AE 551: Introduction to Optimal Control **(Due: 2020/05/22)**

Homework #8 Submission

Instructor: Min-Jea Tahk *Name:* Joshua Julian Damanik, *Student ID:* 20194701

Main code for problem 1 and 2 is attached at the Appendix section. However the rest of the code can be accessed from https://github.com/joshuadamanik/Homework-8.

Problem 1: Primal-Dual Method for Equality Constraints

$$\min \quad f(x_1, x_2) = \frac{1}{2}(x_1 - 1)^2 + 10(x_2 - 1)^2 \tag{1}$$

$$\text{subject to} \quad c(x_1, x_2) = (x_1 - 2)^2 + 2 - x_2 = 0 \tag{2}$$

(Solution)

To solve the optimization problem on equation 1, Primal-Dual method for equality constraint is used. It defines a dual function as

$$\theta(\tau) = \min_x f(X) + v(\tau)^T c(X), \quad \tau > 0 \tag{3}$$

where $X = (x_1, x_2)^T$. The perturbed KKT condictions for barrier problem (KKT-P) is defined as

$$\nabla f(\bar{X}(\tau)) + v^*(\tau)^T \nabla c(\bar{X}(\tau)) = 0 \quad \text{(stationarity)} \tag{4}$$

$$c_i(\bar{X}(\tau)) = 0, \quad \forall i \quad \text{(primal feasibility)} \tag{5}$$

.

Let $z = (X^T, v)^T$ and residual $r(z)$ as

$$r(z) = \begin{bmatrix} r_{dual} \\ r_{primal} \end{bmatrix} = \begin{bmatrix} \nabla f(X) + v(\tau)^T \nabla c(X) \\ c(X) \end{bmatrix} \tag{6}$$

We aim to minimize the residual $r(z)$ by finding Δz that satisfy

$$r(z + \Delta z) \approx r(z) + \left[\frac{\partial r}{\partial z}\right] \Delta z = 0 \tag{7}$$

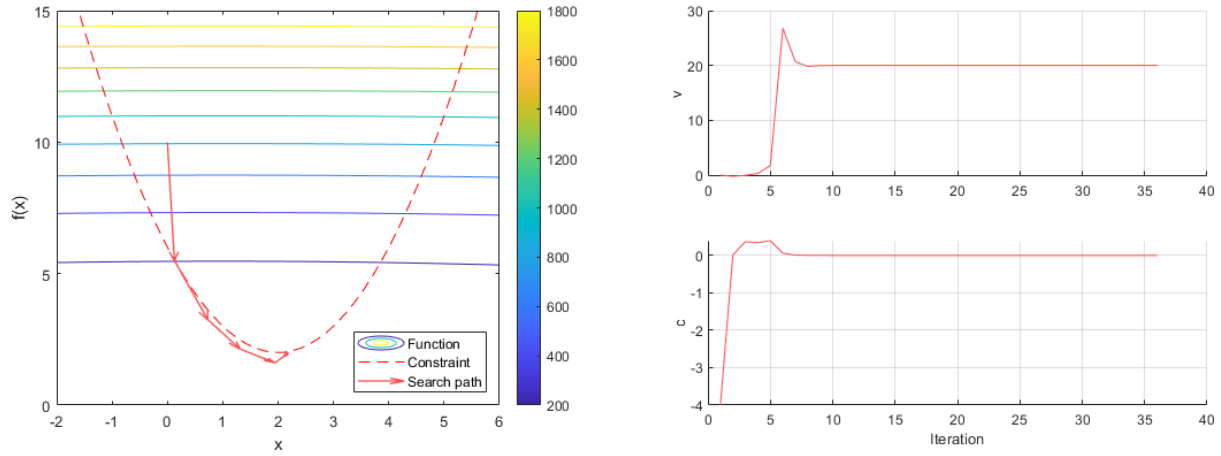
where

$$\left[\frac{\partial r}{\partial z}\right] = \begin{bmatrix} \nabla^2 f(X) + v^T \nabla^2 c(X) & \nabla c(X) \\ \nabla c(X)^T & 0 \end{bmatrix} = S \tag{8}$$

Then, we can solve the equation 18 by solving

$$\begin{aligned} S\Delta z &= -r(z) \\ S^T S \Delta z &= -S^T r(z) \\ \Delta z &= -[S^T S]^{-1} S^T r(z) \end{aligned} \tag{9}$$

In the program, the variable z is calculated iteratively with $z_{k+1} = z_k + \Delta z_k$. The gradient and hessian of of $f(X)$ and $c(X)$ is calculated using central difference method. The result of the simulation is shown at figure 1. Figure 1a shows the path of the search which is initialized at $z_0 = [x_1, x_2, v]_0^T = [0, 10, 0]^T$. The search took 36 iterations to reach final value $z_f = [x_1, x_2, v]_f^T = [1.9756, 2.0006, 20.0119]^T$. Figure 1b shows the value of v_k and $c(X_k)$ at each iteration. By using primal-dual method, the primal variable X and dual variable v is solved simultaneously. As shown at the result, at the end of iteration, both X and v converged to a constant value.



(a) Search path

(b) v_k and $c(X_k)$ value at each iteration

Figure 1: Simulation result of problem 1

Problem 2: Primal-DUal Method with Barrier Function for Inequality Constraints

$$\min \quad f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2 \tag{10}$$

$$\text{subject to} \quad d(x_1, x_2) = (1 + x_1)^2 - x_2 \leq 0 \tag{11}$$

To solve the optimization problem on equation 1, Primal-Dual method with barrier function for equality constraint is used. It defines a dual function as

$$\theta(\tau) = \min_x f(X) + u(\tau)^T d(X), \quad \tau > 0 \tag{12}$$

where $X = (x_1, x_2)^T$. The perturbed KKT condictions for barrier problem (KKT-P) is defined as

$$\nabla f(\bar{X}(\tau)) + \sum_{i=1}^{m} u_i^*(\tau)^T \nabla d_i(\bar{X}(\tau)) = 0 \quad \text{(stationarity)} \tag{13}$$

$$d_i(\bar{X}(\tau)) < 0, \quad \forall i \quad \text{(primal feasibility)} \tag{14}$$

$$u_i^* > 0, \quad \forall i \quad \text{(dual feasibility)} \tag{15}$$

$$u_i^* d_i(\bar{X}(\tau)) = -1/\tau = -\rho, \quad \forall i \quad \text{(complementary slackness)} \tag{16}$$

.

Let $z = (X^T, v)^T$ and residual $r(z)$ as

$$r(z) = \begin{bmatrix} r_{dual} \\ r_{central} \end{bmatrix} = \begin{bmatrix} \nabla f(X) + u(\tau)^T \nabla d(X) \\ diag(u)d(X) + \rho I \end{bmatrix} \tag{17}$$

We aim to minimize the residual $r(z)$ by finding Δz that satisfy

$$r(z + \Delta z) \approx r(z) + \begin{bmatrix} \frac{\partial r}{\partial z} \end{bmatrix} \Delta z = 0 \tag{18}$$

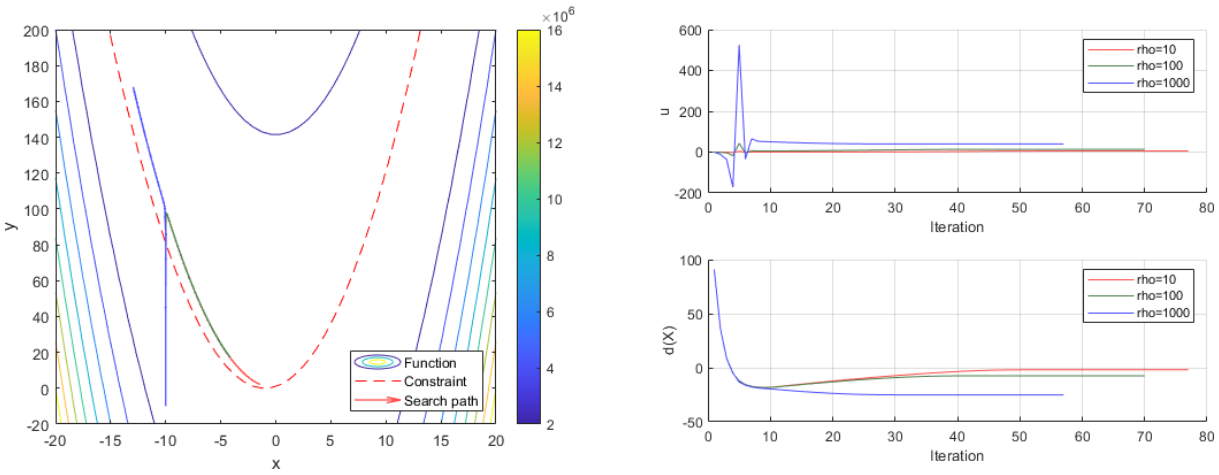
where

$$\begin{bmatrix} \frac{\partial r}{\partial z} \end{bmatrix} = \begin{bmatrix} \nabla^2 f(X) + u^T \nabla^2 d(X) & \nabla d(X) \\ diag(u)\nabla d(X)^T & diag(d(X)) \end{bmatrix} = S \tag{19}$$

Then, we can solve the equation 18 by solving

$$\begin{aligned} S\Delta z &= -r(z) \\ S^T S \Delta z &= -S^T r(z) \\ \Delta z &= -[S^T S]^{-1} S^T r(z) \end{aligned} \tag{20}$$

In the program, the variable z is calculated iteratively with $z_{k+1} = z_k + \Delta z_k$. The gradient and hessian of of $f(X)$ and $d(X)$ is calculated using central difference method. The result of the simulation is shown at figure 2. Figure 2a shows the path of the search which is initialized at $z_0 = [x_1, x_2, v]_0^T = [-10, -10, 0]^T$. The search took 77, 70, and 57 iterations for $rho = \{10, 100, 1000\}$ respectively. Figure 2b shows the value of u_k and $d(X_k)$ at each iteration. By using primal-dual method, the primal variable X and dual variable u is solved simultaneously. As shown at the result, at the end of iteration, both X and u converged to a constant value.



(a) Search path

(b) u_k and $d(X_k)$ value at each iteration

Figure 2: Simulation result of problem 1

Appendix

Listing 1: Main code for problem 1

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% HOMEWORK #8
% Joshua Julian Damanik (20194701)
% AE551 – Introduction to Optimal Control
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear, clc, close all;
addpath('lib');

eps = 0.1;
k = 15;

rho_list = [10, 100, 1000];
delta_rho = 2;

iter = 0;

X_init = [0; 10];
v_init = 0;

f = @(X) (0.5*(X(1)-1).^2+10*(X(2)-1).^2);
c = @(X) (X(1)-2)^2+2-X(2);

X = X_init;
v = v_init;

X_data = X;
v_data = v;
c_data = c(X);

for ii = 2:1000
    iter = iter + 1;

    L = @(X, v) (f(X) + v'*c(X));
    L_X = @(X) L(X, v);
    L_v = @(v) L(X, v);

    grad_L_X = grad_central_diff(X, eps, L_X);
    hess_L_X = hess_central_diff(X, eps, L_X);
    grad_c = grad_central_diff(X, eps, c);

    S = [hess_L_X, grad_c; grad_c', 0];
    r = [grad_L_X; c(X)];
    del_r = -pinv(S)*r;

    X = X + del_r(1:length(X));
    v = v + del_r(length(X)+1:end);

    X_data(:,ii) = X;
    v_data(:,ii) = v;
    c_data(:,ii) = c(X);

    fprintf('ii=%d\n',ii);

    if norm(X_data(:,ii)-X_data(:,ii-1))/norm(X_data(:,ii)) < 1e-10
        break;
    end
end

%% Function Graph
```

```
N_grid = 50;
axis_xy = [-2 6 0 15];
x_cont = linspace(axis_xy(1), axis_xy(2), N_grid);
y_cont = linspace(axis_xy(3), axis_xy(4), N_grid);
[X_cont, Y_cont] = meshgrid(x_cont, y_cont);

F_cont = zeros(N_grid);
Cy_cont = zeros(N_grid);

for i=1:N_grid
    for j=1:N_grid
        F_cont(i,j) = f([X_cont(i,j), Y_cont(i,j)]');
    end
end

figure(1);
s = contour(X_cont, Y_cont, F_cont);
colorbar;
hold on;

%% Constraint Graph

C_data = (x_cont-2).^2+2;
plot(x_cont, C_data, 'r--');
axis(axis_xy);

%% Search Path Graph
color = [1, 0.3, 0.3;
         0.3, 0.5, 0.3;
         0.3, 0.3, 1];
% for j=1:3
    points = X_data;
    for i=1:size(points,2)-1
        F_data = f(points(:,i));
        qlen = [points(:,i+1) - points(:,i)];% f(X_data(:,i+1))-F_data];
        quiver(points(1,i), points(2,i), ...% F_data, ...
                    qlen(1), qlen(2), ... % qlen(3), ...
                    'r', 'AutoScale', 'off', 'LineWidth', 1, ...
                    'MaxHeadSize', min(1 / norm(qlen),1), ...
                    'color', color(1,:));
    end
% end
xlabel('x');
ylabel('f(x)');
zlabel('z');
legend('Function', 'Constraint', 'Search path', 'Location', 'SouthEast');

%% v Graph
figure(2); subplot(2,1,1); hold on;
% for j=1:3
    p=plot(1:length(v_data), v_data, 'Color', color(1,:));
% end
grid on;
ylabel('v');

%% c Graph
subplot(2,1,2); hold on;
% for j=1:3
    p=plot(1:length(c_data), c_data, 'Color', color(1,:));
% end
grid on;
xlabel('Iteration');
```

```
ylabel('c');
```

Listing 2: Main code for problem 2

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% HOMEWORK #8
% Joshua Julian Damanik (20194701)
% AE551 - Introduction to Optimal Control
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear, clc, close all;
addpath('lib');

eps = 0.1;
k = 15;

rho_list = [10, 100, 1000];
delta_rho = 2;

iter = 0;

X_init = [-10, -10]';
u_init = 0;

f = @(X) (1-X(1)).^2+100*(X(2)-X(1).^2).^2;
d = @(X) ((1+X(1)).^2-X(2));


for jj = 1:3
    X = X_init;
    u = u_init;

    X_data{jj} = X;
    u_data{jj} = u;
    f_data{jj} = f(X);
    d_data{jj} = d(X);

    rho = rho_list(jj);
    for ii = 2:1000
        iter = iter + 1;

        L = @(X, u) (f(X) + u'*d(X));
        L_X = @(X) L(X, u);

        grad_L_X = grad_central_diff(X, eps, L_X);
        hess_L_X = hess_central_diff(X, eps, L_X);
        grad_d = grad_central_diff(X, eps, d);

        S = [hess_L_X, grad_d; diag(u)*grad_d', ones(length(u))*d(X)];
        r = [grad_L_X; diag(u)*d(X) + ones(length(u),1)*rho];
        del_r = -pinv(S)*r;

        X = X + del_r(1:length(X));
        u = u + del_r(length(X)+1:end);

        X_data{jj}(:,ii) = X;
        u_data{jj}(:,ii) = u;
        f_data{jj}(:,ii) = f(X);
        d_data{jj}(:,ii) = d(X);

        fprintf('jj=%d,ii=%d\n',jj,ii);

        if norm(X_data{jj}(:,ii)-X_data{jj}(:,ii-1))/norm(X_data{jj}(:,ii)) < 1e-10
```

```
            break;
        end
    end
end

%% Function Graph

N_grid = 50;
axis_xy = [-20 20 -20 200];
x_cont = linspace(axis_xy(1), axis_xy(2), N_grid);
y_cont = linspace(axis_xy(3), axis_xy(4), N_grid);
[X_cont, Y_cont] = meshgrid(x_cont, y_cont);

F_cont = zeros(N_grid);
Cy_cont = zeros(N_grid);

for i=1:N_grid
    for j=1:N_grid
        F_cont(i,j) = f([X_cont(i,j), Y_cont(i,j)]');
    end
end

figure(1);
s = contour(X_cont, Y_cont, F_cont);
colorbar;
hold on;

%% Constraint Graph

C_data = (1+x_cont).^2;
plot(x_cont, C_data, 'r--');
axis(axis_xy);

%% Search Path Graph
color = [1, 0.3, 0.3;
         0.3, 0.5, 0.3;
         0.3, 0.3, 1];
for j=1:3
    points = X_data{j};
    for i=1:size(points,2)-1
        F_data = f(points(:,i));
        qlen = [points(:,i+1) - points(:,i)];% f(X_data(:,i+1))-F_data];
        quiver(points(1,i), points(2,i), ...% F_data, ...
                    qlen(1), qlen(2), ... % qlen(3), ...
                    'r', 'AutoScale', 'off', 'LineWidth', 1, ...
                    'MaxHeadSize', min(1 / norm(qlen),1), ...
                    'color', color(j,:));
    end
end
xlabel('x');
ylabel('y');
zlabel('z');
legend('Function', 'Constraint', 'Search path', 'Location', 'SouthEast');

%% Mu Graph
figure; subplot(2,1,1); hold on;
for j=1:3
    p=plot(1:length(u_data{j}), u_data{j}, 'Color', color(j,:));
end
grid on;
xlabel('Iteration');
ylabel('u');
legend('rho=10', 'rho=100', 'rho=1000');
```

```
%% f Graph
% figure; hold on;
% for j=1:3
%     p=plot(1:length(f_data{j}), f_data{j}, 'Color', color(j,:));
% end
% grid on;
% xlabel('Iteration');
% ylabel('f(X)');
% legend('rho=10', 'rho=100', 'rho=1000');

%% d Graph
subplot(2,1,2); hold on;
for j=1:3
    p=plot(1:length(d_data{j}), d_data{j}, 'Color', color(j,:));
end
grid on;
xlabel('Iteration');
ylabel('d(X)');
legend('rho=10', 'rho=100', 'rho=1000');
```

Listing 3: Code for function *grad_central_diff*

```
function g = grad_central_diff(X, eps, f)
    dim = length(X);
    E = eye(dim);

    g = zeros(dim, 1);
    for i = 1:dim
        g(i) = (f(X + eps*E(:,i)) - f(X - eps*E(:,i))) / (2*eps);
    end
end
```

Listing 4: Code for function *hess_central_diff*

```
function G = hess_central_diff(X, eps, f)
    gk = grad_central_diff(X, eps, f);
    gkh_xp = grad_central_diff(X + [eps; 0], eps, f);
    gkh_yp = grad_central_diff(X + [0; eps], eps, f);

    gkh_xn = grad_central_diff(X - [eps; 0], eps, f);
    gkh_yn = grad_central_diff(X - [0; eps], eps, f);
    Y = 1/eps * [gkh_xp-gkh_xn, gkh_yp-gkh_yn];
    G = 0.5*[Y+Y'];
end
```