Optimization

Hakon Sandaker,Vincent Wilmet,Sauraj Verma

March 2021

1 Problem 5

1.1 Calculate the gradient and the Hessian of the objective function.

The gradient ∇f is computed as:

$$f(x) = x^2(x^2 - 4) = x^4 - 4x^2$$
$$f'(x) = 4x^3 - 8x$$

Since we only have to estimate x, we get the Hessian:

$$f''(x) = 12x^2 - 8$$

1.2 Is the objective function convex? strictly convex ?

No, f"(x) can be less than zero in some cases. Example is when $x \in [0, 1]$, then f"(0) = -8. Thus, its not strictly convex either.

1.3 Determine the critical point(s) of this objective function.

$$f'(x) = 0$$

Yields the solutions $x = -\sqrt{2}$, x=0, $x = \sqrt{2}$.

1.4 Does the problem admit a solution? If so, calculate the optimal solutions.

$$f(-\sqrt{2}) = -4$$
$$f(0) = 0$$
$$f(-\sqrt{2}) = -4$$

Hence, $x = -\sqrt{2}$ and $x = \sqrt{2}$ admits the solutions.

1.5 Analyze this plot and comment it (regarding the obtained previous results).

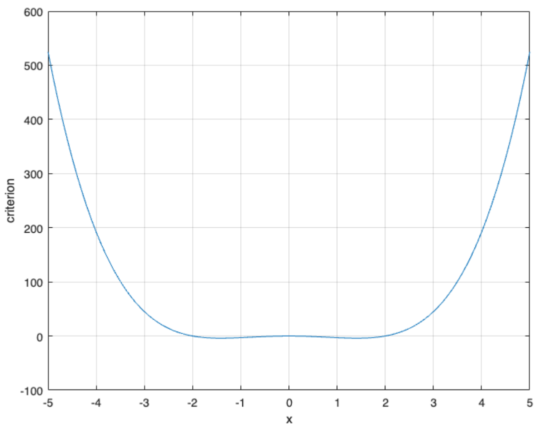

Figure 1: Our Special Plot

1.6 Solve the problem using Nelder and Mead simplex and determine the value of optimal cost.

```
% Clear
clear all;
close all;
clc;

% Define the region bounds
x = [-5:0.1:5]';
f = x.^4 - 4*x.^2;

% Plot
plot(x,f);
grid, xlabel('x');
ylabel('criterion'); % plot

% Nelder and Mead method
f_2 = @(x) x.^4 - 4*x.^2;

options = optimset('fminsearch');
options = optimset(options, 'Display', 'iter');
options = optimset(options, 'TolX', 0.000001, 'TolFun', 0.000001, 'MaxIter', 1000, 'MaxFunEvals',
    1000);

v = 4;
[a1, fv1] = fminsearch(f_2, v, options);
% a1 = 1.4142 & fv1 = -4

v2 = -4;
[a2, fv2] = fminsearch(f_2, v2, options);
% a2 = -1.4142 & fv2 = -4
```

2 Problem 6

2.1 Plot the function and its contour. How many minimizers does the problem has?

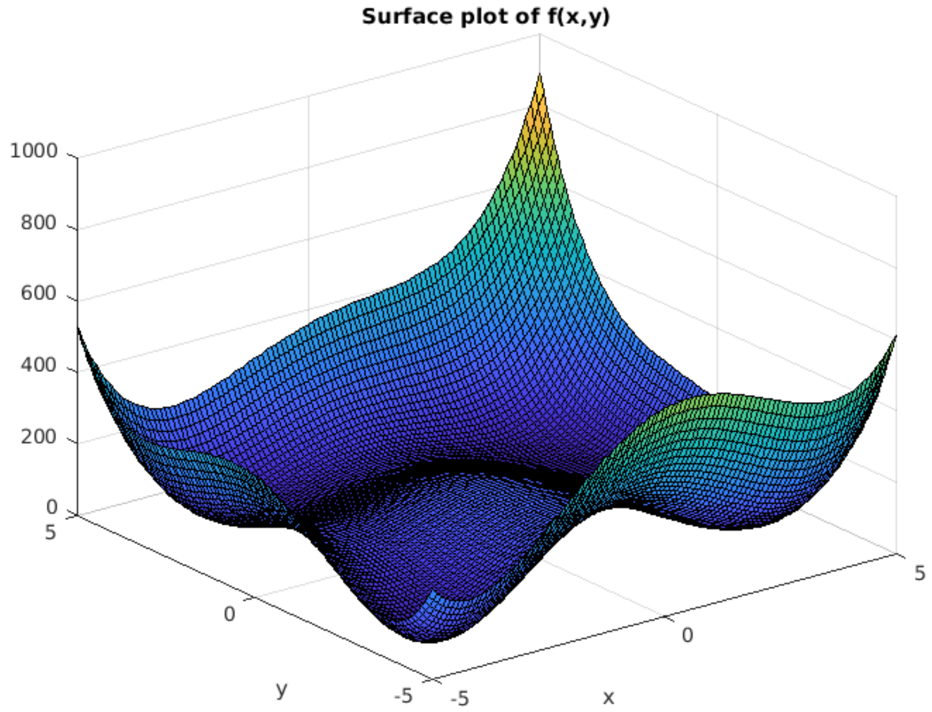

Figure 2: Surface Plot

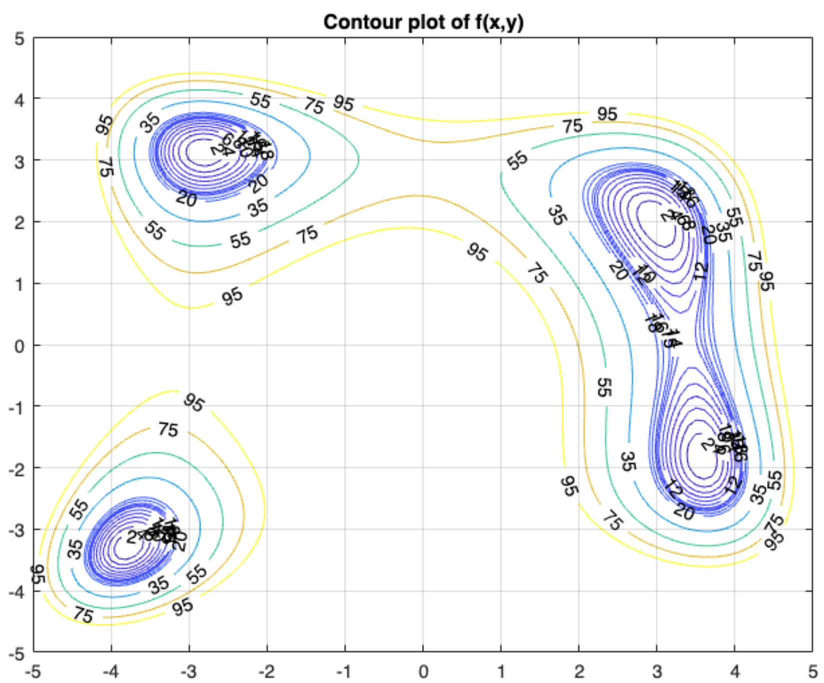

Figure 3: Contour Plot

We can see that the function has 4 minimum points: (-3.779, -3.283), (-2.805, 3.131), (3, 1.999), (3.584, -1.848).

2.2 From this plot, is the function convex?

Figure 3 shows that this function is not convex as it has multiple critical points, but can be convex is we zoom to a local window.

2.3 Using Matlab, determine the optimal solution(s).

```matlab
clear all;
close all;
clc;

%-------------------------------------------
% Plotting the function
%-------------------------------------------
x=-5:.1:5;
y=-5:.1:5;
[xx,yy] = meshgrid(x,y);

zz=(xx.^2 + yy - 11).^2 + (xx + yy.^2 - 7).^2;

figure(1);
surf(xx,yy,zz);  xlabel('x'); ylabel('y');
title('Surface plot of f(x,y)');

figure(2);
levels = [-20:2:20, 15:20:100];
contour(xx,yy,zz,levels,'ShowText','on'); grid;
title('Contour plot of f(x,y)');

figure(3);
levels = [0:0.1:1];
contour(xx,yy,zz,levels,'ShowText','off');


%-------------------------------------------
% Minimization of the function
%-------------------------------------------
% Gradient-free methods : Lender & Mead
%-------------------------------------------
fprintf('Lender & Mead');
Theta_initial = [3.5;2]; % init points
options = optimset('fminsearch');
options = optimset(options, 'Display', 'iter', 'MaxIter', 100000, 'TolFun', 1e-5, 'TolX', 1e-3);
Theta_opt = fminsearch(@Cost_Rosenbrock, Theta_initial, options);
Theta_opt = fminsearch(@Cost_Rosenbrock, Theta_opt, options);
disp(Theta_opt)

%-------------------------------------------
% Gradient-based methods
%-------------------------------------------
Theta_initial = [-4;-4];
options = optimset('fminunc');
options = optimset(options, 'Display', 'iter', 'MaxIter', 10000, 'TolFun', 1e-10, 'TolX', 1e-10);

%-------------------------------------------
% BFGS
%-------------------------------------------
fprintf('BFGS');
tic
options.Algorithm = 'quasi-newton';
options.HessUpdate = 'bfgs';
options.GradObj = 'off';
Theta_opt = fminunc(@Cost_Rosenbrock,Theta_initial, options );
toc
Theta_opt = fminunc(@Cost_Rosenbrock,Theta_opt, options );
disp(Theta_opt)

%-------------------------------------------
% DFP
%-------------------------------------------
```

```matlab
fprintf('DFP');
tic
options.Algorithm = 'quasi-newton';
options.HessUpdate = 'DFP';
options.GradObj = 'off';
Theta_opt = fminunc(@Cost_Rosenbrock,Theta_initial, options );
toc
Theta_opt = fminunc(@Cost_Rosenbrock,Theta_opt, options );
disp(Theta_opt)

%------------------------------------------------
% Steepest descent
%------------------------------------------------
fprintf('Steepest Descent');
tic
options.Algorithm = 'quasi-newton';
options.HessUpdate = 'steepdesc';
options.GradObj = 'off';
Theta_opt = fminunc(@Cost_Rosenbrock, Theta_initial, options);
toc
Theta_opt = fminunc(@Cost_Rosenbrock,Theta_opt, options );
disp(Theta_opt)
```

2.3.1 Compare performances of several optimization methods (Nelder and Mead, BFGS, DFP, steepest descent).

Lender & Mead

data.txt

```
 Iteration   Func-count     min fx         Procedure
     0            1         10512.5
     1            3         10308.5         initial simplex
     2            5          6045.8         expand
     3            7         4197.27         expand
     4            9         484.379         expand
     5           11         112.416         expand
     6           13         112.416         contract outside
     7           15            6.99         contract inside
     8           16            6.99         reflect
     9           18            6.99         contract inside
    10           20            6.99         contract outside
    11           22         6.19346         contract inside
    12           24         2.23013         contract inside
    13           26        0.601036         contract inside
    14           28        0.601036         contract outside
    15           30        0.601036         contract inside
    16           32        0.542317         contract inside
    17           34        0.512786         contract inside
    18           36        0.512786         contract outside
    19           38         0.50625         contract inside
    20           40        0.501867         reflect
    21           42        0.501867         contract inside
    22           44        0.500625         reflect
    23           46        0.497449         expand
    24           48        0.492813         expand
    25           50        0.484288         expand
    26           52        0.468242         expand
    27           54        0.449721         expand
    28           56        0.391355         expand
    29           57        0.391355         reflect
    30           59        0.275217         expand
    31           60        0.275217         reflect
    32           62        0.202371         reflect
    33           64        0.202371         contract inside
    34           66        0.141605         expand
    35           68        0.141605         contract inside
    36           69        0.141605         reflect
    37           70        0.141605         reflect
    38           72        0.084271         expand
    39           74        0.084271         contract inside
    40           76       0.0832419         reflect
    41           78       0.0419705         expand
    42           80       0.0419705         contract inside
    43           82       0.0272179         expand
```

44	83	0.0272179	reflect
45	85	0.000863056	expand
46	87	0.000863056	contract inside
47	89	0.000863056	contract inside
48	91	0.000863056	contract inside
49	92	0.000863056	reflect
50	94	0.000863056	contract inside
51	95	0.000863056	reflect
52	97	0.000128861	contract inside
53	99	0.000128861	contract inside
54	101	0.000128861	contract inside
55	103	4.33006e-05	contract inside
56	105	1.81408e-05	contract inside
57	107	1.18579e-05	contract inside
58	109	1.01877e-05	contract inside
59	111	3.25322e-06	contract inside
60	113	5.66168e-07	contract inside
61	115	5.66168e-07	contract outside
62	117	4.2039e-07	contract inside
63	119	1.46743e-07	contract inside
64	121	5.40924e-08	contract inside
65	123	5.40924e-08	contract inside

BFGS

data.txt

Iteration	Func-count	fx	Step-size	optimality
0	3	40025		3.2e+04
1	6	16592.7	3.12402e-05	1.55e+04
2	9	6312.48	1	6.56e+03
3	12	2852.33	1	2.91e+03
4	15	1504.79	1	1.2e+03
5	18	1005.15	1	634
6	21	839.831	1	579
7	24	787.603	1	561
8	27	738.358	1	543
9	30	614.381	1	581
10	33	370.916	1	662
11	36	79.1993	1	390
12	39	1.69639	1	61.2
13	42	0.0371258	1	2.05
14	45	0.0358836	1	0.453
15	51	0.035012	10	0.325
16	54	0.0324287	1	1.09
17	57	0.0157532	1	1.69
18	66	0.00380661	0.138153	1.5
19	69	0.00317321	1	1.18

DFP

data.txt

Iteration	Func-count	fx	Step-size	optimality
0	3	40025		3.2e+04
1	6	16592.7	3.12402e-05	1.55e+04
2	9	6312.59	1	6.56e+03
3	12	2852.67	1	2.91e+03
4	15	1505.5	1	1.2e+03
5	18	1006.52	1	634
6	21	842.656	1	580
7	24	794.252	1	563
8	27	762.515	1	552
9	30	725.895	1	549
10	33	693.922	1	674
11	36	658.042	1	815
12	39	625.633	1	930
13	42	590.331	1	1.05e+03
14	45	557.568	1	1.16e+03
15	48	522.73	1	1.26e+03

```
16          51         489.698             1        1.35e+03
17          54         455.239             1        1.44e+03
18          57         422.023             1        1.51e+03
19          60         387.887             1        1.58e+03
```

Steepest Descent

data.txt

```
                                                    First-order
Iteration  Func-count       fx          Step-size       optimality
     0           3       0.0490781                          0.174
     1          15        0.049033     0.00212104           0.244
     2          27        0.048988      0.0010831           0.174
     3          39        0.048943     0.00212171           0.244
     4          51        0.048898     0.00108339           0.174
     5          63        0.048853     0.00212238           0.244
     6          75       0.0488081     0.00108368           0.174
     7          87       0.0487632     0.00212305           0.243
     8          99       0.0487183     0.00108397           0.174
     9         111       0.0486734     0.00212372           0.243
    10         123       0.0486286     0.00108426           0.174
    11         135       0.0485838     0.00212439           0.243
    12         147        0.048539     0.00108455           0.174
    13         159       0.0484943     0.00212506           0.243
    14         171       0.0484496     0.00108484           0.174
    15         183       0.0484049     0.00212573           0.243
    16         195       0.0483603     0.00108513           0.173
```

Timing table:

Algorithm	**Time (seconds)**
Lender and Mead	0.132839
BFGS	0.342803
DFP	0.044757
Steepest Descent	0.040857

We can see that BFGS is the slowest algorithm, especially given that it runs around the same amount as iterations as DFP and Steepest Descent.

2.3.2 Consider several initialization.

We tried with several different initialization points, and was able to adjust the number of iterations required, by having a close point.

3 Appendix

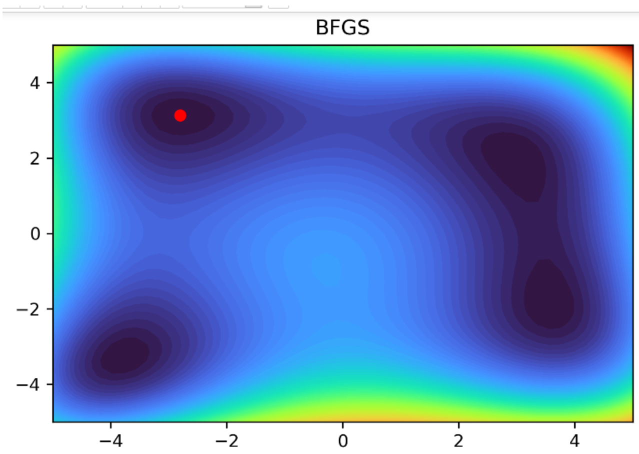

Figure 4:

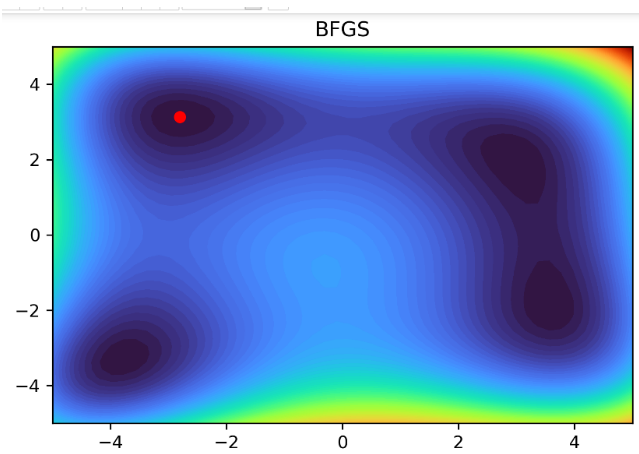

Figure 5:

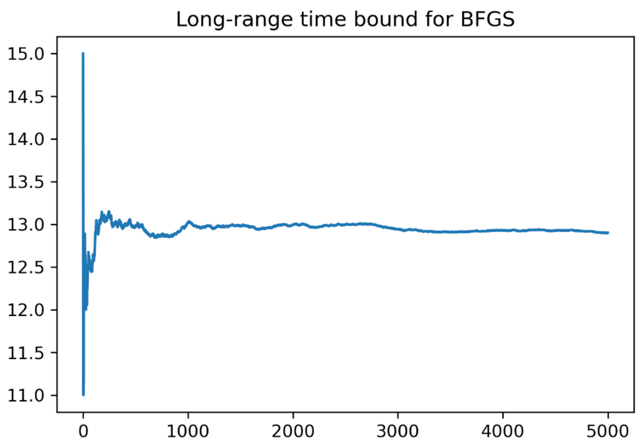

Figure 6:

```
Out[340]:  final_simplex: (array([[-3.77933518, -3.28316778],
       [-3.7793292 , -3.28326484],
       [-3.77924691, -3.28310557]]), array([6.35953394e-08, 2.52954517e-07, 3.74673628e-07]))
           fun: 6.359533941641861e-08
       message: 'Optimization terminated successfully.'
          nfev: 67
           nit: 34
        status: 0
       success: True
             x: array([-3.77933518, -3.28316778])
```

Figure 7: