[1] $f(x) = 100(x_2 - x_1^2)^2 + (1-x_1)^2$

$\alpha_0 = 1$

ⓐ Notice that Newton converges much faster, usually has step length 1, while steepest descent has a much smaller step length.

The condition in the loop is the sufficient descent condition.

With $\varepsilon = 0{,}001$ Newton has 22 iterations, while steepest descent has 1082!

ⓑ BFGS has 35 iterations, which is very close to Newton!

ⓒ We observe that BFGS alternates between 0,5 and 1 mostly, while Newton "converges" to 1 after some time.
3.3 notes that $\alpha_k = 1$ for large k for Newton - this fits with the observations!

[2] It is desirable when we cannot guarantee a nonsingular Hessian, which in practice is usual. By modifying the search direction we make the algorithm more robust.

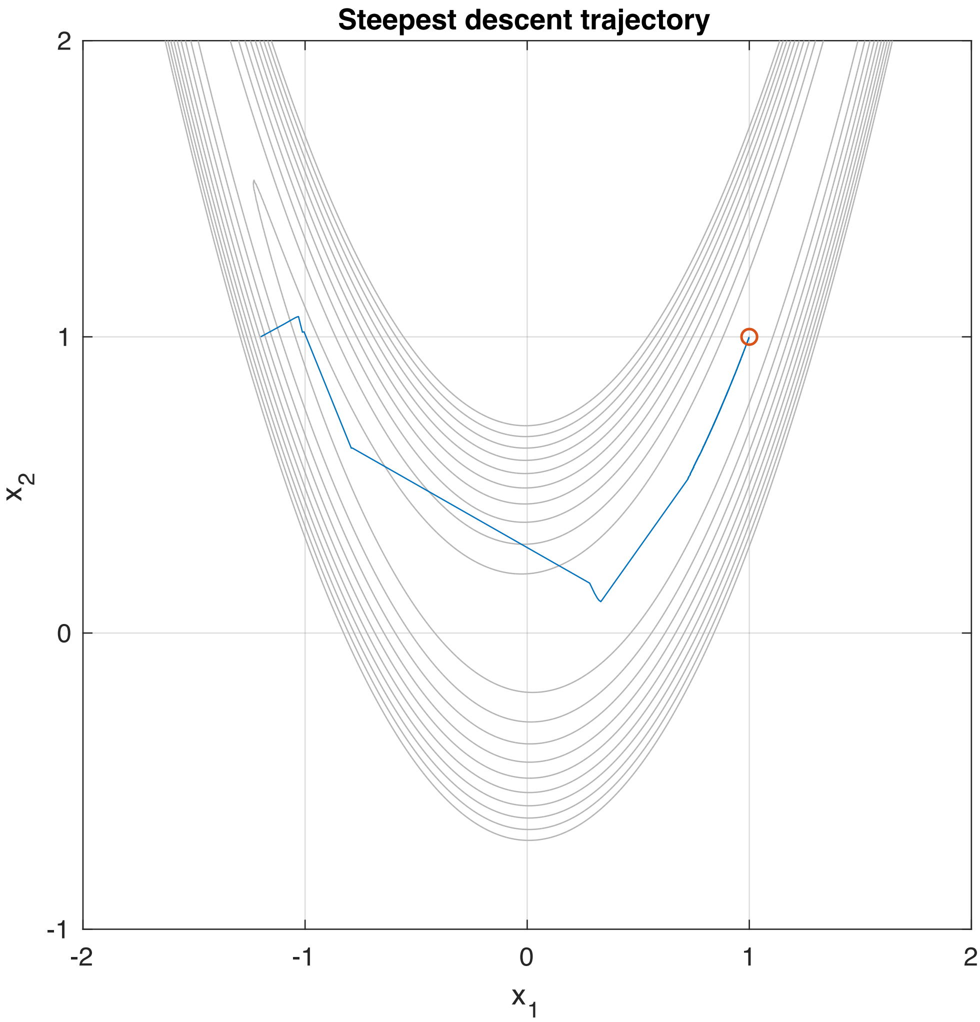
Scanned by CamScanner

```
rho = 0.5;
c = 0.5;
f = @(x) 100*(x(2)-x(1)^2)^2+(1-x(1))^2;
f_grad = @(x) [-400*(x(2)-x(1)^2)*x(1) - 2*(1-x(1)); 200*(x(2)-
x(1)^2)];
f_hessian = @(x) [(1200*x(1)^2 - 400*x(2) + 2) -400*x(1); -400*x(1)
 200];

H_k = eye(2);
epsilon = 0.001;
x_k = [-1.2; 1];
x_trajectory = [x_k; f(x_k)];
alpha_values = [];


while norm(f_grad(x_k)) > epsilon
    %p_k = - f_hessian(x_k)\f_grad(x_k);  %Newton
    %p_k = - f_grad(x_k);  %Steepest
    p_k = - H_k * f_grad(x_k); %BFGS

    alpha = 1;
    while f(x_k + alpha*p_k) > f(x_k) + c*alpha*f_grad(x_k)'*p_k
        alpha = rho*alpha;
    end
    alpha_values = [alpha_values; alpha];
    x_k_new = x_k + alpha*p_k;

    s_k = x_k_new - x_k;
    y_k = f_grad(x_k_new) - f_grad(x_k);
    g_k = 1/(y_k'*s_k);
    H_k = (eye(2) - g_k*s_k*y_k')*H_k*(eye(2) - g_k*y_k*s_k') +
 g_k*s_k*s_k';

    x_k = x_k_new;
    x_trajectory = [x_trajectory [x_k; f(x_k)]];
end

disp(x_k)

x = -1.5:0.1:1.5;
y = -1.5:0.1:1.5;
[X,Y] = meshgrid(x,y);
Z = 100*(Y-X.^2).^2+(1-X).^2;
mesh(X,Y,Z);
hold on;
plot3(x_trajectory(1,:), x_trajectory(2,:),
 x_trajectory(3,:), 'LineWidth', 5);
title('BFGS x0=(-1.2,1)');

figure(2);
plot(x_trajectory(3,:));
grid on;
```

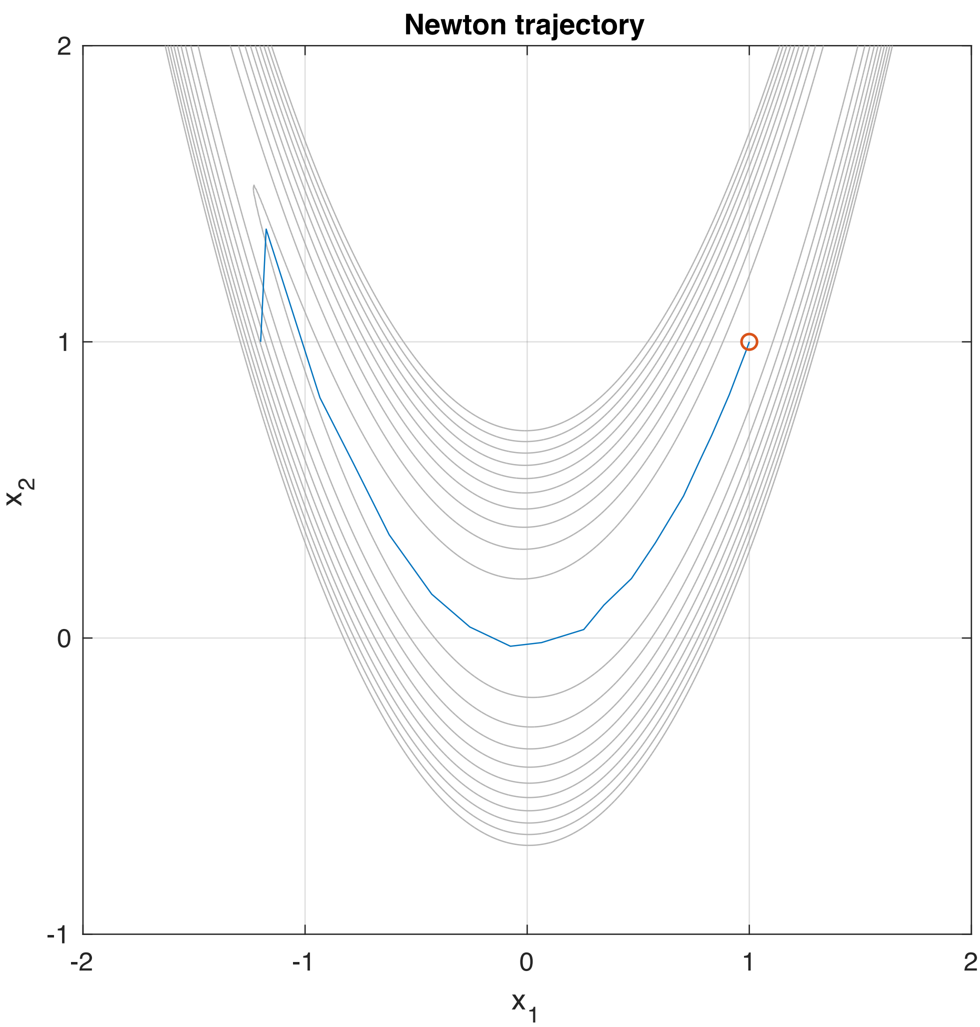
```
title('f(k) for BFGS');

figure(3);
plot(alpha_values);
grid on;
title('alpha for each iteration for steepest descent');
```

```
    1.0000
    1.0000
```

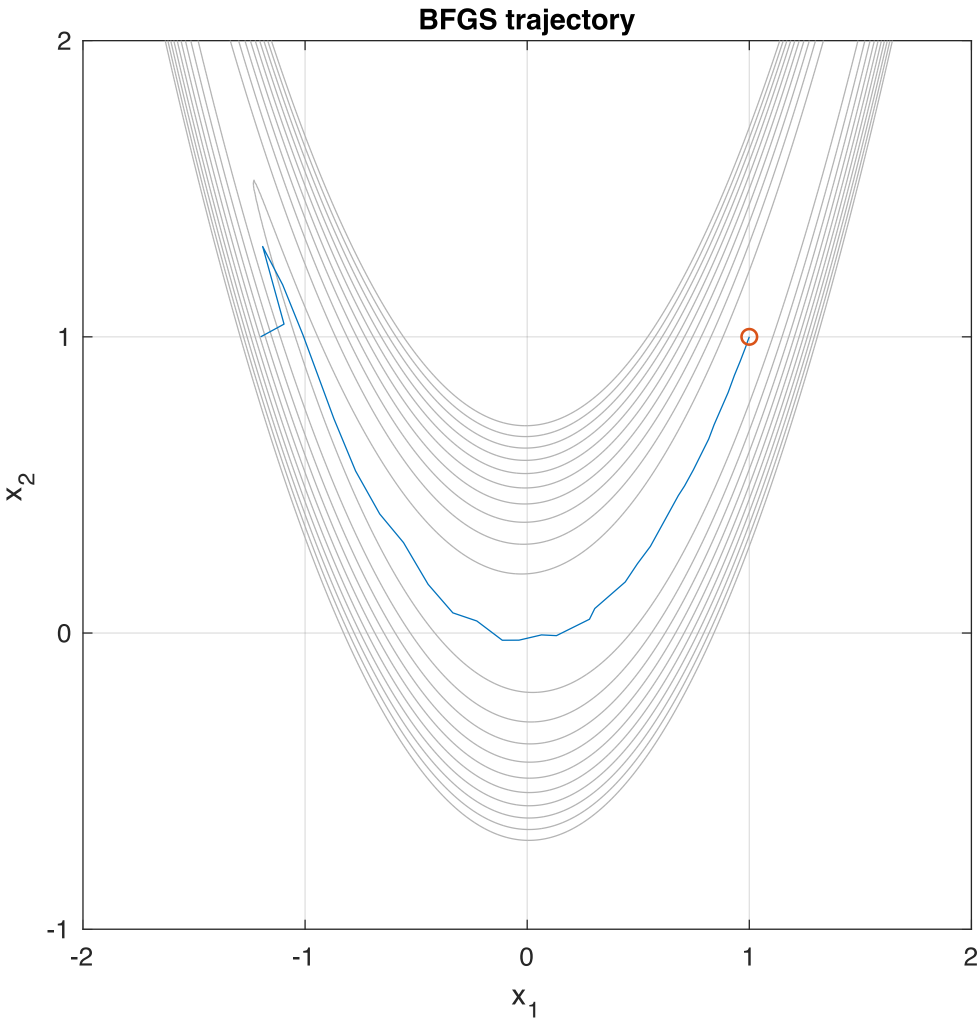
Published with MATLAB® R2018b



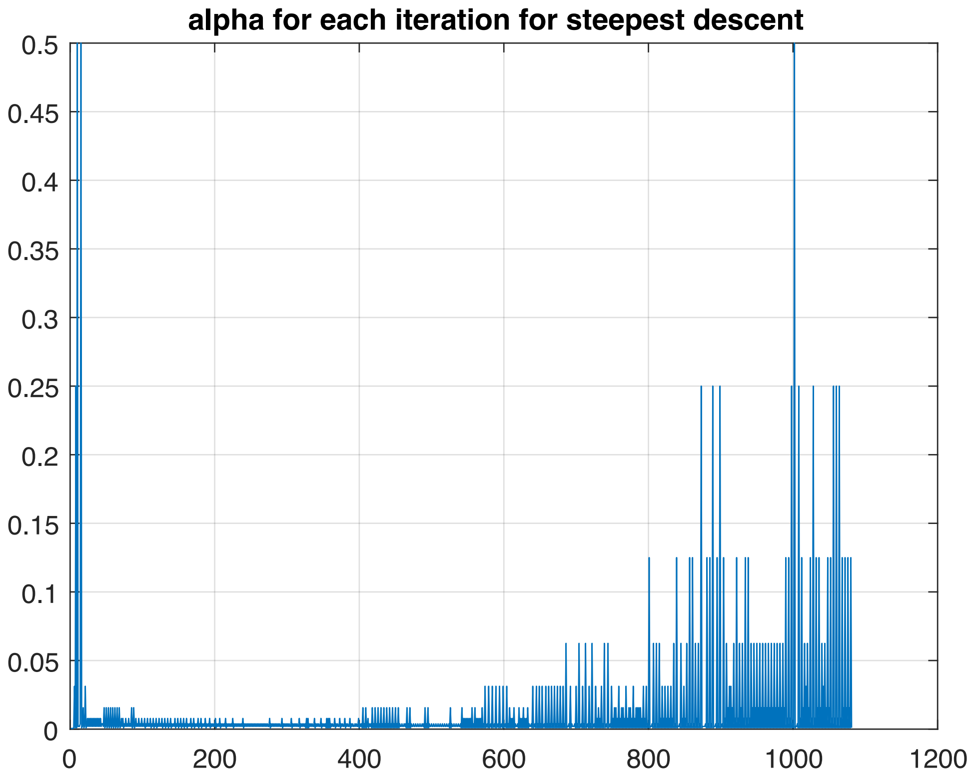
Steepest descent trajectory
2
1
0
-1
-2
-1
0
1
2
x_1
x_2

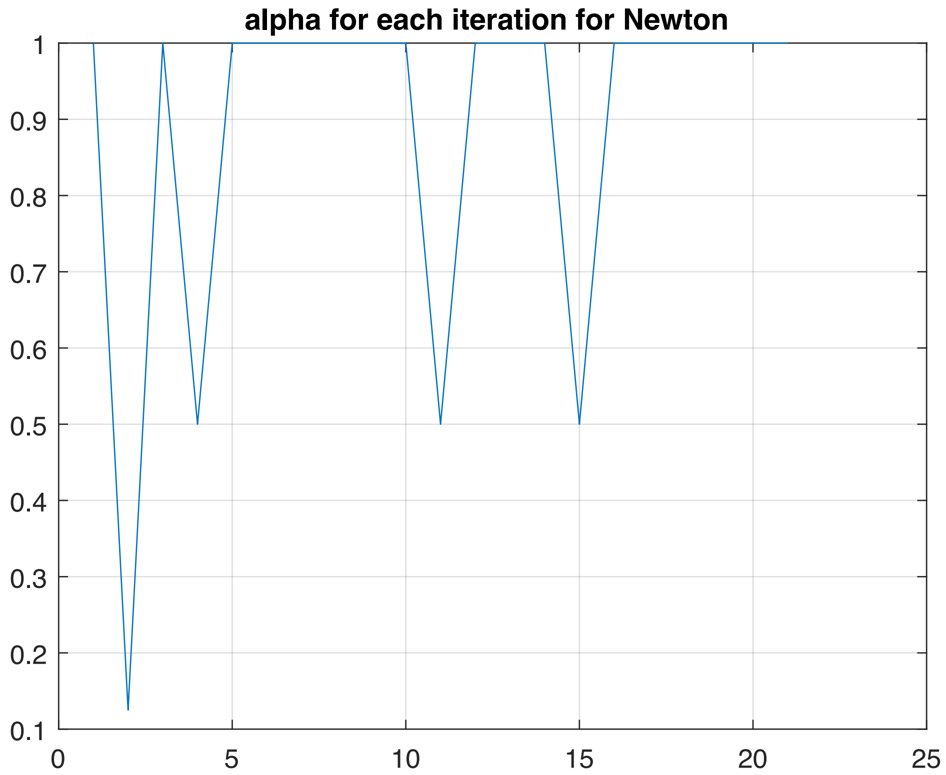


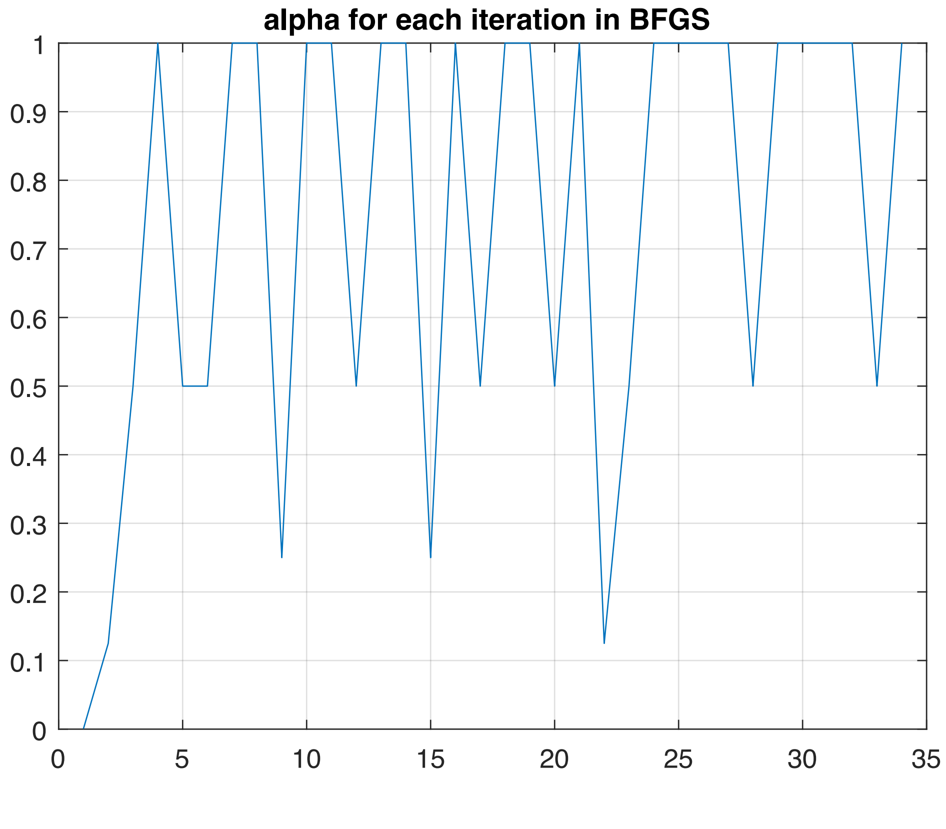
Newton trajectory
2
1
0
-1
-2
-1
0
1
2
x_1
x_2



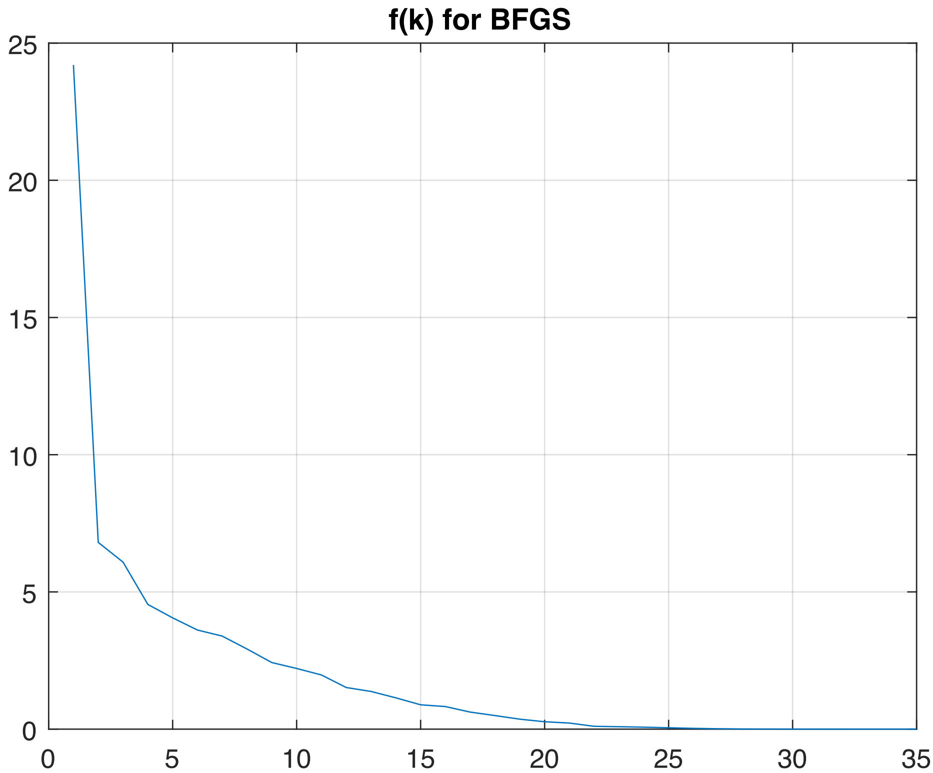
BFGS trajectory
2
1
0
-1
-2
-1
0
1
2
x1
x2


alpha for each iteration for steepest descent
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
200
400
600
800
1000
1200


alpha for each iteration for Newton
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5
10
15
20
25



alpha for each iteration in BFGS
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25
30
35



$\boxed{3}$ $f(x) = 100(x_2 - x_1)^2 + (1 - x_1)^2$

ⓐ $\frac{\partial f}{\partial x_i}(x) \approx \frac{f(x+\varepsilon e_i) - f(x)}{\varepsilon}$

$$\frac{\partial f}{\partial x_1}(x) \approx \frac{1}{\varepsilon_1}\left(100(x_2 - (x_1+\varepsilon_1))^2 + (1-(x_1+\varepsilon_1))^2 - 100(x_2-x_1)^2 - (1-x_1)^2\right)$$

$$= \frac{1}{\varepsilon_1}\big(100(x_2^2 - 2x_2x_1 - 2x_2\varepsilon_1 + x_1^2 + 2x_1\varepsilon_1 + \varepsilon_1^2 - x_2^2 + 2x_2x_1 - x_1^2)$$
$$+ 1 - 2x_1 - 2\varepsilon_1 + x_1^2 + 2x_1\varepsilon_1 + \varepsilon_1^2 - 1 + 2x_1 - x_1^2\big)$$

$$= \frac{1}{\varepsilon_1}\left(100(-2x_2\varepsilon_1 + 2x_1\varepsilon_1 + \varepsilon^2) - 2\varepsilon_1 + 2x_1\varepsilon_1 + \varepsilon_1^2\right)$$

$$= -200x_2 + 202x_1 + 101\varepsilon_1 - 2$$

$$\frac{\partial f}{\partial x_2}(x) \approx \frac{1}{\varepsilon_2}\left(100((x_2+\varepsilon_2) - x_1)^2 + (1-x_1)^2 - 100(x_2-x_1)^2 - (1-x_1)^2\right)$$

$$= \frac{1}{\varepsilon_2}\left(100(x_2^2 + 2x_2\varepsilon_2 + \varepsilon_2^2 - 2x_1 2x_2 - 2x_1\varepsilon_2 + x_1^2 - x_2^2 + 2x_1x_2 - x_1^2\right)$$

$$= \frac{1}{\varepsilon_2}100(2x_2\varepsilon_2 + \varepsilon_2^2 - 2x_1\varepsilon_2) = 200x_2 - 200x_1 + \varepsilon_2$$

$$\nabla f(x) \approx \begin{bmatrix} -200x_2 + 202x_1 + 101\varepsilon_1 - 2 \\ 200x_2 - 200x_1 + \varepsilon_2 \end{bmatrix}$$

Ops! $\varepsilon_1 = \varepsilon_2$

$$\nabla f(x) \approx \begin{bmatrix} -200x_2 + 202x_1 + 101\varepsilon - 2 \\ 200x_2 - 200x_1 + \varepsilon \end{bmatrix}$$

(b) $$\nabla f(x) = \begin{bmatrix} -200(x_2 - x_1) - 2(1-x_1) \\ 200(x_2 - x_1) \end{bmatrix}$$

$$\nabla f(x) = \begin{bmatrix} 202x_1 - 200x_2 - 2 \\ 200x_2 - 200x_1 \end{bmatrix}$$

(notice that this is the same as the approximation with $\epsilon = 0$)

$$\nabla f(x^1) = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad \nabla f(x^2) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\nabla f(x^1) \approx \begin{bmatrix} 101\epsilon_1 - 1 \\ \epsilon \end{bmatrix}, \quad \nabla f(x^2) \approx \begin{bmatrix} 101\epsilon \\ \epsilon \end{bmatrix}$$

For large ϵ the two approximations are almost the same, for small ϵ they are almost equal to the real gradient.

(c) The error is linear in ϵ, which is quite poor. A small ϵ has to be chosen to get a satisfactory approximation.

Scanned by CamScanner

4 (a) The corresponding matrix

$$V(s) = [z_2 - z_1 \quad z_3 - z_1 \quad \cdots \quad z_{n+1} - z_1]$$

has to be nonsingular.

In geometric terms this loosely means that the simplex "fills" the entire space $\mathbb{R}^n$, not just a subspace of $\mathbb{R}^n$, i.e. all the faces on the simplex are linearly independent.

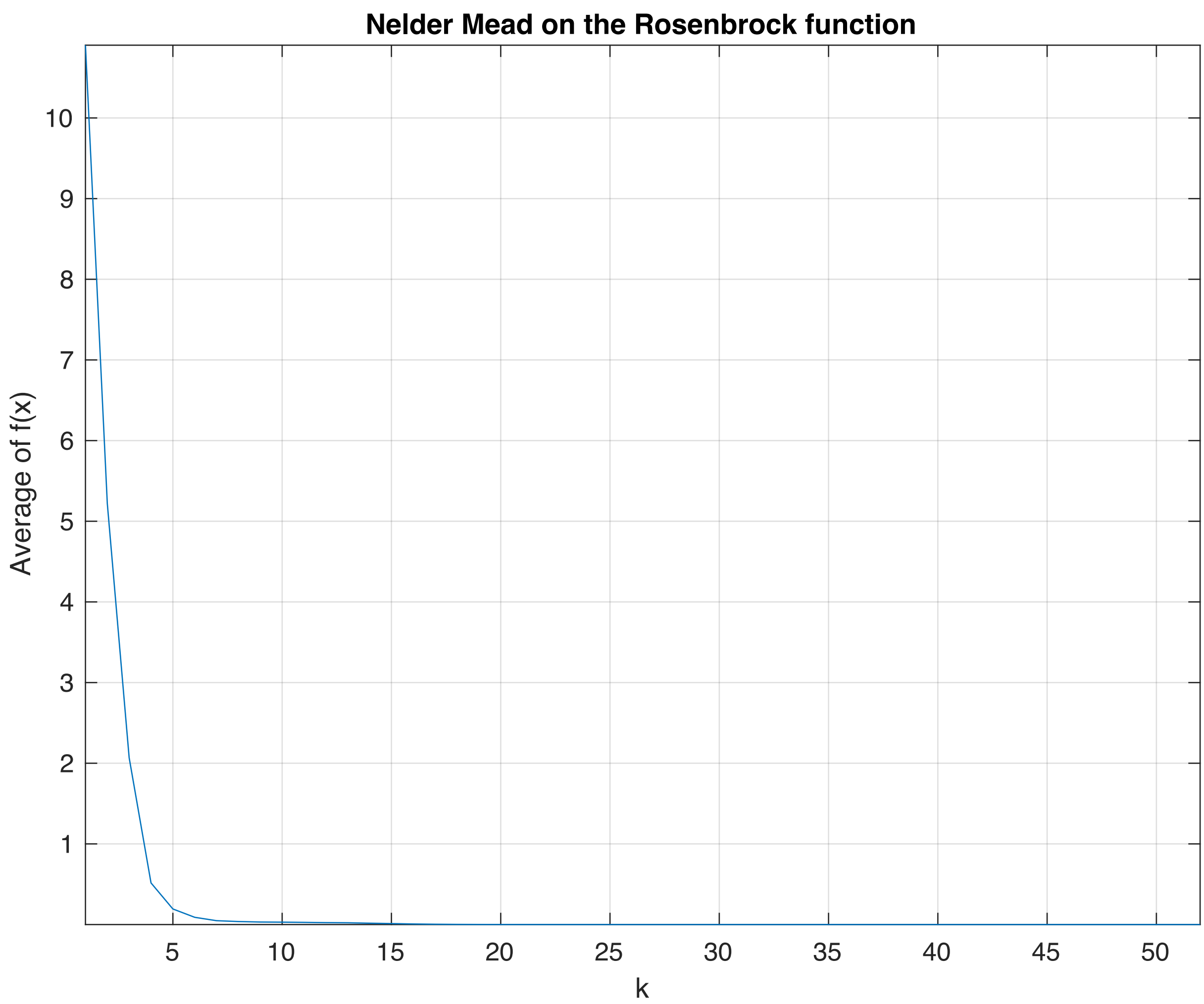
Example:

$$S = \left\{ \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1{,}5 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix} \right\}$$

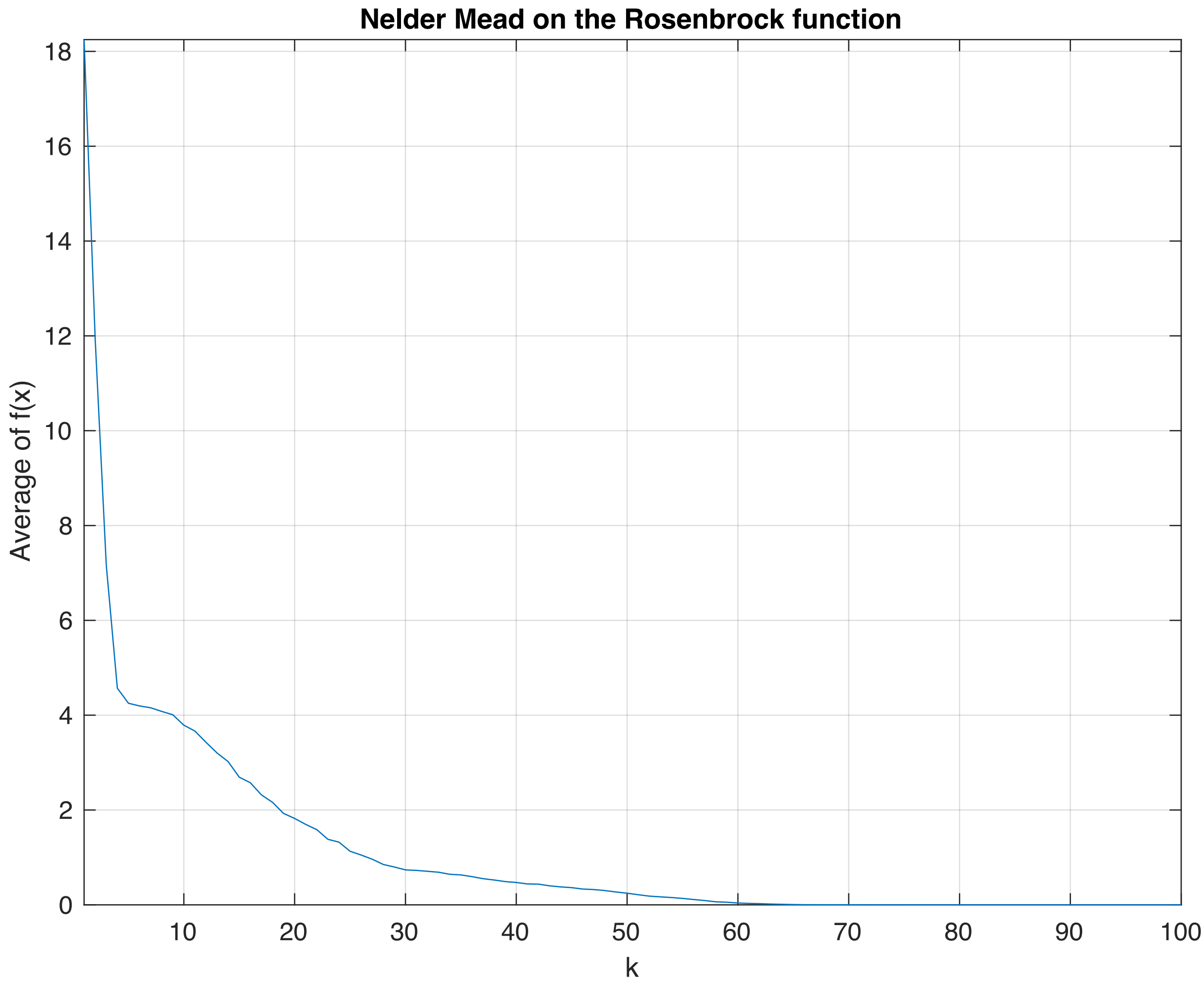
(b) (c) Without the derivative we observe that the trajectory is a lot more oscillatory, almost like a random walk, compared to the derivative-based methods.

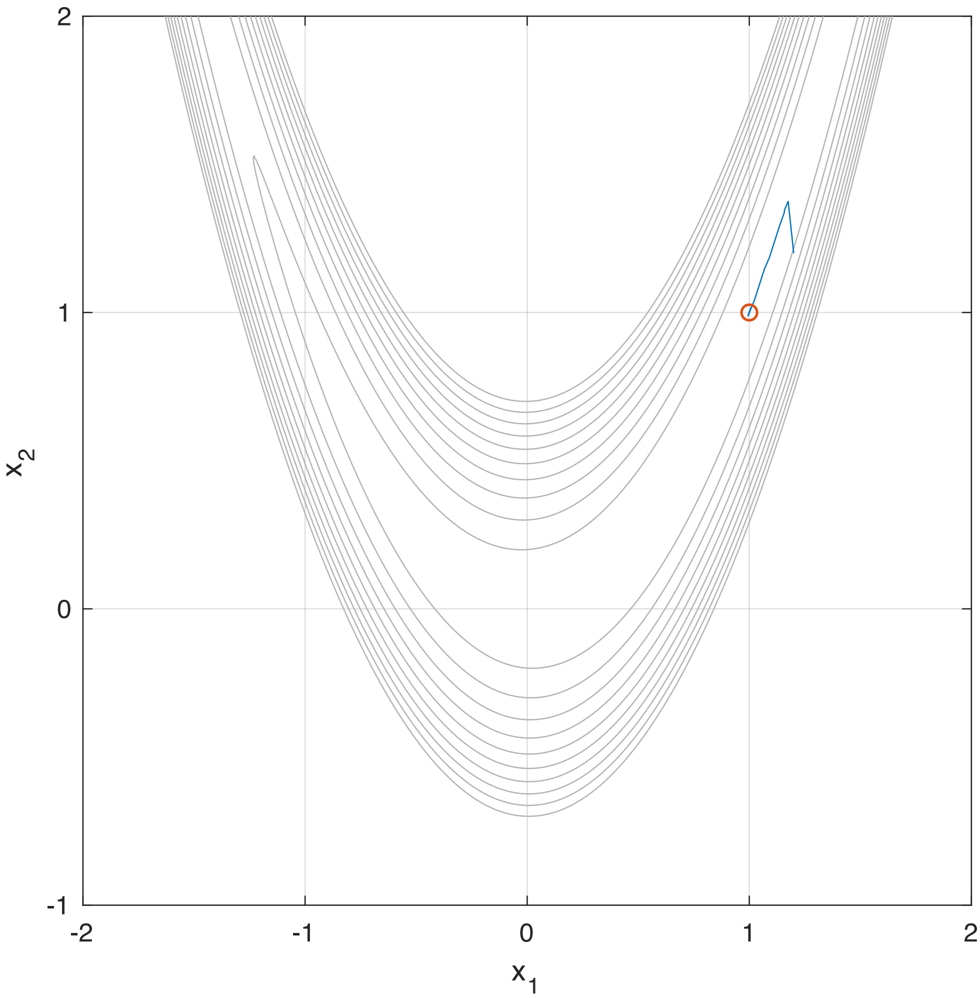
Moral of the story - use the gradient if you have access to it!

We observe that NM is slower than BFGS, but faster than steepest descent - about 70 iterations to converge.

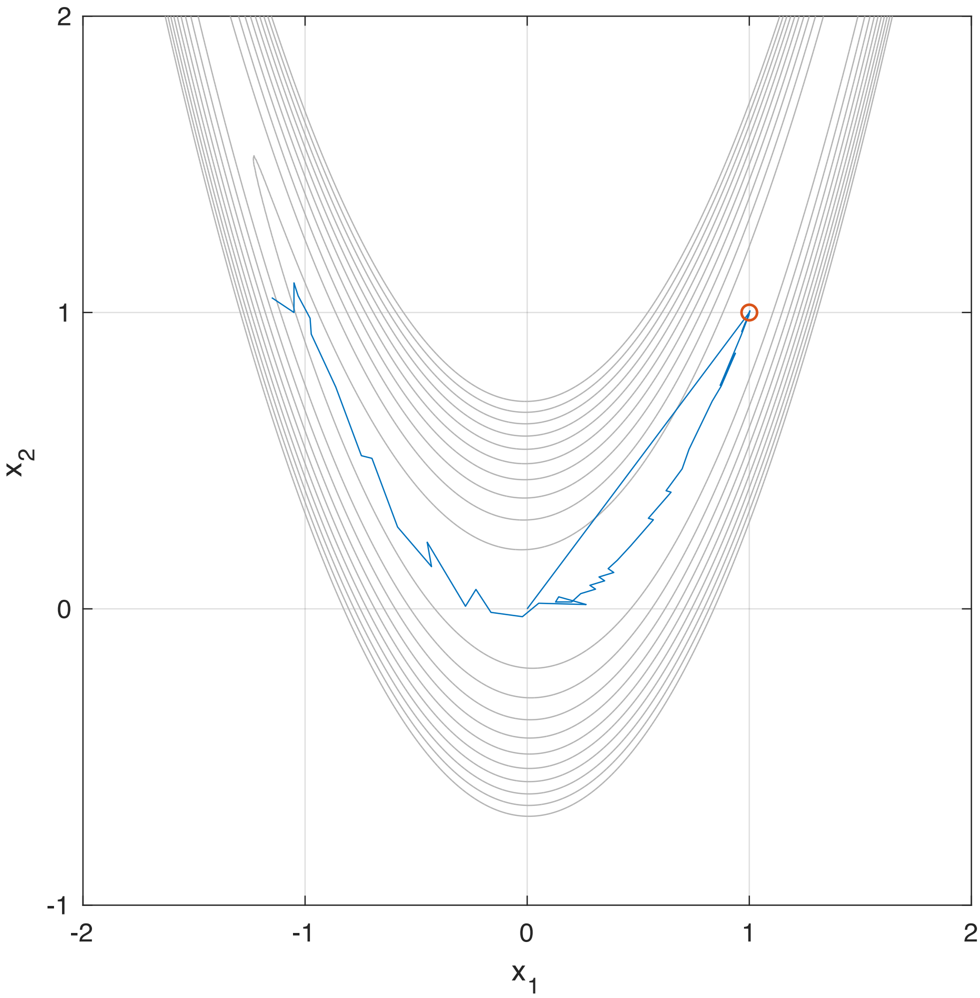


Nelder Mead on the Rosenbrock function
Average of f(x)
k
10
9
8
7
6
5
4
3
2
1
5
10
15
20
25
30
35
40
45
50


Nelder Mead on the Rosenbrock function
Average of f(x)
k
0
2
4
6
8
10
12
14
16
18
10
20
30
40
50
60
70
80
90
100



2
1
0
-1
-2
-1
0
1
2
x_1
x_2



2
1
0
-1
-2
-1
0
1
2
x_1
x_2