Report on Problem 3.1

王睿 3180103650

> $\rho = 0.5; c_1 = 10^{-4}; alpha\_orig = 1; tol = 10^{-6}$

1. Startint point (1.2, 1.2)

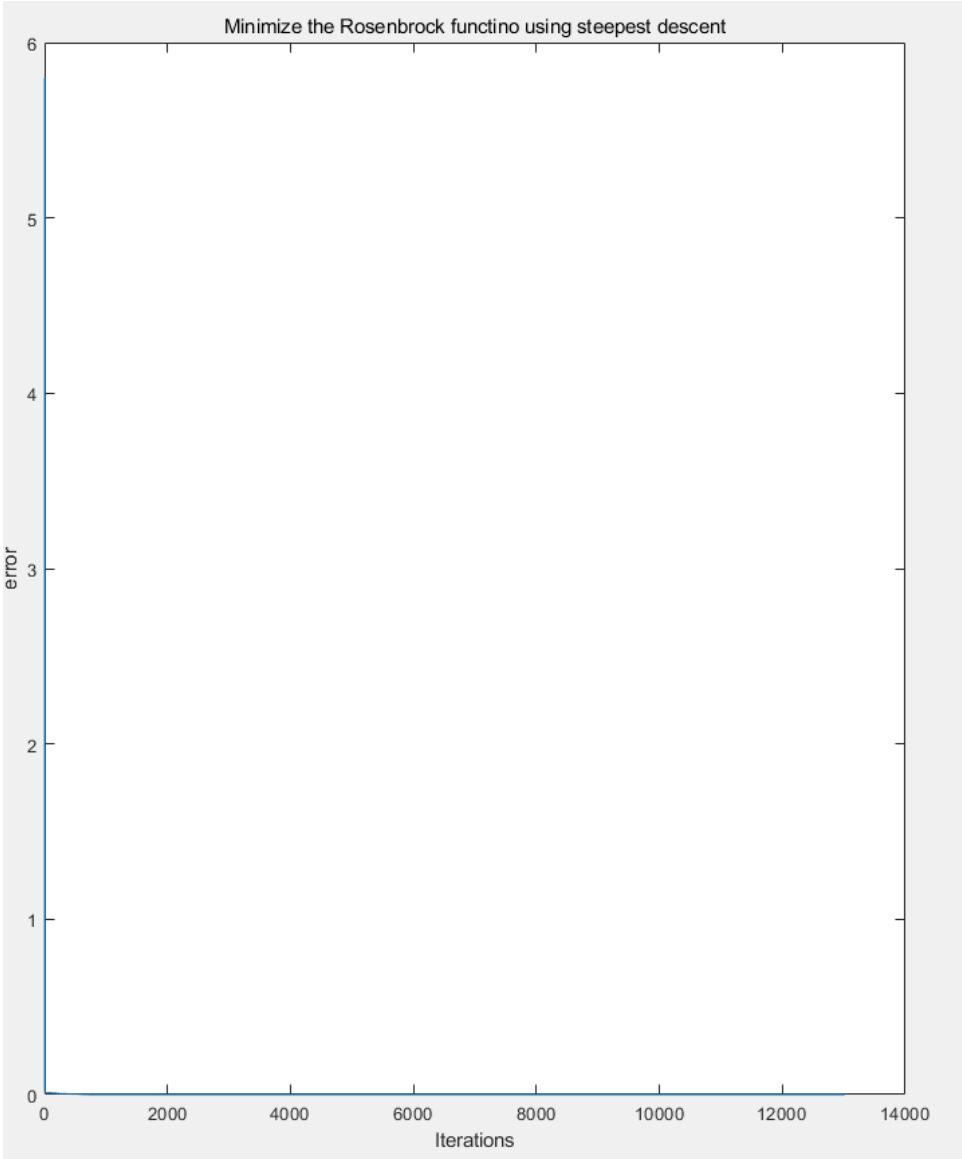
1.1 Steepest Desceding Method

- iterative value

iteration	iterative value
1	(1.200000000000000, 1.200000000000000)
2	(1.087109375000000, 1.246875000000000)
3	(1.114570590131916, 1.234166365861893)
...	...
13010	(1.000000786154006, 1.000001577190118)
13011	(1.000000786896757, 1.000001575283287)
13012	(1.000000784986344, 1.000001574701586)

- error

iteration	error
1	5.800000000000000
2	0.430975196661893
3	0.019689420206267
...	...
13010	6.214176813516914e-13
13011	6.204210132071140e-13
13012	6.194282638369803e-13



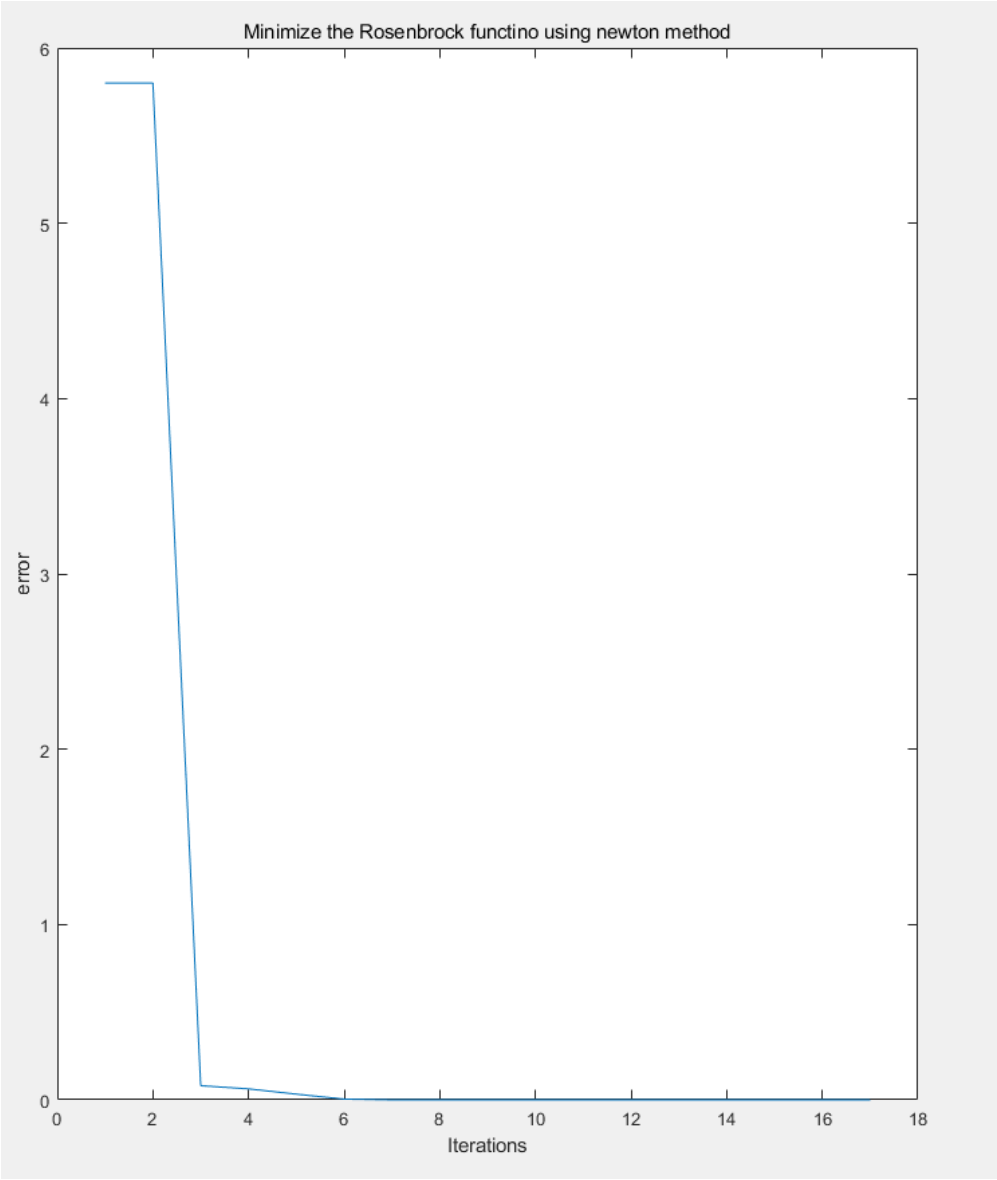
1.2 Newton Method

- iterative value

iteration	iterative value
1	(1.200000000000000, 1.200000000000000)
2	(1.195568072465431, 1.449960712949712)
3	(1.250084738876800, 1.562445970132116)
4	(1.157206637467995, 1.330338972653161)
5	(1.052185925067608, 1.103467533360077)
6	(0.995933975872727, 0.991839946215247)
7	(1.002913537104435, 1.005819443703099)
8	(0.999420067576715, 0.998831537943223)
9	(1.000288569973151, 1.000580938084132)
10	(0.999854174977503, 0.999706309417690)
11	(1.000071336819735, 1.000143641457884)
12	(0.999962746493339, 0.999924980108330)
13	(1.000017039584652, 1.000034311812009)
14	(0.999989892369197, 0.999979646292791)
15	(1.000003465879976, 1.000006979100545)
16	(0.999996679056908, 0.999993312730277)
17	(1.000000072468056, 1.000000145915603)

- error

iteration	error
1	5.800000000000000
2	0.080591032560144
3	0.062549446062015
4	0.032437224017700
5	0.004039382471424
6	1.673091628204808e-05
7	8.514681333419767e-06
8	3.443024146596463e-07
9	8.465265175018618e-08
10	2.169004003921006e-08
11	5.181626653460732e-09
12	1.414270728046684e-09
13	2.957462069120710e-10
14	1.040837488285476e-10
15	1.223632345631299e-11
16	1.123472969281601e-11
17	5.347360582954264e-15



2. Startint point (1.2, 1)

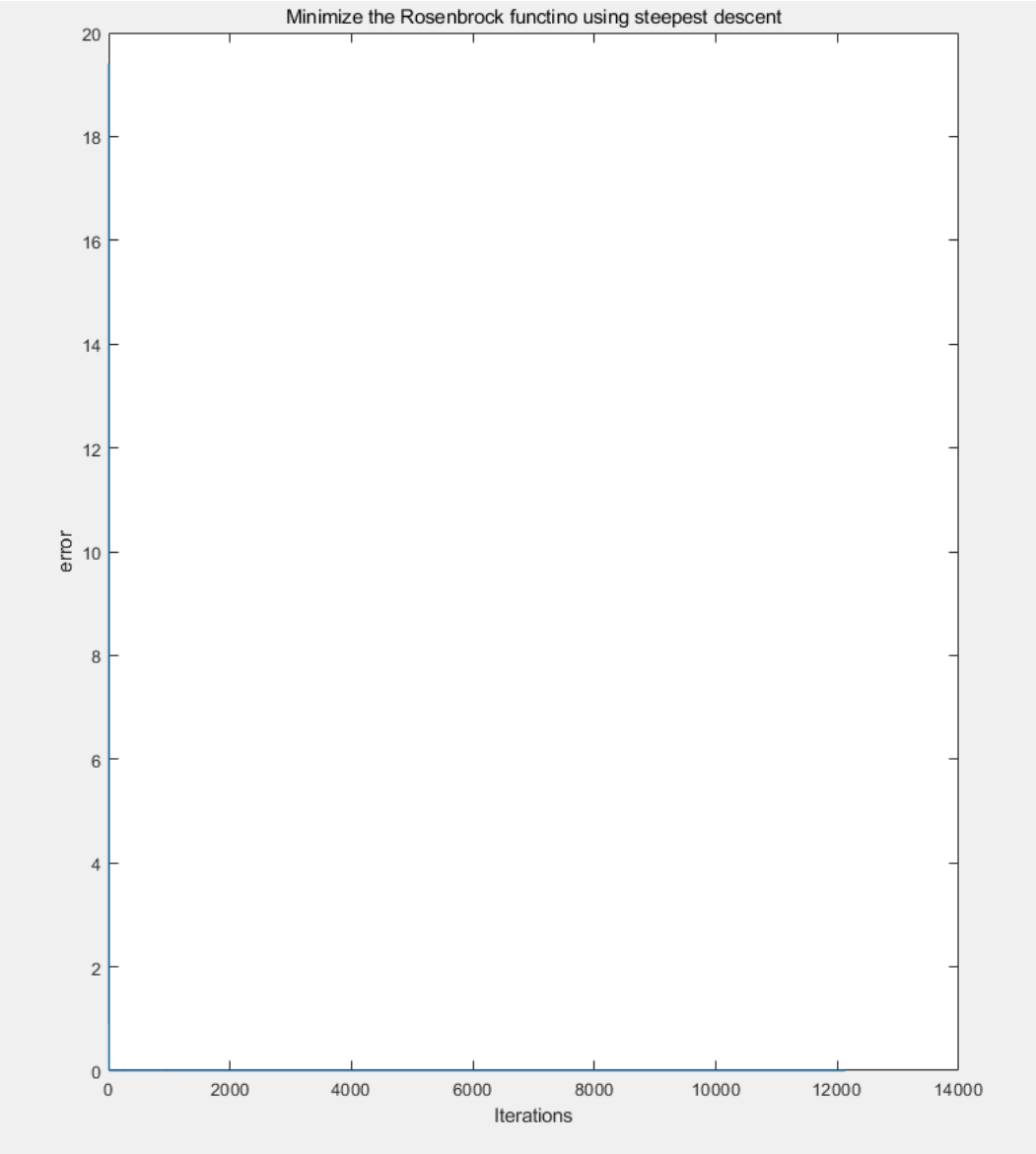
1.1 Steepest Desceding Method

- iterative value

iteration	iterative value
1	(1.200000000000000, 1)
2	(0.993359375000000, 1.085937500000000)
3	(1.070350994961336, 1.047197401523590)
...	...
12129	(1.000000789658741, 1.000001580753313)
12130	(1.000000787695393, 1.000001580192685)
12131	(1.000000788369459, 1.000001578317186)

- error

iteration	error
1	19.399999999999995
2	0.983605259808249
3	0.974265338021387
...	...
12129	6.247684553983333e-13
12130	6.237669088858528e-13
12130	6.227692593714632e-13



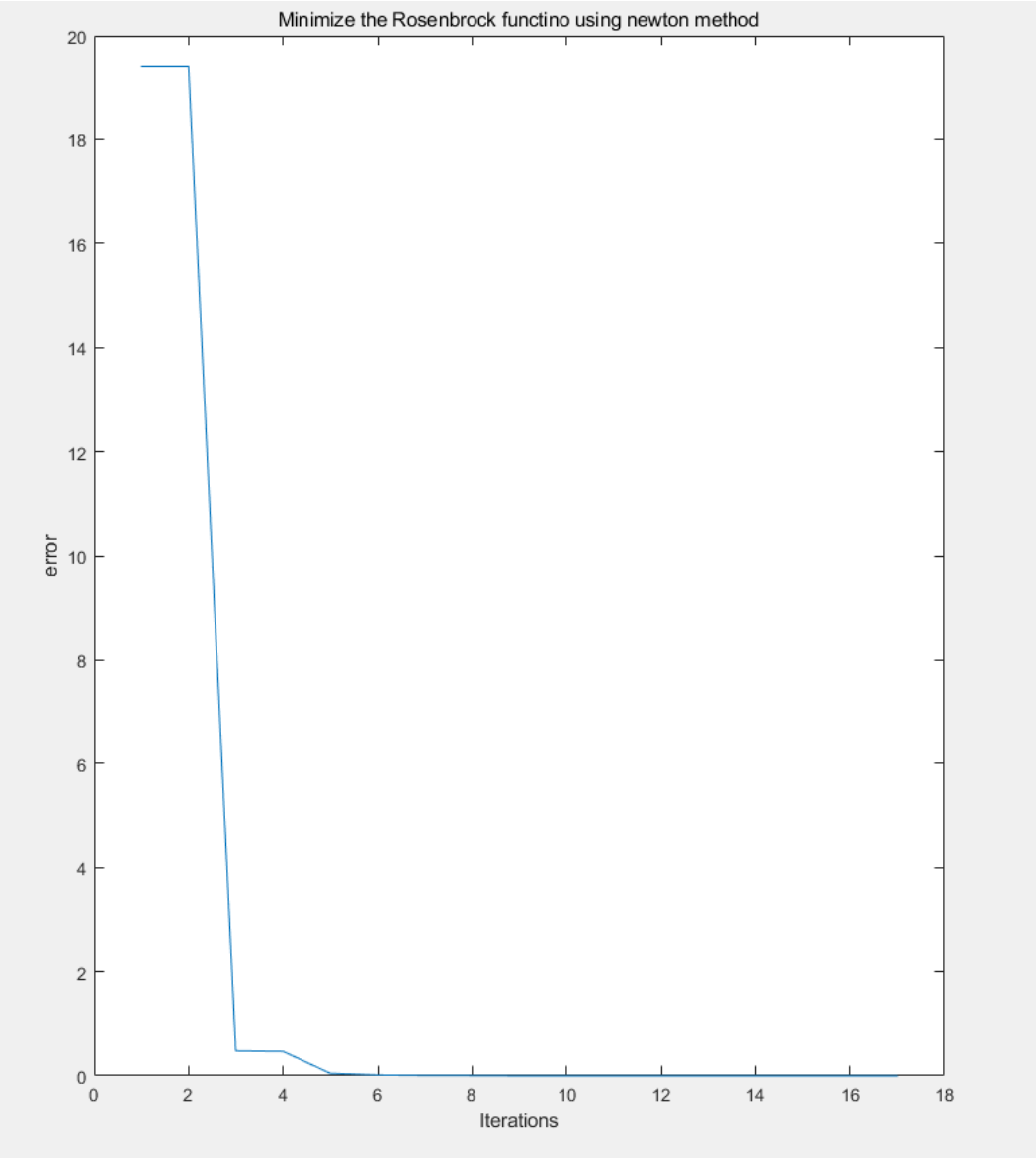
1.2 Newton Method

- iterative value

iteration	iterative value
1	(1.200000000000000, 1)
2	(1.197414719150578, 1.499993316278258)
3	(1.228985506073434, 1.446053206776407)
4	(1.211161132689385, 1.471721362406799)
5	(1.115792939031264, 1.240626343770768)
6	(1.010798871867163, 1.013742572071821)
7	(1.003030245799270, 1.012915767822161)
8	(1.013172600933867, 1.024801610548855)
9	(0.998320299057804, 0.997306672305929)
10	(1.000341191221935, 1.000649542917164)
11	(0.999887007577115, 0.999785024824129)
12	(1.000000590482209, 1.000001134908220)
13	(0.999999710134475, 0.999999442877879)
14	(1.000000150308378, 1.000000288893031)
15	(0.999999930221434, 0.999999865885451)
16	(1.000000040264909, 1.000000077389240)
17	(0.999999985243172, 0.999999971637345)

- error

iteration	error
1	19.399999999999995
2	0.477101478805718
3	0.466554506412544
4	0.046902704251927
5	0.015315544439719
6	0.006471554903042
7	0.004696082468021
8	4.683636558841746e-04
9	4.681182227150957e-05
10	2.250208359041570e-07
11	2.486047433418362e-08
12	5.607897744942004e-13
13	1.351380077689517e-13
14	3.633723188164070e-14
15	7.831213215739661e-15
16	2.607586445768588e-15
17	3.502443028154872e-16



Source Code

```
function main()
    alpha_orig = 1;
    rho = 0.5;
    c = 1e-4;

    x_0 = [1.2; 1.2]; % the starting point
    [x_k, error, alpha, iter] = backtracking_line_search(x_0, 'steepest descend', alpha_orig, rho, c);
    [x_k, error, alpha, iter] = backtracking_line_search(x_0, 'newton method', alpha_orig, rho, c);

    x_1 = [1.2; 1]; % a more difficult starting point
    [x_k, error, alpha, iter] = backtracking_line_search(x_1, 'steepest descend', alpha_orig, rho, c);
    [x_k, error, alpha, iter] = backtracking_line_search(x_1, 'newton method', alpha_orig, rho, c)

end


```

```
function [x_k, error, alpha, iter] = backtracking_line_search(x_0, method, alpha_orig, rho, c)
% [x_k, error, alpha, iter] = backtracking_line_search(x_0, method, alpha_orig, rho, c)
% x_k: the iterative point at each iteration; error: the difference between
%the global minimization and the function value at each iteration
% alpha: the step length at each iteration; iter: the iteration times
% x_0: the starting point; method: the used method(steepest descend or newton);
% alpha_orig: the original step length; rho: a scale factor; c: c1 in the Armijo condition

    iter = 1;
    tol = 1e-6;
    max_iter = 1e6;
    error = zeros(max_iter, 1);
    f_k = zeros(max_iter, 1);
    x_k = zeros(max_iter, 2);
    alpha = alpha_orig * ones(max_iter, 1);
    x_k(1,1) = x_0(1);
    x_k(1,2) = x_0(2);
    alpha(1) = 0;
    f_k(1) = Rosenbrock(x_0);

    x = x_0;
    while iter < max_iter && norm(Rb_gradient(x)) > tol
        iter = iter + 1;
        p = step_direction(x, method);
        [alpha(iter), f_k(iter)] = step_length(x, p, alpha(iter), rho, c);
        x = x + alpha(iter) * p;
        x_k(iter, 1) = x(1);
        x_k(iter, 2) = x(2);
    end
    error = abs(0 - f_k);
end


function p = step_direction(x_k, method)
% p = step_direction(x_k, method)
% x_k: a 2-d column vector; method: a string ('steepest descend' or 'newton methd')
% This function returns the descending direction using steepest descend or
% newton method
    if strcmp(method, 'steepest descend')
        p = - Rb_gradient(x_k);
    else
        p = - Rb_hessian(x_k)^(-1) * Rb_gradient(x_k);

    p = p / norm(p);    %unitization the direction p
end


function [alpha, f_k] = step_length(x_k, p_k, alpha_orig, rho, c)
%[alpha, f_k] = step_length(x_k, p_k, alpha_orig, rho, c)
% x_k: a 2-d column vector; p_k: the chosen descending direction
% alpha_orig: the original step length; rho: a scale factor; c: c1 in the Armijo condition
```

```
% This function returns the step length according to the Armijo condition

    alpha = alpha_orig;
    f_k = Rosenbrock(x_k);
    while(Rosenbrock(x_k + alpha * p_k) > Rosenbrock(x_k) + c * alpha *
Rb_gradient(x_k)' * p_k)
        alpha = rho * alpha;
    end

end


function h = Rb_hessian(x)
% h = Rb_hessian(x)
% x : a 2-d column vector
% This function calculates the Hessian matrix of the Rosenbrock function
at
% point x
    h = [1200*x(1)^2-400*x(2)+2, -400*x(1);
        -400*x(1),               200      ];
end


function g = Rb_gradient(x)
% g = Rb_gradient(x)
% x : 2-d column vector
% This function calculates the gradient of Rosenbrock function at point x
    g = [-400*x(1)*(x(2)-x(1)^2)-2*(1-x(1));200*(x(2)-x(1)^2)];
end


function f = Rosenbrock(x)
% f = Rosenbrock(x)
% x : 2-d column vector
% This function calculates the value of 2-d Rosenbrock function at point x
    f = 100 * (x(2)-x(1).^2).^2+(1-x(1)).^2;
end
```