feasible domain

x0 = eval(inp('x_0    ','[-1.2; 1]'));
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%   Vector of initial point coordinates

ipr= eval(inp('iprint ','5'));
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%   Control variable (step in iterations) for display intermediate results

if r>0                 %   nonzero feasible domain
    dst = inp('dist   ','lin');
%   ~~~~~~~~~~~~~~~~~~~~~~~~~~~
%   form of a penalty function (3rd equation), a distance from the border
%       'lin'   linear
%       'sqr'   quadratic

    switch dst
        case 'lin'
            d = @(x) sqrt(x'*x)-r;    %   A distance from the border
        case 'sqr'
            d = @(x) x(1)^2+x(2)^2-r^2;
        otherwise
            error(['in d = ' dst])
    end

    w  = eval(inp('weight ','1000'));
%   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
%   Weight of the penalty

else                   %   without constraint
    d = @(x) 0;
    w = 0;
end
                %   RESIDUALS:
```

```
ros= @(x) [10*(x(2)-x(1)^2)
           1-x(1)
           (r>0)*(d(x)>0)*d(x)*w
          ];

sd = eval(inp('ScaleD ','[]')); %   D = diag(J'*J)
lm = eval(inp('Lambda ','0'));  %   initial lambda
xy = eval(inp('Trace  ','1'));  %   save intermediate results
disp(' ');

t  = clock;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[xf,ssq,cnt,loops,XY] = LMFnlsq ...
    (ros,x0,'Display',ipr, 'ScaleD',sd, 'Lambda',lm, 'Trace',xy ...
    );
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
R = sqrt(xf'*xf);
fprintf('\n  Distance from the origin R =%9.6f,   R^2 = %9.6f\n', ...
    R, R^2);
separator(['t = ',num2str(etime(clock,t)),' sec'])

if xy
    fig(4);
    plot(-2,-2,2,2)
    axis square
    hold on
    fi=(0:pi/18:2*pi)';
    plot(cos(fi)*r,sin(fi)*r,'r')   %   circle
    grid
    fill(cos(fi)*r,sin(fi)*r,'y')   %   circle = fesible domain
    x=-2:.1:2;
    y=-2:.1:2;
    [X,Y]=meshgrid(x,y);
    Z=100*(Y-X.^2).^2 - (1-X).^2;   %   Rosenbrock's function
    contour(X,Y,Z,30)
    plot(x0(1),x0(2),'ok')          %   starting point
    plot(xf(1),xf(2),'or')          %   terminal point
    plot([x0(1),XY(1,:)],[x0(2),XY(2,:)],'-k.') %   iteration path
    if r>0
        tit = 'Constrained';
    else
        tit = '';
    end
    title([tit,' Rosenbrock valley'],...
        'FontSize',14,'FontWeight','demi')
    xlabel('x_1','FontSize',12,'FontWeight','demi')
    ylabel('x_2','FontSize',12,'FontWeight','demi')
end
```

Reference

[1] Fletcher, R., (1971): A Modified Marquardt Subroutine for Nonlinear Least Squares. Rpt. AERE-R 6799, Harwell

[2] Balda, M.,(2007): LMFsolve: Levenberg-Marquardt-Fletcher's algoritm for non-linear least squares problem. MathWorks, MATLAB Central, File Exchange, http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=16063