# ELEC 531 HW3-A:QuadraticMinimization

Ender Erkaya

November 7, 2021

## Contents



## Listings

### Quadratic Minimization

```
clearvars
```

### a) Gradient of the Function

$$\nabla f = -A^T(b - Ax) - A^T(b - Ax) = -2A^T(b - Ax)$$
$$\nabla^2 f = 2A^T A$$

```
load("Question1Data.mat")
x_solution = (A'*A)\(A'*b);
obj_value  = norm(b-A*x_solution)^2;
```

### b) Condition Parameters

```
n = size(A,2);
eigenvalues_A = eig(2*A'*A);
L = max(eigenvalues_A); % Lipschitz constant
m = min(eigenvalues_A); % strong convexity
```

$A'A$ operates onto $R^2$. It has 2 eigenvalues: {688,5700}

The function is strongly convex since with strong convexity parameter

$$m = \lambda_{min}(\nabla^2 f) = 688 > 0$$

The function is smooth(bounded curvature), ie L-Lipschitz continuous gradient with Lipschitz constant

$$L = \lambda_{max}(\nabla^2 f) = 5700$$

## c) Step Size Selection

```
mu = 2/(L+m);
```

Reference: Theorem 3.12 p 279, Convex Optimization: Algorithms and Complexity, Bubeck Step size is chosen as $\mu = 2/(L+m) = 3.13e-4$. The bound $1/L \leq \mu \leq min\{2/L, 1/m\}$ ensures linear convergence, utilizes quadratic upper bounds stated in the lectures, ie:

$$f(x^{(i)}) - f^* \leq c^k \frac{L}{2} ||(x^1 - x^*)||^2{}_2$$

Using chosen $\mu$, the update rule is given as.

$$x^{(i+1)} = x - \mu \nabla(f(x^{(i)}))$$

## d) Gradient Descent with Constant Step Size

```
num_of_iter          = 100;
x_constant           = zeros(n,num_of_iter+1);
objective_constant = zeros(num_of_iter+1,1);
x_initial            = randn(n,1);

% Initializations
x_constant(:,1)          = x_initial;
objective_constant(1,1) = norm(b-A*x_constant(:,1))^2;
mu_constant              = mu;
% Gradient Descent with Constant Step Size
for it = 1:num_of_iter
    x_constant(:,it+1)          = x_constant(:,it) - mu_constant * (2*A
        '*(A*x_constant(:,it)-b));
    objective_constant(it+1,1) = norm(b - A * x_constant(:,it+1))^2;
end
```

## e) Armijo's Rule Line Search Gradient Descent

Initializations

```
x_backtracking              = zeros(n,num_of_iter+1);
objective_backtracking      = zeros(num_of_iter+1,1);
x_backtracking(:,1)         = x_initial;
objective_backtracking(1,1)= norm(b-A*x_backtracking(:,1))^2;
beta                        = 1/2;
gamma                       = 1e-3;
iteration                   = zeros(num_of_iter,1);
% Gradient Descent with Backtracking Line Search aka Armijo's Rule
% Gradient Descent with Constant Step Size
for it = 1:num_of_iter
    % Line Search
```

```
    temp_gradient = (2 * A' * (A*x_backtracking(:,it) - b));
    tau           = 1;
    while 1
        iteration(it,1) = iteration(it,1) + 1;
        x_temp = x_backtracking(:,it)-tau*temp_gradient;
        if ((norm(b-A*x_temp)^2)-objective_backtracking(it,1)) < -gamma
            * tau * norm(temp_gradient)^2
            break
        end
        tau = tau * beta;
    end
    x_backtracking(:,it+1) = x_backtracking(:,it) - tau * temp_gradient
       ;
    objective_backtracking(it+1,1) = norm(b-A*x_backtracking(:,it+1))
       ^2;
end
```

## f) Nesterov's Accelerated Gradient Descent

Initializations

```
x_nesterov                = zeros(n,num_of_iter+1);
y_nesterov                = zeros(n,num_of_iter+1);
objective_nesterov        = zeros(num_of_iter+1,1);
x_nesterov(:,1)           = x_initial;
objective_nesterov(1,1)= norm(b-A*x_nesterov(:,1))^2;
K                         = L/m ; % condition number
gamma                     = (1-sqrt(K))/(sqrt(K)+1);
% reference:
% https://blogs.princeton.edu/imabandit/2014/03/06/nesterovs-
   accelerated-gradient-descent-for-smooth-and-strongly-convex-
   optimization/
% Gradient Descent Updates via Nesterov's Accelerated Approach
for it = 1:num_of_iter
    temp_gradient  = (2 * A' * (A*x_nesterov(:,it) - b));
    y_nesterov(:,it+1) = x_nesterov(:,it) - (1/L) * temp_gradient;
    x_nesterov(:,it+1) = (1 - gamma) * y_nesterov(:,it+1) + gamma *
       y_nesterov(:,it);
    objective_nesterov(it+1,1) = norm(b-A*x_nesterov(:,it+1))^2;
end
```

## g) Visualization of Convergence

```
data_length1 = linspace(min(x_constant(1,:))-0.2,max(x_constant(1,:))
   +0.2,100);
data_length2 = linspace(min(x_constant(2,:))-0.2,max(x_constant(2,:))
   +0.2,100);
[X1,X2]      = meshgrid(data_length1,data_length2);
Z            = zeros(size(X1));
for i = 1:size(X1,1)
    for j = 1:size(X1,2)
        Z(i,j)= norm(b-A*[X1(i,j) ; X2(i,j)])^2;
    end
end

figure
```

```
contour(X1,X2,Z,30,'--');
hold on
plot(x_constant(1,:),x_constant(2,:),'-o');
hold on
plot(x_backtracking(1,:),x_backtracking(2,:),'-sr')
hold on
plot(x_nesterov(1,:),x_nesterov(2,:),'-dm')
grid on
legend('Objective Contour','Constant Step Size','Backtracking Line
    Search','Nesterov','Interpreter','Latex','Location','northwest');
xlabel('$x_1$','Interpreter','latex')
ylabel('$x_2$','Interpreter','latex')
title('Search Vector Convergence')
```

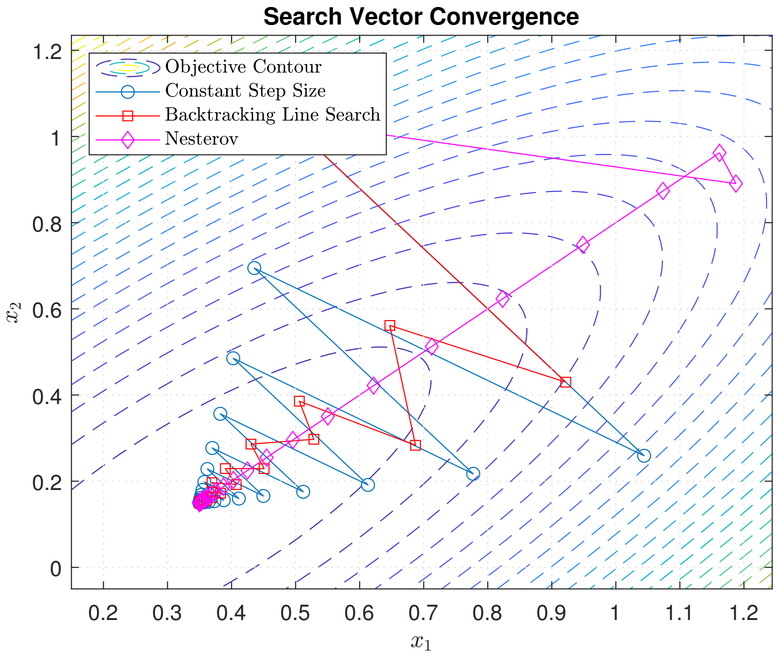


Figure 1:

## h) Convergence of Objective Functions Plots

```
figure
semilogy(0:num_of_iter,objective_constant-obj_value);
hold on
semilogy(0:num_of_iter,objective_backtracking-obj_value,'r');
hold on
semilogy(0:num_of_iter,objective_nesterov-obj_value,'m');
grid on
xlabel('number of iterations','Interpreter','latex');
ylabel('$f(x^{(k)})-f^*$','Interpreter','latex');
legend('Constant Step Size','Backtracking','Nesterov','Interpreter','
    latex');
title('Objective Value Convergence')
```

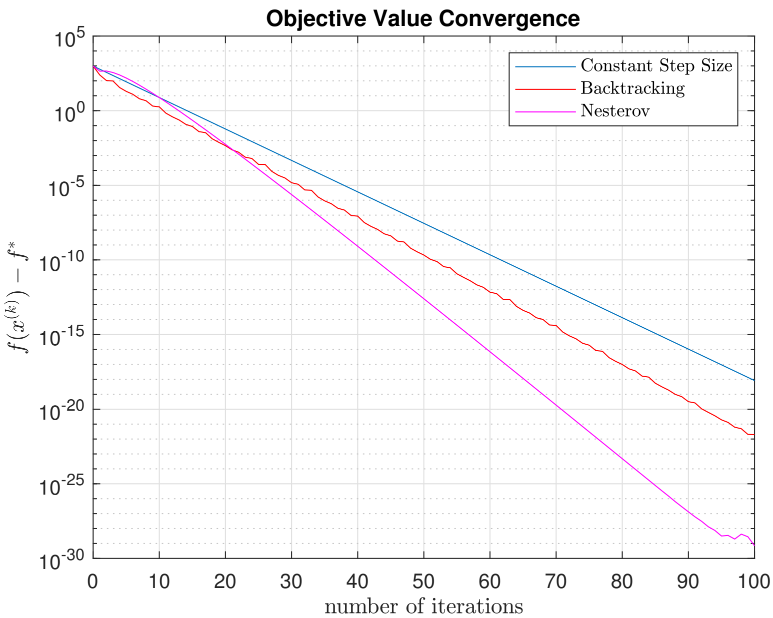
Objective Value Convergence
Constant Step Size
Backtracking
Nesterov
$f(x^{(k)}) - f^*$
number of iterations
10^5
10^0
10^-5
10^-10
10^-15
10^-20
10^-25
10^-30
0
10
20
30
40
50
60
70
80
90
100

Figure 2: