Nelder-Mead
User's Manual
– The Fminsearch Function –

Michaël BAUDIN

Version 0.3
September 2009

Contents

Chapter 1

The *fminsearch* function

In this chapter, we analyze the implementation of the *fminsearch* which is provided in Scilab. In the first part, we describe the specific choices of this implementation with respect to the Nelder-Mead algorithm. In the second part, we present some numerical experiments which allows to check that the feature is behaving as expected, by comparison to Matlab's *fminsearch*.

1.1 *fminsearch*'s algorithm

1.1.1 The algorithm

The algorithm used is the Nelder-Mead algorithm. This corresponds to the "variable" value of the "-method" option of the *neldermead*. The "non greedy" version is used, that is, the expansion point is accepted only if it improves over the reflection point.

1.1.2 The initial simplex

The fminsearch algorithm uses a special initial simplex, which is an heuristic depending on the initial guess. The strategy chosen by fminsearch corresponds to the -simplex0method flag of the neldermead component, with the "pfeffer" method. It is associated with the - simplex0deltausual = 0.05 and -simplex0deltazero = 0.0075 parameters. Pfeffer's method is an heuristic which is presented in "Global Optimization Of Lennard-Jones Atomic Clusters" by Ellen Fan [2]. It is due to L. Pfeffer at Stanford. See in the help of optimsimplex for more details.

1.1.3 The number of iterations

In this section, we present the default values for the number of iterations in fminsearch.

The options input argument is an optionnal data structure which can contain the options.MaxIter field. It stores the maximum number of iterations. The default value is 200n, where n is the number of variables. The factor 200 has not been chosen by chance, but is the result of experiments performed against quadratic functions with increasing space dimension.

This result is presented in "Effect of dimensionality on the nelder-mead simplex method" by Lixing Han and Michael Neumann [5]. This paper is based on Lixing Han's PhD, "Algorithms in Unconstrained Optimization" [4]. The study is based on numerical experiment with a quadratic function where the number of terms depends on the dimension of the space (i.e. the number of variables). Their study shows that the number of iterations required to reach the tolerance criteria is roughly 100n. Most iterations are based on inside contractions. Since each step of the Nelder-Mead algorithm only require one or two function evaluations, the number of required function evaluations in this experiment is also roughly 100n.

1.1.4 The termination criteria

The algorithm used by *fminsearch* uses a particular termination criteria, based both on the absolute size of the simplex and the difference of the function values in the simplex. This termination criteria corresponds to the "-tolssizedeltafvmethod" termination criteria of the *neldermead* component.

The size of the simplex is computed with the $\sigma-+$ method, which corresponds to the "sigmaplus" method of the *optimsimplex* component. The tolerance associated with this criteria is given by the "TolX" parameter of the *options* data structure. Its default value is 1.e-4.

The function value difference is the difference between the highest and the lowest function value in the simplex. The tolerance associated with this criteria is given by the "TolFun" parameter of the *options* data structure. Its default value is 1.e-4.

1.2 Numerical experiments

In this section, we analyse the behaviour of Scilab's *fminsearch* function, by comparison of Matlab's *fminsearch*. We especially analyse the results of the optimization, so that we can check that the algorithm is indeed behaving the same way, even if the implementation is completely different.

We consider the unconstrained optimization problem [6]

$$\min f(\mathbf{x}) \tag{1.1}$$

where $\mathbf{x} \in \mathbb{R}^2$ and the objective function f is defined by

$$f(\mathbf{x}) = 100 * (x_2 - x_1^2)^2 + (1 - x_1)^2. \tag{1.2}$$

The initial guess is

$$\mathbf{x}^0 = (-1.2, 1.)^T, \tag{1.3}$$

where the function value is

$$f(\mathbf{x}^0) = 24.2. \tag{1.4}$$

The global solution of this problem is

$$\mathbf{x}^\star = (1, 1.)^T \tag{1.5}$$

where the function value is

$$f(\mathbf{x}^\star) = 0. \tag{1.6}$$

1.2.1 Algorithm and numerical precision

In this section, we are concerned by the comparison of the behavior of the two algorithms. We are going to check that the algorithms produces the same intermediate and final results. We also analyze the numerical precision of the results, by detailing the number of significant digits.

To make a more living presentation of this topic, we will include small scripts which allow to produce the output that we are going to analyze. Because of the similarity of the languages, in order to avoid confusion, we will specify, for each script, the language we use by a small comment. Scripts and outputs written in Matlab's language will begin with

```
% Matlab
% ...
```

while script written in Scilab's language will begin with

```
// Scilab
// ...
```

The following Matlab script allows to see the behaviour of Matlab's *fminsearch* function on Rosenbrock's test case.

```
% Matlab
format long
banana = @(x)100*(x(2)-x(1)^2)^2+(1-x(1))^2;
[x,fval,exitflag,output] = fminsearch(banana,[-1.2, 1])
output.message
```

When this script is launched in Matlab, the following output is produced.

```
>> % Matlab
>> format long
>> banana = @(x)100*(x(2)-x(1)^2)^2+(1-x(1))^2;
>> [x,fval] = fminsearch(banana,[-1.2, 1])
>> [x,fval,exitflag,output] = fminsearch(banana,[-1.2, 1])
x =
   1.000022021783570   1.000042219751772
fval =
     8.177661197416674e-10
exitflag =
     1
output =
    iterations: 85
     funcCount: 159
     algorithm: 'Nelder-Mead simplex direct search'
       message: [1x194 char]
>> output.message
ans =
Optimization terminated:
the current x satisfies the termination criteria using
OPTIONS.TolX of 1.000000e-04
and F(X) satisfies the convergence criteria using
OPTIONS.TolFun of 1.000000e-04
```

The following Scilab script allows to solve the problem with Scilab's *fminsearch*.

```
// Scilab
format(25)
function y = banana (x)
  y = 100*(x(2)-x(1)^2)^2 + (1-x(1))^2;
endfunction
[x , fval , exitflag , output] = fminsearch ( banana , [-1.2 1] )
output.message
```

The output associated with this Scilab script is the following.

```
-->// Scilab
-->format(25)
-->function y = banana (x)
-->  y = 100*(x(2)-x(1)^2)^2 + (1-x(1))^2;
-->endfunction
-->[x , fval , exitflag , output] = fminsearch ( banana , [-1.2 1] )
 output  =
   algorithm: "Nelder-Mead simplex direct search"
   funcCount: 159
   iterations: 85
   message: [3x1 string]
 exitflag  =
    1.
 fval  =
    0.0000000008177661099387
 x  =
    1.0000220217835567027009    1.0000422197517710998227
```

```
-->output.message
 ans  =

!Optimization terminated:                                                        !
!                                                                                !
!the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-004  !
!                                                                                !
!and F(X) satisfies the convergence criteria using OPTIONS.TolFun of 1.000000e-004     !
```

Because the two softwares do not use the same formatting rules to produce their outputs, we must perform additionnal checking in order to check our results.

The following Scilab script displays the results with 16 significant digits.

```
// Scilab
// Print the result with 15 significant digits
mprintf ( "%.15e" , fval );
mprintf ( "%.15e %.15e" , x(1) , x(2) );
```

The previous script produces the following output.

```
-->// Scilab
-->mprintf ( "%.15e" , fval );
8.177661099387146e-010
-->mprintf ( "%.15e %.15e" , x(1) , x(2) );
1.000022021783557e+000 1.000042219751771e+000
```

These results are reproduced verbatim in the table 1.1.

Matlab Iterations Scilab Iterations	85 85	
Matlab Function Evaluations Scilab Function Evaluations	159 159	
Matlab $\mathbf{x}^\star$ Scilab $\mathbf{x}^\star$	1.000022021783570 1.000022021783557e+000	1.000042219751772 1.000042219751771e+000
Matlab $f(\mathbf{x}^\star)$ Scilab $f(\mathbf{x}^\star)$	8.177661197416674e-10 8.177661099387146e-010	

Fig. 1.1 : Numerical experiment with Rosenbrock's function – Comparison of results produced by Matlab and Scilab.

We must compute the common number of significant digits in order to check the consistency of the results. The following Scilab script computes the relative error between Scilab and Matlab results.

```
// Scilab
// Compare the result
xmb = [1.000022021783570    1.000042219751772 ];
err = norm(x - xmb) / norm(xmb);
mprintf ( "Relative Error on x : %e\n", err );
fmb = 8.177661197416674e-10;
err = abs(fval - fmb) / abs(fmb);
mprintf ( "Relative Error on f : %e\n", err );
```

The previous script produces the following output.

```
// Scilab
Relative Error on x : 9.441163e-015
Relative Error on f : 1.198748e-008
```

We must take into account for the floating point implementations of both Matlab and Scilab. In both these numerical softwares, double precision floating point numbers are used, i.e. the relative precision is both these softwares is $\epsilon \approx 10^{-16}$. That implies that there are approximately 16 significant digits. Therefore, the relative error on x, which is equivalent to 15 significant digits, is acceptable.

Therefore, the result is as close as possible to the result produced by Matlab. More specifically :

- the optimum x is the same up to 15 significant digits,
- the function value at optimum is the same up to 8 significant digits,
- the number of iterations is the same,
- the number of function evaluations is the same,
- the exit flag is the same,
- the content of the output is the same (but the string is not display the same way).

The output of the two functions is the same. We must now check that the algorithms performs the same way, that is, produces the same intermediate steps.

The following Matlab script allows to get deeper information by printing a message at each iteration with the "Display" option.

```
% Matlab
opt = optimset('Display','iter');
[x,fval,exitflag,output] = fminsearch(banana,[-1.2, 1] , opt );
```

The previous script produces the following output.

```
% Matlab
 Iteration   Func-count     min f(x)         Procedure
     0            1           24.2
     1            3          20.05         initial simplex
     2            5         5.1618         expand
     3            7         4.4978         reflect
     4            9         4.4978         contract outside
     5           11        4.38136         contract inside
     6           13        4.24527         contract inside
     7           15        4.21762         reflect
     8           17        4.21129         contract inside
     9           19        4.13556         expand
    10           21        4.13556         contract inside
    11           23        4.01273         expand
    12           25        3.93738         expand
    13           27        3.60261         expand
    14           28        3.60261         reflect
    15           30        3.46622         reflect
    16           32        3.21605         expand
    17           34        3.16491         reflect
    18           36        2.70687         expand
    19           37        2.70687         reflect
    20           39        2.00218         expand
    21           41        2.00218         contract inside
    22           43        2.00218         contract inside
    23           45        1.81543         expand
    24           47        1.73481         contract outside
    25           49        1.31697         expand
    26           50        1.31697         reflect
    27           51        1.31697         reflect
    28           53         1.1595         reflect
    29           55        1.07674         contract inside
    30           57       0.883492         reflect
    31           59       0.883492         contract inside
    32           61       0.669165         expand
    33           63       0.669165         contract inside
    34           64       0.669165         reflect
    35           66       0.536729         reflect
    36           68       0.536729         contract inside
    37           70       0.423294         expand
    38           72       0.423294         contract outside
    39           74       0.398527         reflect
    40           76        0.31447         expand
    41           77        0.31447         reflect
    42           79       0.190317         expand
    43           81       0.190317         contract inside
    44           82       0.190317         reflect
    45           84        0.13696         reflect
    46           86        0.13696         contract outside
    47           88       0.113128         contract outside
    48           90        0.11053         contract inside
    49           92        0.10234         reflect
    50           94       0.101184         contract inside
    51           96      0.0794969         expand
    52           97      0.0794969         reflect
    53           98      0.0794969         reflect
    54          100      0.0569294         expand
    55          102      0.0569294         contract inside
    56          104      0.0344855         expand
```

```
     57        106        0.0179534        expand
     58        108        0.0169469        contract outside
     59        110       0.00401463        reflect
     60        112       0.00401463        contract inside
     61        113       0.00401463        reflect
     62        115      0.000369954        reflect
     63        117      0.000369954        contract inside
     64        118      0.000369954        reflect
     65        120      0.000369954        contract inside
     66        122     5.90111e-005        contract outside
     67        124     3.36682e-005        contract inside
     68        126     3.36682e-005        contract outside
     69        128     1.89159e-005        contract outside
     70        130     8.46083e-006        contract inside
     71        132     2.88255e-006        contract inside
     72        133     2.88255e-006        reflect
     73        135     7.48997e-007        contract inside
     74        137     7.48997e-007        contract inside
     75        139     6.20365e-007        contract inside
     76        141     2.16919e-007        contract outside
     77        143     1.00244e-007        contract inside
     78        145     5.23487e-008        contract inside
     79        147     5.03503e-008        contract inside
     80        149      2.0043e-008        contract inside
     81        151     1.12293e-009        contract inside
     82        153     1.12293e-009        contract outside
     83        155     1.12293e-009        contract inside
     84        157     1.10755e-009        contract outside
     85        159     8.17766e-010        contract inside

Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-004
 and F(X) satisfies the convergence criteria using OPTIONS.TolFun of 1.000000e-004
```

The following Scilab script set the "Display" option to "iter" and run the *fminsearch* function.

```
// Scilab
opt = optimset ( "Display" , "iter" );
[x , fval , exitflag , output] = fminsearch ( banana , [-1.2 1] , opt );
```

```
// Scilab
 Iteration   Func-count     min f(x)         Procedure
     0            3           24.2
     1            3          20.05           initial simplex
     2            5       5.161796           expand
     3            7       4.497796           reflect
     4            9       4.497796           contract outside
     5           11      4.3813601           contract inside
     6           13      4.2452728           contract inside
     7           15      4.2176247           reflect
     8           17      4.2112906           contract inside
     9           19      4.1355598           expand
    10           21      4.1355598           contract inside
    11           23      4.0127268           expand
    12           25      3.9373812           expand
    13           27       3.602606           expand
    14           28       3.602606           reflect
    15           30      3.4662211           reflect
    16           32      3.2160547           expand
    17           34      3.1649126           reflect
    18           36      2.7068692           expand
    19           37      2.7068692           reflect
    20           39      2.0021824           expand
    21           41      2.0021824           contract inside
    22           43      2.0021824           contract inside
    23           45      1.8154337           expand
    24           47      1.7348144           contract outside
    25           49      1.3169723           expand
    26           50      1.3169723           reflect
    27           51      1.3169723           reflect
    28           53      1.1595038           reflect
    29           55      1.0767387           contract inside
    30           57      0.8834921           reflect
    31           59      0.8834921           contract inside
    32           61      0.6691654           expand
    33           63      0.6691654           contract inside
    34           64      0.6691654           reflect
    35           66      0.5367289           reflect
    36           68      0.5367289           contract inside
    37           70      0.4232940           expand
    38           72      0.4232940           contract outside
    39           74      0.3985272           reflect
    40           76      0.3144704           expand
    41           77      0.3144704           reflect
    42           79      0.1903167           expand
    43           81      0.1903167           contract inside
    44           82      0.1903167           reflect
    45           84      0.1369602           reflect
    46           86      0.1369602           contract outside
    47           88      0.1131281           contract outside
    48           90      0.1105304           contract inside
    49           92      0.1023402           reflect
    50           94      0.1011837           contract inside
    51           96      0.0794969           expand
```

```
52      97    0.0794969    reflect
53      98    0.0794969    reflect
54     100    0.0569294    expand
55     102    0.0569294    contract inside
56     104    0.0344855    expand
57     106    0.0179534    expand
58     108    0.0169469    contract outside
59     110    0.0040146    reflect
60     112    0.0040146    contract inside
61     113    0.0040146    reflect
62     115    0.0003700    reflect
63     117    0.0003700    contract inside
64     118    0.0003700    reflect
65     120    0.0003700    contract inside
66     122    0.0000590    contract outside
67     124    0.0000337    contract inside
68     126    0.0000337    contract outside
69     128    0.0000189    contract outside
70     130    0.0000085    contract inside
71     132    0.0000029    contract inside
72     133    0.0000029    reflect
73     135    0.0000007    contract inside
74     137    0.0000007    contract inside
75     139    0.0000006    contract inside
76     141    0.0000002    contract outside
77     143    0.0000001    contract inside
78     145    5.235D-08    contract inside
79     147    5.035D-08    contract inside
80     149    2.004D-08    contract inside
81     151    1.123D-09    contract inside
82     153    1.123D-09    contract outside
83     155    1.123D-09    contract inside
84     157    1.108D-09    contract outside
85     159    8.178D-10    contract inside

Optimization terminated:
 the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-004
 and F(X) satisfies the convergence criteria using OPTIONS.TolFun of 1.000000e-004
```

A close inspection at the data reveals that the two softwares produces indeed the same intermediate results.

1.2.2 Plot functions

In this section, we check that the plotting features of the *fminsearch* function are the same.

The following output function plots in the current graphic window the value of the current parameter **x**. To let Matlab load that script, save the content in a .m file, in a directory known by Matlab.

```
% Matlab
function stop = outfun(x, optimValues, state)
stop = false;
hold on;
plot(x(1),x(2),'.');
drawnow
```

The following Matlab script allows to perform the optimization so that the output function is called back at each iteration.

```
% Matlab
options = optimset('OutputFcn', @outfun);
[x fval] = fminsearch(banana, [-1.2, 1], options)
```

This produces the plot which is presented in figure 1.2.

The following Scilab script sets the "OutputFcn" option and then calls the *fminsearch* in order to perform the optimization.

```
// Scilab
function outfun ( x , optimValues , state )
  plot( x(1),x(2),'.');
endfunction
opt = optimset ( "OutputFcn" , outfun);
[x fval] = fminsearch ( banana , [-1.2 1] , opt );
```

The previous script produces the plot which is presented in figure 1.3.

Except for the size of the dots (which can be configured in both softwares), the graphics are exactly the same.

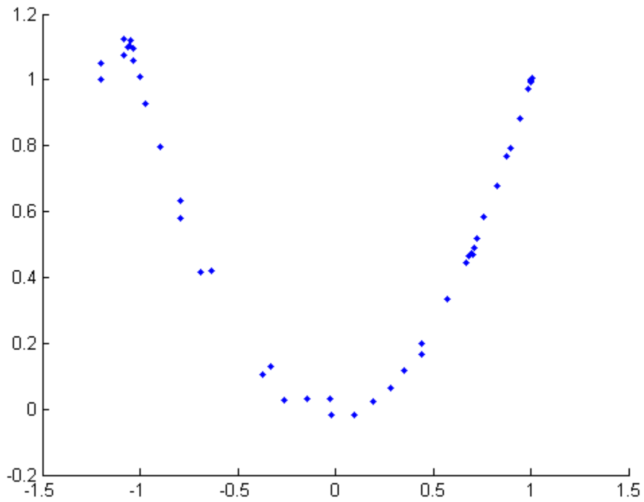

Fig. 1.2 : Plot produced by Matlab's fminsearch, with customized output function.

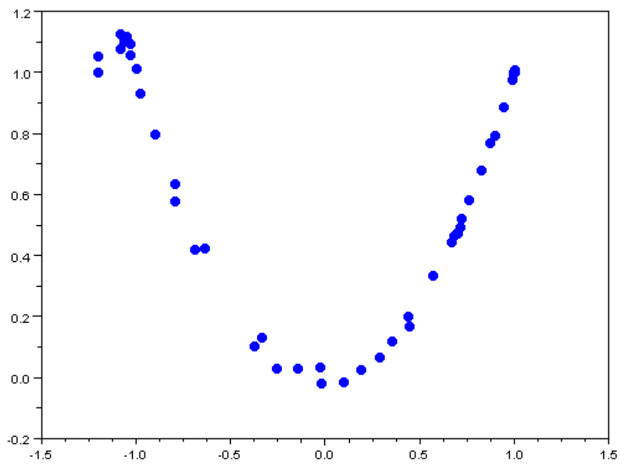

Fig. 1.3 : Plot produced by Scilab's fminsearch, with customized output function.

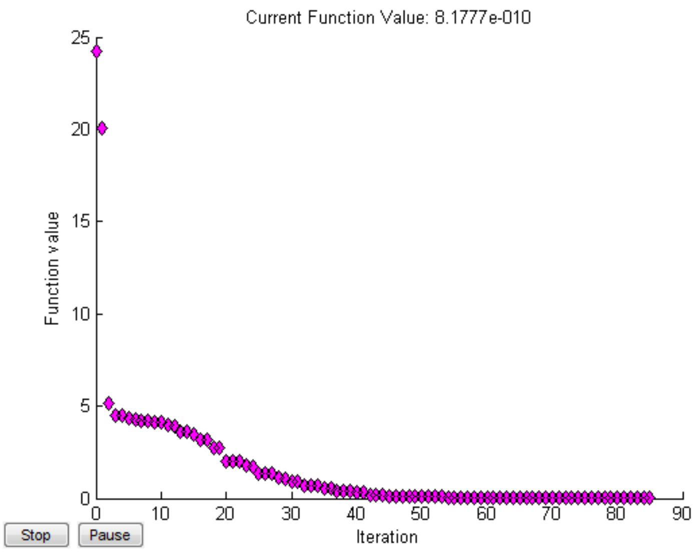

Fig. 1.4 : Plot produced by Matlab's fminsearch, with the *optimplotfval* function.

1.2.3 Predefined plot functions

Several pre-defined plot functions are provided with the *fminsearch* function. These functions are

- *optimplotfval*,
- *optimplotx*,
- *optimplotfunccount*.

In the following Matlab script, we use the *optimplotfval* pre-defined function.

```
% Matlab
options = optimset('PlotFcns',@optimplotfval);
[x fval] = fminsearch(banana , [-1.2, 1] , options)
```

The previous script produces the plot which is presented in figure 1.4.

The following Scilab script uses the *optimplotfval* pre-defined function.

```
// Scilab
opt = optimset ( "OutputFcn" , optimplotfval );
[x fval] = fminsearch ( banana , [-1.2 1] , opt );
```

The previous script produces the plot which is presented in figure 1.5.

The comparison between the figures 1.4 and 1.5 shows that the two features produce very similar plots. Notice that Scilab's *fminsearch* does not provide the "Stop" and "Pause" buttons.

The figures 1.6 and 1.7 present the results of Scilab's *optimplotx* and *optimplotfunccount* functions.

1.3 Conclusion

The current version of Scilab's *fminsearch* provides the same algorithm as Matlab's *fminsearch*. The numerical precision is the same. The *optimset* and *optimget* functions allows to configure the optimization, as well as the output and plotting function. Pre-defined plotting function allows to get a fast and nice plot of the optimization.

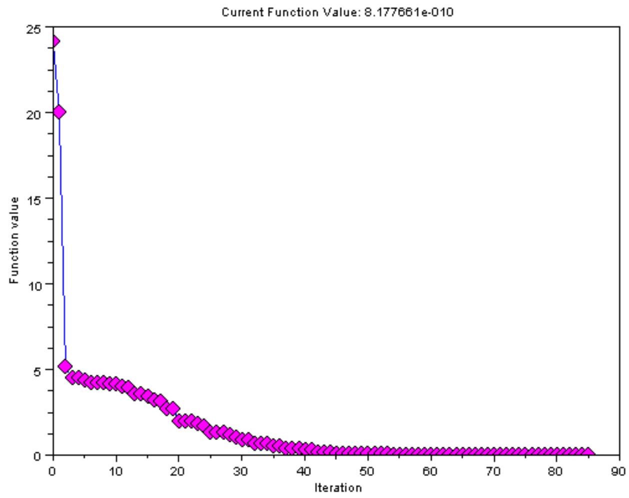

Fig. 1.5 : Plot produced by Scilab's fminsearch, with the *optimplotfval* function.

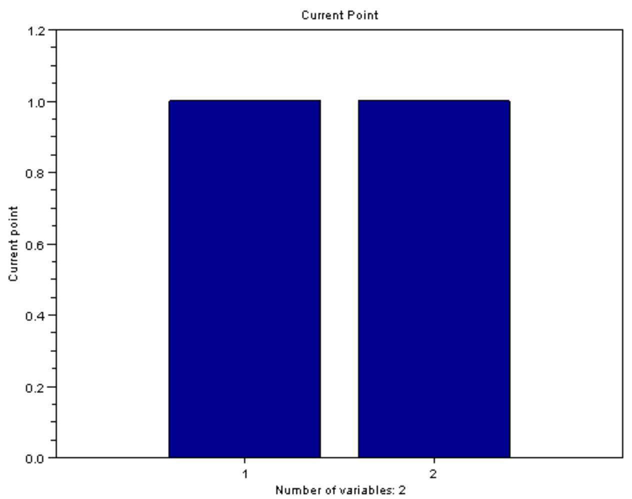

Fig. 1.6 : Plot produced by Scilab's fminsearch, with the *optimplotx* function.

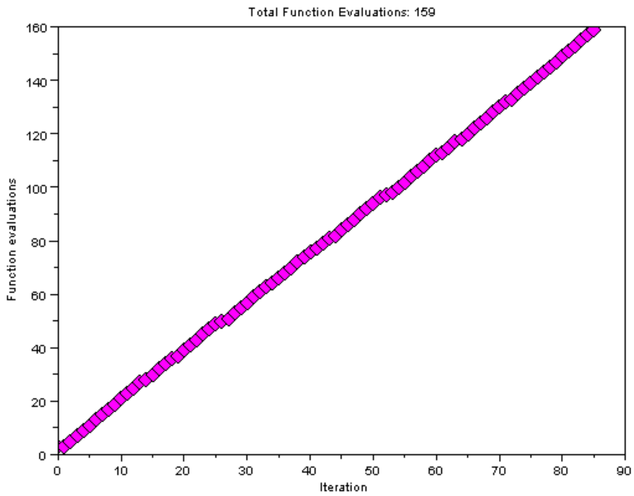

Fig. 1.7 : Plot produced by Scilab's fminsearch, with the *optimplotfunccount* function.

Bibliography

[1] Richard P. Brent. Algorithms for minimization without derivatives, 1973.

[2] Ellen Fan. Global optimization of lennard-jones atomic clusters. Technical report, McMaster University, February 2002.

[3] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization*. Academic Press, London, 1981.

[4] Lixing Han. *Algorithms in Unconstrained Optimization*. Ph.D., The University of Connecticut, 2000.

[5] Lixing Han and Michael Neumann. Effect of dimensionality on the nelder-mead simplex method. *Optimization Methods and Software*, 21(1):1–16, 2006.

[6] H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, March 1960.