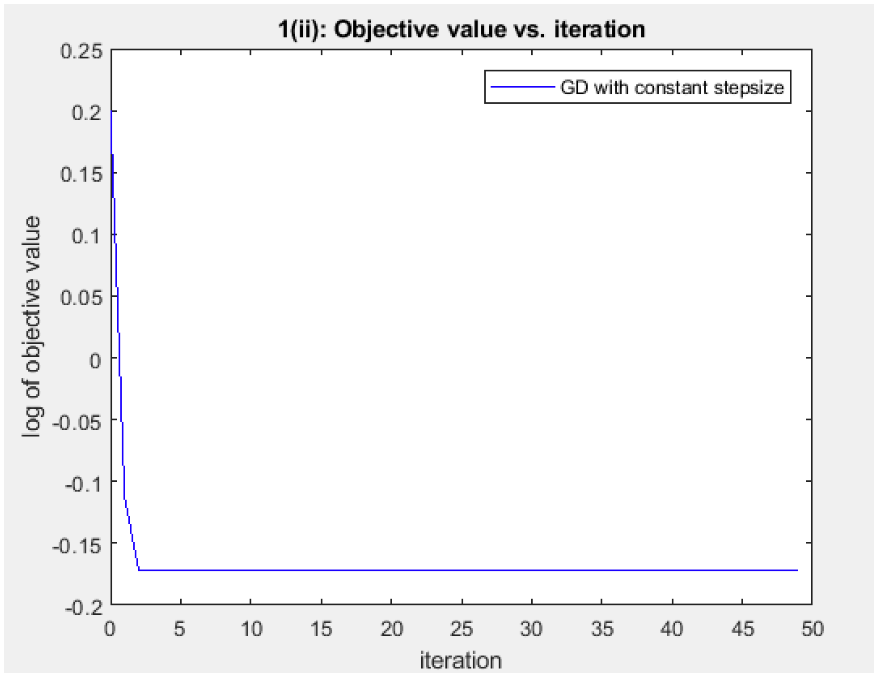
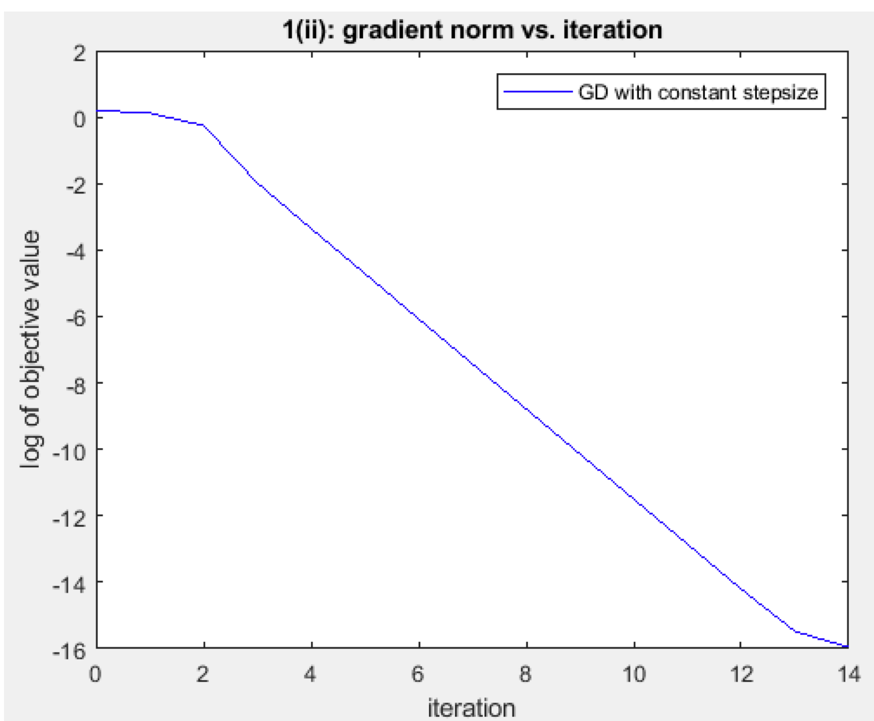
IE510 HW3

1.(i)

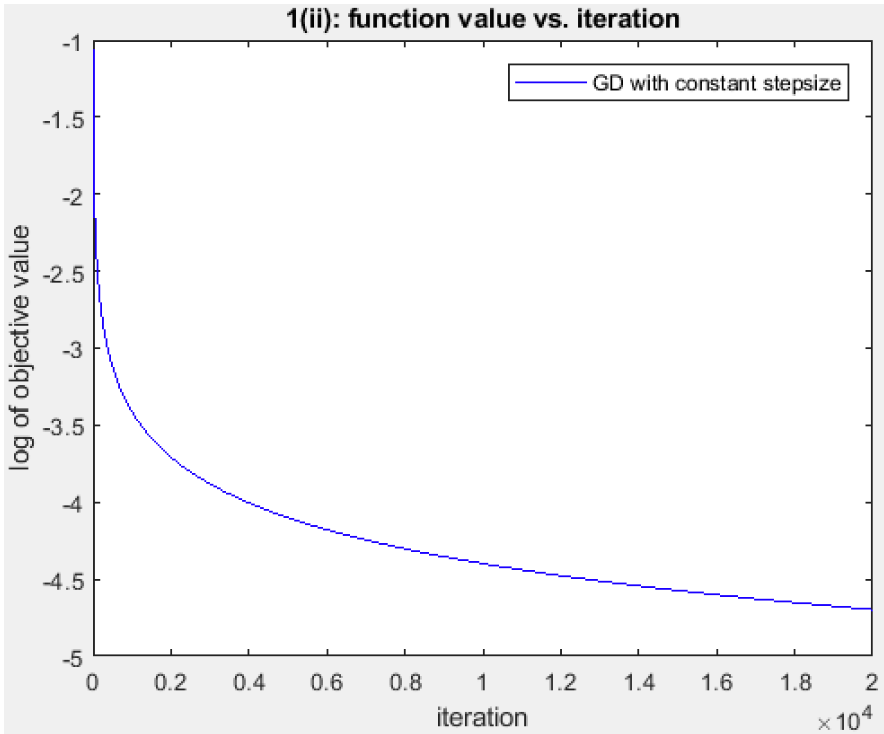
When x =(2,3), y = (1,-1), The function value vs. iterations is below:



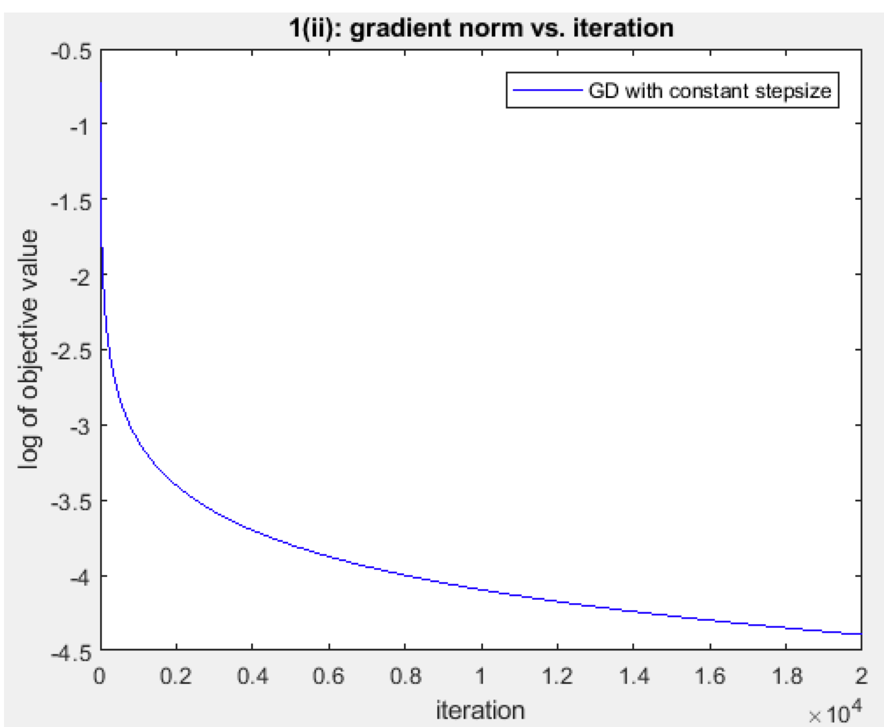
The gradient norm vs. iterations is below:



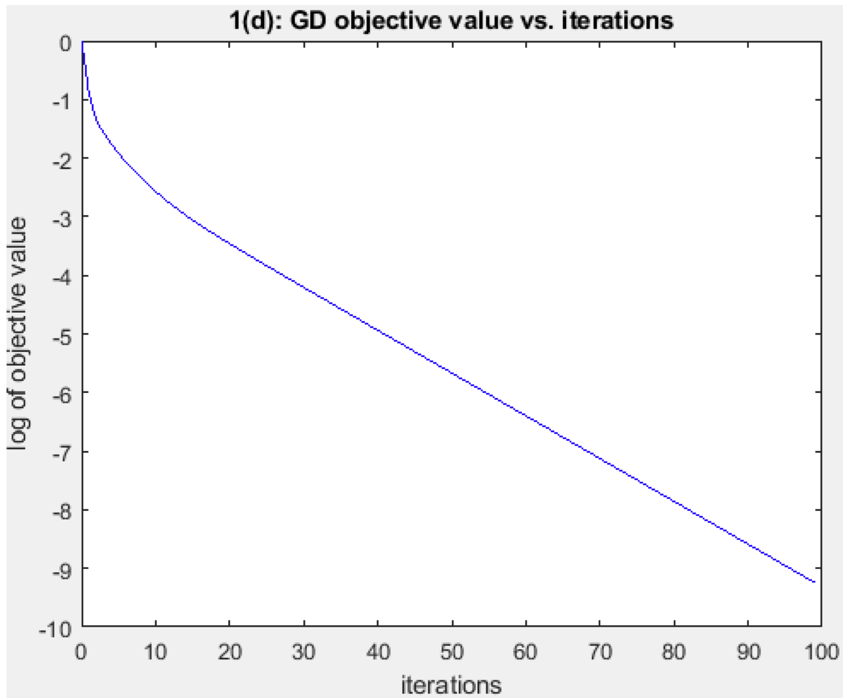
When x =(2,-3), y = (1,-1), The function value vs. iterations is below:



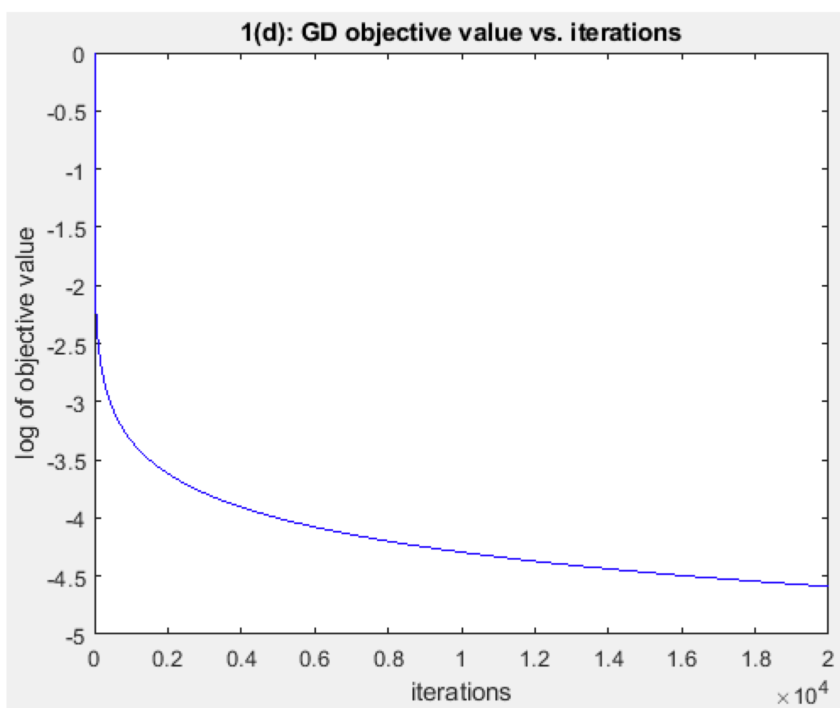
The gradient norm vs. iterations is below:



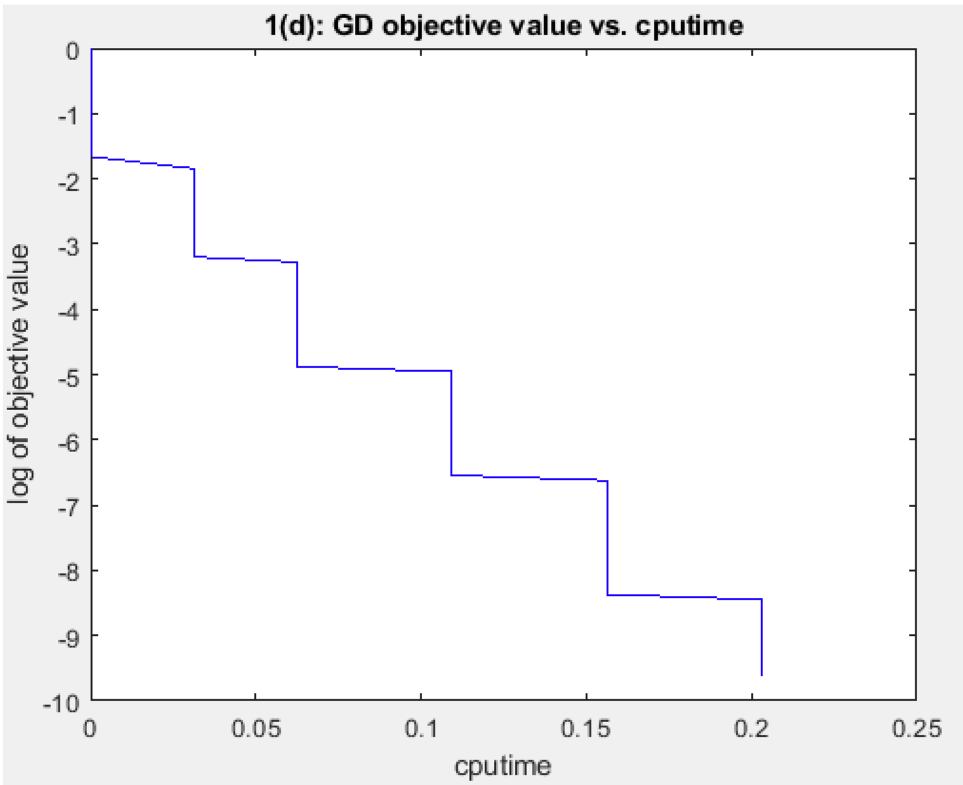
1.(d) Setting 1 : Random case



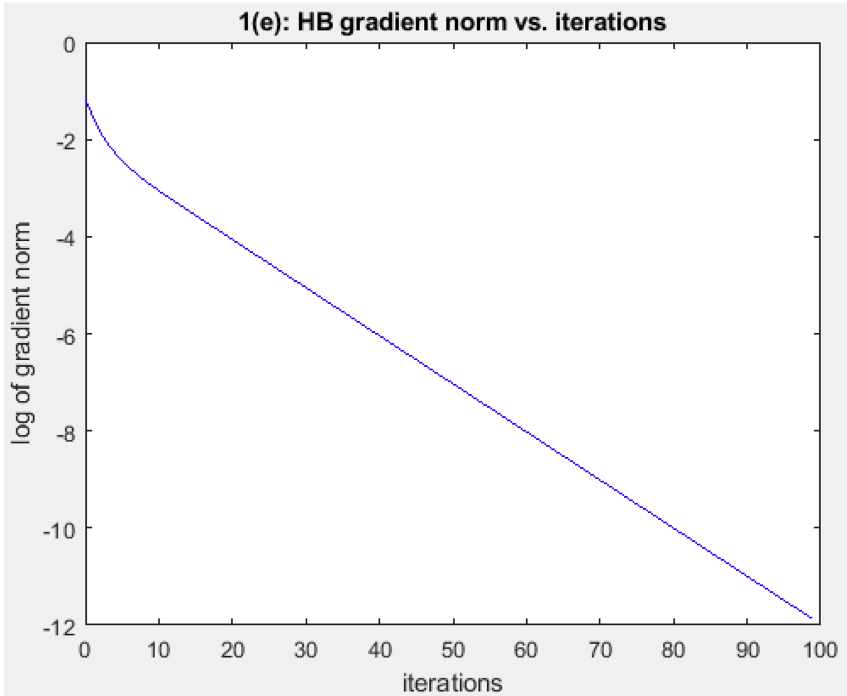
Setting 2 : Separable case



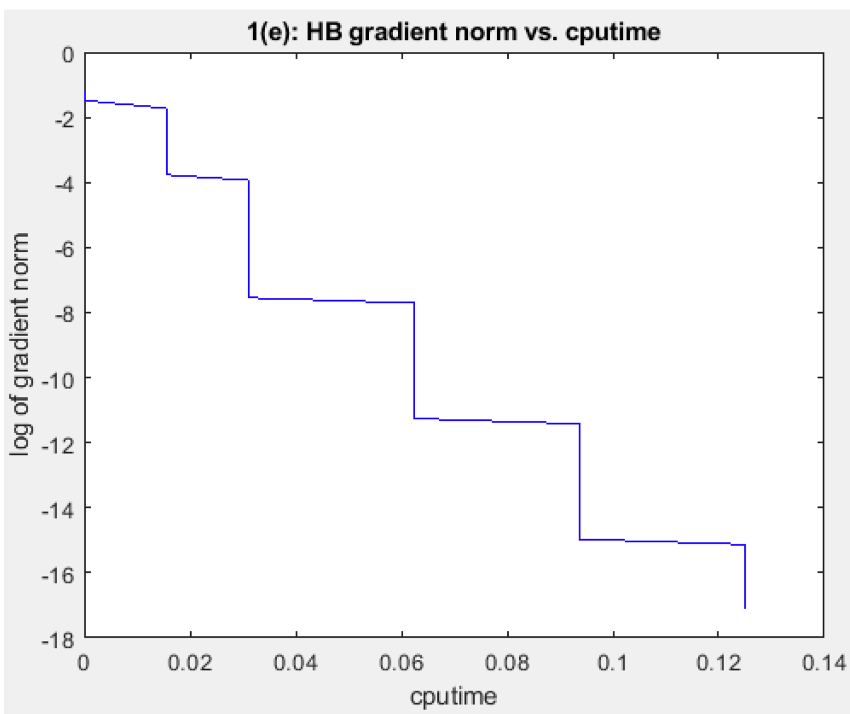
1.(e) Gradient Descent : gradient norm vs. cputime



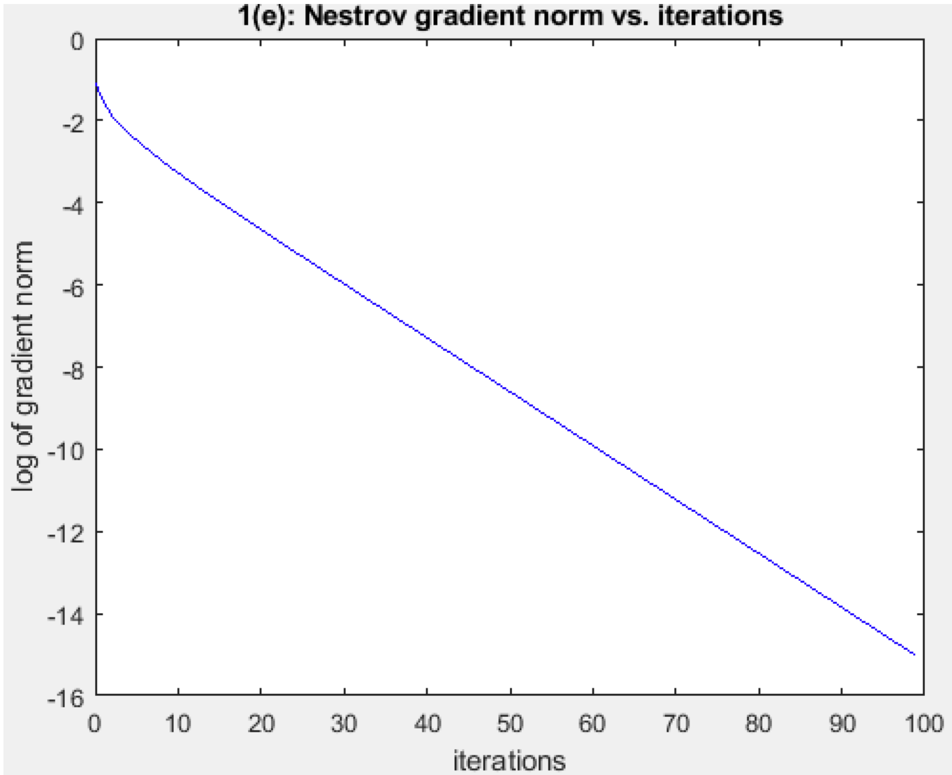
Heavy Ball method : gradient norm vs. iteration

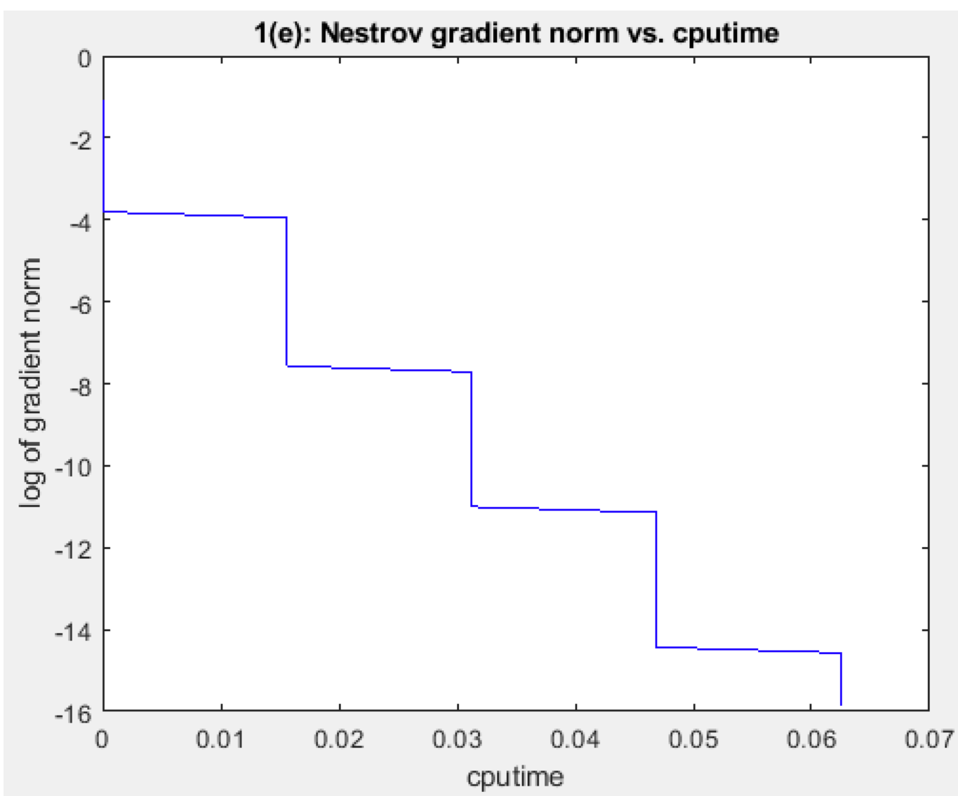


gradient norm vs. cputime



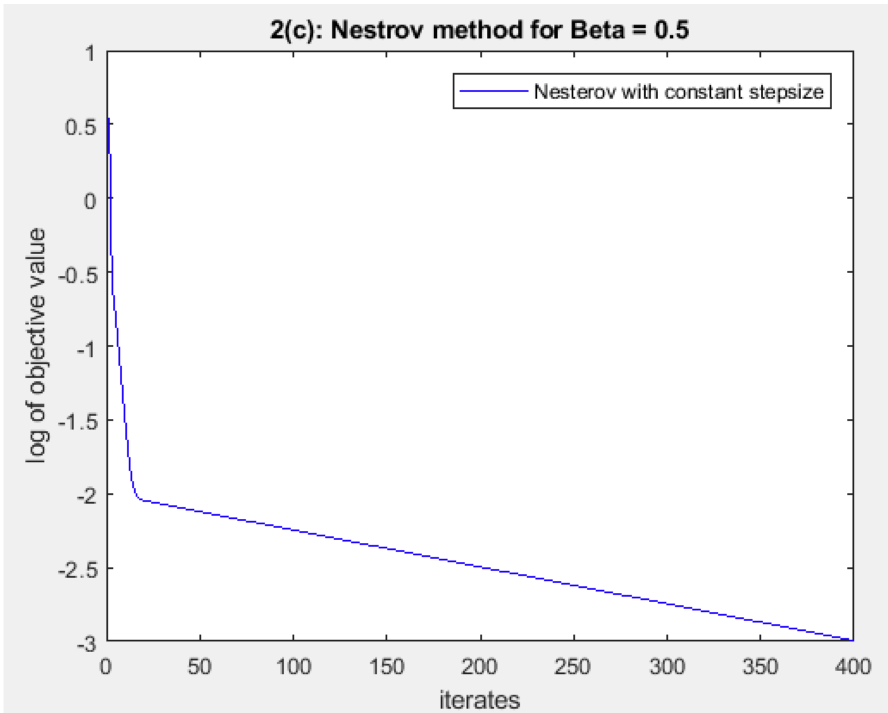
Nesterov’s Method: gradient norm vs. iterations

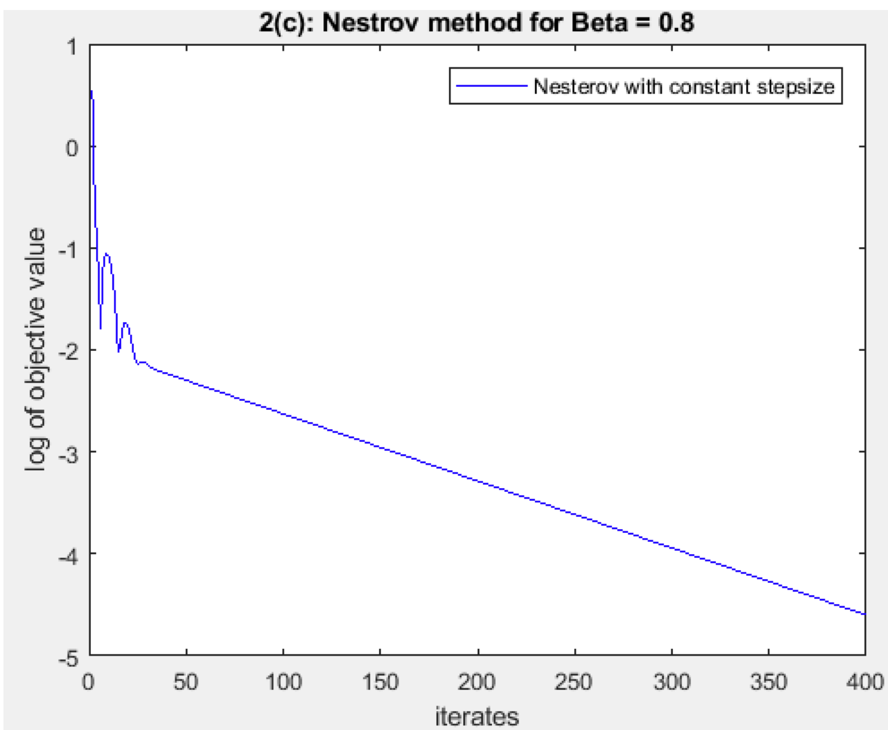


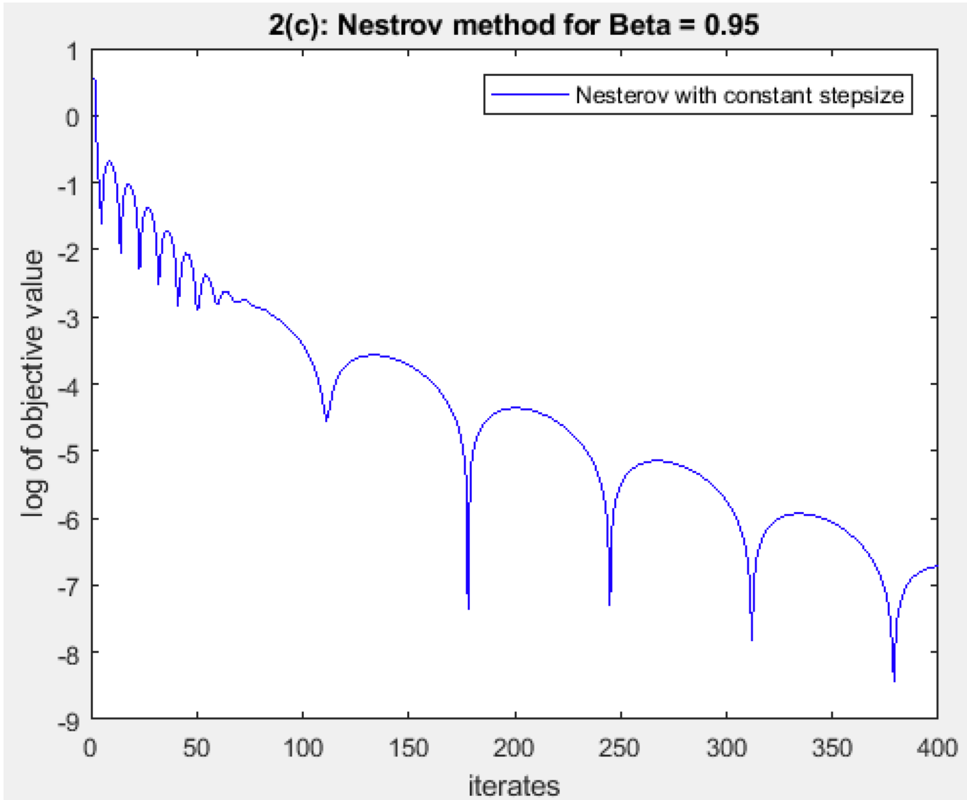


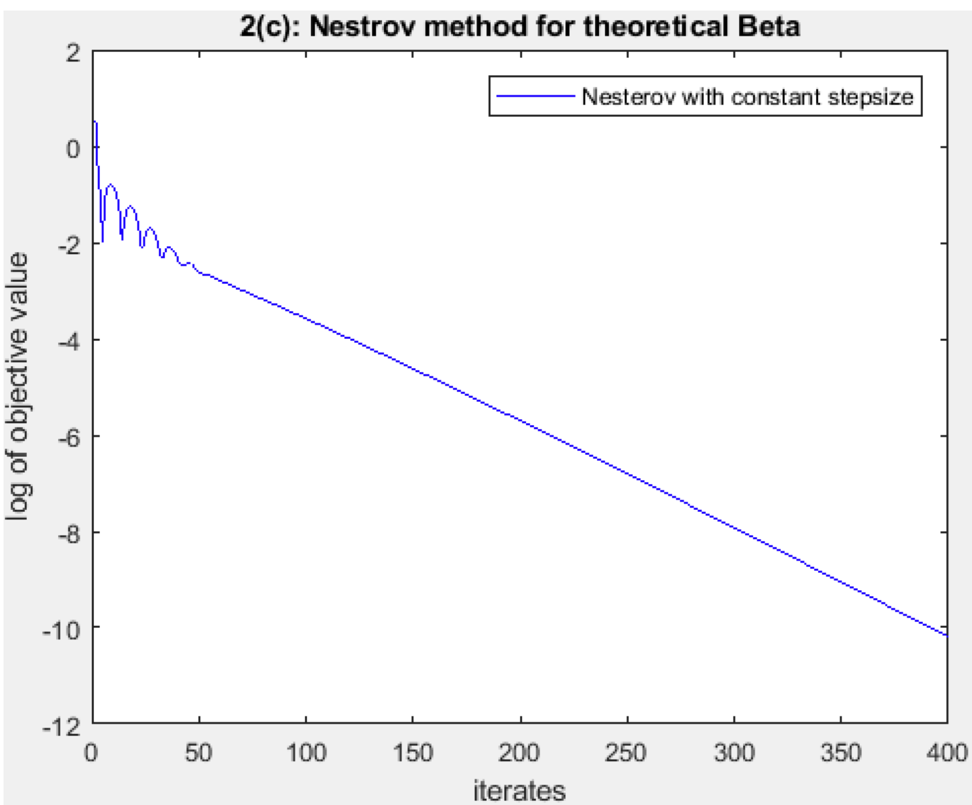
gradient norm vs. cputime

2. (C)






2(c): Nestrov method for Beta = 0.95
Nesterov with constant stepsize
log of objective value
iterates
1
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
0
50
100
150
200
250
300
350
400


2(c): Nestrov method for theoretical Beta
Nesterov with constant stepsize
log of objective value
iterates
2
0
-2
-4
-6
-8
-10
-12
0
50
100
150
200
250
300
350
400

Code: 1.(b)

```
%% Set up Problem
x = [2; 3];
x = [2; -3];
y = [1; -1];
max_it = 20000;
w = 1;
L = 1/8*(x(1)^2+x(2)^2);
alpha = 1/L;
ite = 1;

%% Set up function value
cur_func = 1/2*(log(1+exp(-x(1)*y(1)*w))+log(1+exp(-x(2)*y(2)*w)));
func_value = zeros(max_it +1, 1);
func_value(1) = log10(cur_func);

%% run the algorithm
while (ite <= max_it)
    z1 = exp(-y(1)*x(1)*w);
    z2 = exp(-y(2)*x(2)*w);
    term1 = -y(1)*x(1)*z1/(1+z1);
    term2 = -y(2)*x(2)*z2/(1+z2);
    gradient = 1/2*(term1+term2);
    new_w = w - alpha*gradient;
    w = new_w;
    func = 1/2*(log(1+exp(-x(1)*y(1)*w))+log(1+exp(-x(2)*y(2)*w)));
    func_value(ite+1) = log10(func);
    %func_value(ite+1) = log10(norm(gradient));
    ite = ite+1;
end

%% Plot the function & gradient norm
figure;
plot_length = max_it ;
plot_vec = 0:1:plot_length-1;
plot(plot_vec, func_value(1:plot_length), 'b-');
xlabel('iteration');
ylabel('log of objective value');
legend('GD with constant stepsize');
title('1(ii): function value vs. iteration');
%title('1(ii): gradient norm vs. iteration');
```

1. (d). GD

```matlab
%% Set up Problem
n = 50;
d = 10;
x = rand(n,d)-0.5;
y1 = binornd(1,0.5, n, 1);
y1(y1==0)=-1;

w = ones(d,1);
y2 = x*w;
y2(y2<0)=-1;
y2(y2>=0)=1;
max_it = 100;

%% Set up gradient norm & cputime
gradient_norm = zeros(max_it +1, 1);
time = zeros(max_it +1, 1);

%% run the algorithm
t = cputime;
L = 1/(4*n)*max(eig(x'*x));
alpha = 1/L;
ite = 1;
while (ite <= max_it)
    sum = 0;
    for i = 1:n
        z = exp(-y1(i)*w'*x(i,:)');
        g = -y1(i)*x(i,:)'*z/(1+z);
        %z = exp(-y2(i)*w'*x(i,:)');
        %g = -y2(i)*x(i,:)'*z/(1+z);
        sum = sum + g;
    end
    gradient = sum/n;
    w = w - alpha*gradient;
    gradient_norm(ite+1) = log10(norm(gradient));
    time(ite+1) = cputime - t;
    ite = ite+1;
end

%% Plot the function
figure;
plot_length = max_it ;
plot_vec = 0:1:plot_length-1;
plot(plot_vec, gradient_norm(1:plot_length), 'b-');
plot(time(1:plot_length), gradient_norm(1:plot_length),'b-');

%% Plot the iterates
xlabel('iterations');
xlabel('cputime');
ylabel('log of objective value');
%title('1(d): GD objective value vs. iterations');
title('1(d): GD objective value vs. cputime');
```

1.(e) HB

```
%% Set up Problem
n = 50;
d = 10;
x = rand(n,d)-0.5;
y1 = binornd(1,0.5, n, 1);
y1(y1==0)=-1;
w = ones(d,1);
max_it = 100;

%% Set up gradient norm & cputime
sum = 0;
for i = 1:n
    z = exp(-y1(i)*w'*x(i,:)');
    g = -y1(i)*x(i,:)'*z/(1+z);
    sum = sum + g;
end
gradient = sum/n;
gradient_norm = zeros(max_it +1, 1);
gradient_norm(1) = log10(norm(gradient));
time = zeros(max_it + 1, 1);
t = cputime;

%% run the algorithm
L = 1/(4*n)*max(eig(x'*x));
alpha = 1/L;
k = max(eig(x'*x))/min(eig(x'*x));
beta = ((k^(1/2)-1)/(k^(1/2)+1))^2;
ite = 1;
new_w = w - alpha*gradient;
old_w = 0;
time(1) = cputime - t;

while (ite <= max_it)
    old_w = w;
    w = new_w;
    sum = 0;
    for i = 1:n
        z = exp(-y1(i)*w'*x(i,:)');
        g = -y1(i)*x(i,:)'*z/(1+z);
        sum = sum + g;
    end
    gradient = sum/n;
    new_w = w - alpha*gradient + beta * (w - old_w);
    gradient_norm(ite+1) = log10(norm(gradient));
    time(ite+1) = cputime - t;
    ite = ite+1;
end

%% Plot the function
figure;
plot_length = max_it ;
plot_vec = 0:1:plot_length-1;
plot(plot_vec, gradient_norm(1:plot_length), 'b-');
plot(time(1:plot_length), gradient_norm(1:plot_length), 'b-');

%% Plot the iterates
xlabel('cputime');
%xlabel('iterations');
ylabel('log of gradient norm');
%title('1(e): HB gradient norm vs. iterations');
title('1(e): HB gradient norm vs. cputime’);
```

1.(e) Nesterov

```
%% Set up Problem
n = 50;
d = 10;
x = rand(n,d)-0.5;
y1 = binornd(1,0.5, n, 1);
y1(y1==0)=-1;
w = ones(d,1);
max_it = 100;

%% Set up gradient norm & cputime
sum = 0;
for i = 1:n
    z = exp(-y1(i)*w'*x(i,:)');
    g = -y1(i)*x(i,:)'*z/(1+z);
    sum = sum + g;
end
gradient = sum/n;
gradient_norm = zeros(max_it + 1, 1);
gradient_norm(1) = log10(norm(gradient));
time = zeros(max_it +1, 1);
t = cputime;

%% run the algorithm
L = 1/(4*n)*max(eig(x'*x));
alpha = 1/L;
k = max(eig(x'*x))/min(eig(x'*x));
beta = (1-1/k^(1/2))^2;
ite = 1;
new_w = w - alpha*gradient;
old_w = 0;
time(1) = cputime - t;
while (ite <= max_it)
    old_w = w;
    w = new_w;
    sum = 0;
    yr = w +  beta * (w - old_w);
    for i = 1:n
        z = exp(-y1(i)*yr'*x(i,:)');
        g = -y1(i)*x(i,:)'*z/(1+z);
        sum = sum + g;
    end
    gradient = sum/n;
    new_w = yr - alpha*gradient;
    gradient_norm(ite+1) = log10(norm(gradient));
    ite = ite+1;
    time(ite+1) = cputime - t;
end
%% Plot the function
figure;
plot_length = max_it ;
plot_vec = 0:1:plot_length-1;
plot(plot_vec, gradient_norm(1:plot_length), 'b-');
plot(time(1:plot_length), gradient_norm(1:plot_length), 'b-');
%% Plot the iterates
xlabel('cputime');
ylabel('log of gradient norm');
title('1(e): Nestrov gradient norm vs. cputime');
```

2.(c)

```
%% setup problem
x = [1;1;1];
new_x = x;
v = [3.5,0.4,0.01];
Q = diag(v);
L = max(eig(Q));

alpha = 1/L;
k = L/min(eig(Q));
beta = 1-2/(k^(1/2)+1);
%beta = 0.95;
gradient_norm = zeros(max_it +1 , 1);

max_it = 400;
ite = 1;

%% run algorithm
while (ite <= max_it)
    old_x = x;
    x = new_x;
    yr = x +  beta * (x - old_x);
    gradient = Q*yr;
    new_x = yr - alpha*gradient;
    gradient_norm(ite) = log10(norm(gradient));
    ite = ite+1;
end

%% Plot the iterates
figure;
plot_length = max_it ;
plot_vec = 1:1:plot_length;
plot(plot_vec, gradient_norm(1:plot_length), 'b-');
xlabel('iterates');
ylabel('log of objective value');
legend('Nesterov with constant stepsize');
title('2(c): Nestrov method for theoretical Beta');
```