**Arunachalam Ramesh**

MINI PROJECT - 2 MPC

# (Demonstration needs ‘Optimisation Toolbox, communications toolbox’ added in matlab.)

**Part A**

**a)**

**clc; clear; close all;**

**global kf;**

**global g; global mpc;**

**%Kf structure contains all paramets and solutions**

**kf = struct();**

**kf.A1 = 28; %(cm^2)**

**kf.A2 = 32; %(cm^2)**

**kf.A3 = 28; %(cm^2)**

**kf.A4 = 32; %(cm^2)**

**kf.A = [kf.A1, kf.A2, kf.A3, kf.A4]; %continuous form**

**kf.a1 = 0.071; kf.a3 = 0.071; %(cm^2)**

**kf.a2 = 0.057; kf.a4 = 0.057;**

**kf.a = [kf.a1, kf.a2, kf.a3, kf.a4]; kf.kc = 0.5; % (V/cm)**

**g = 981; %(cm/s^2)**

**kf.gamma1 = 0.7; kf.gamma2 = 0.6; kf.k1 = 3.33; kf.k2 = 3.35; %[cm^3/Vs**

**kf.kc = 1; % [V/cm]**

**kf.v1 = 3; kf.v2 = 3; % (V)**

**kf.U = [kf.v1; kf.v2];**

**kf.h0 = [12.4; 12.7; 1.8; 1.4];**

**kf.P\_pr = 1000\*eye(4);**

**kf.Q = 100\*eye(4);**

**kf.R = 10\*eye(2);**

**T = [];**

**kf.x\_pr = [];**

**kf.err=[];**

**% desired outputs for plotting**

**% Finding the value for the term T for all the Elements**

**for j = 1:4**

**T(j) = (kf.A(j)/kf.a(j))\*sqrt(2\*kf.h0(j)/g) ;**

**end**

**% Initializing the Control Input Matrix, State Matrix and Output Matrix**

**kf.Ac = [ -1/T(1), 0, kf.A3/(kf.A1\*T(3)), 0 ; 0, -1/T(2), 0,**

**kf.A4/(kf.A2\*T(4)); 0, 0, -1/T(3), 0; 0, 0, 0, -1/T(4)];**

**kf.Bc = [kf.gamma1\*kf.k1/kf.A1 0 ; 0 kf.gamma2\*kf.k2/ kf.A2; 0 (1 -**

**kf.gamma2)\*kf.k2/kf.A3; (1-kf.gamma1)\*kf.k1/kf.A4 0];**

**kf.Dc = 0;**

**kf.Hc = [kf.kc 0 0 0; 0 kf.kc 0 0];**

**kf.Hcc = [0 0 kf.kc 0; 0 0 0 kf.kc];**

**kf.x\_po(:,1) = kf.h0; %Posterior x initialise kf.x\_po(:,2) = kf.h0;**

**kf.Y = zeros(2,1); kf.Zest = zeros(2,1);**

**state\_space = ss(kf.Ac, kf.Bc, kf.Hc, kf.Dc); %discretizing step state\_space\_discrete = c2d(state\_space, 0.1);**

**kf.Ad = state\_space\_discrete.A; kf.Bd = state\_space\_discrete.B; kf.Hd = state\_space\_discrete.C; kf.Dd = state\_space\_discrete.D; mpc = struct();**

**mpc.Np = 50; mpc.Nc = 30 ;**

**mpc.A = [kf.Ad, zeros(4,2);kf.Hcc\*kf.Ad, eye(2)]; mpc.B = [kf.Bd; kf.Hcc\*kf.Bd];**

**mpc.C = [zeros(2,4), eye(2)]; mpc.F = mpc.C\*mpc.A; mpc.reference = [2.8; 2.4]; mpc.R = 0.5\*eye(2\*mpc.Nc); mpc.U = [kf.v1; kf.v2]; mpc.F = mpc.C\*mpc.A;**

**dims1 = size(kf.Hcc); mpc.q = dims1(1); dims2 = size(kf.Bd); mpc.m = dims2(2); mpc.b = [];**

**for n = 2:mpc.Np**

**mpc.F = vertcat(mpc.F, mpc.C\*mpc.A^n); mpc.reference = vertcat(mpc.reference, [2.8; 2.4]);**

**end**

**dim = size(mpc.C\*mpc.B);**

**mpc.phi = horzcat(mpc.C\*mpc.B, zeros(dim(1), dim(2)\*(mpc.Nc-1))); for p = 1:mpc.Np-1**

**filler = mpc.C\*(mpc.A^p)\*mpc.B; for c = 1: p**

**if c+1<=mpc.Nc**

**filler = horzcat(filler, mpc.C\*(mpc.A^(p-c))\*mpc.B);**

**end**

**end**

**if (p+1<=mpc.Nc)**

**for c = 2:mpc.Nc-p**

**filler = horzcat(filler, zeros(dim));**

**end**

**end**

**mpc.phi = vertcat(mpc.phi, filler);**

**end**

**iter = 1000; for i = 1:iter**

**mpc = uncstr(mpc,kf); kf = plant(mpc,kf); kf = Kalman(kf,mpc);**

**end**

**figure(1)**

**plot(kf.x\_po(1,:), 'LineWidth', 1); hold on**

**plot(kf.x\_po(2,:), 'LineWidth', 1);**

**plot(kf.x\_po(3,:), 'LineWidth', 1);**

**plot(kf.x\_po(4,:), 'LineWidth', 1); grid on**

**ax = gca; ax.FontSize = 10;**

**legend("H1","H2","H3","H4",'FontSize',6) xlabel("Iteration",'FontWeight','bold','FontSize',10) ylabel("Height (cm)",'FontWeight','bold','FontSize',10) title("Heights", 'FontWeight', 'bold','FontSize',10) axis tight**

**ylim([0 15]) hold off figure(2)**

**plot(kf.err(1,:), 'LineWidth', 1,'Color','b'); hold on**

**plot(kf.err(2,:), 'LineWidth',1, 'Color','r'); grid on**

**ax = gca; ax.FontSize = 10;**

**legend("delH3","delH4",'FontSize',6) xlabel("Iteration",'FontWeight','bold','FontSize',10) ylabel("Error (cm)",'FontWeight','bold','FontSize',10) title("Error from Setpoint", 'FontWeight', 'bold','FontSize',10) axis tight**

**hold off**

**function kf = priori(kf, mpc) Xpost = kf.x\_po(:,end);**

**Xpr= kf.Ad\*Xpost + kf.Bd \* mpc.U(:,end); kf.P\_pr = kf.Ad \* kf.P\_pr \* kf.Ad' + kf.Q ; kf.x\_pr(:,end+1) = Xpr;**

**end**

**function kf = Kappa(kf)**

**dr = kf.Hd \* kf.P\_pr \* kf.Hd' + kf.R; nr = kf.P\_pr \* kf.Hd' ;**

**kf.kappa = nr / dr; end**

**function kf = posterior(kf,mpc)**

**kf.x\_po(:,end+1) = kf.x\_pr(:,end) + kf.kappa \* ( kf.Y(:,end) - kf.Hd \* kf.x\_pr(:,end) ) ;**

**kf.P\_pr = (eye(4) - kf.kappa\*kf.Hd)\*kf.P\_pr; kf.Zest(:,end+1) = kf.kc \* kf.x\_po(3:4,end);**

**kf.err(:,end+1) = kf.x\_po(3:4,end) - [2.8;2.4]; end**

**function kf = Kalman(kf,mpc) kf = priori(kf,mpc);**

**kf = Kappa(kf);**

**kf = posterior(kf,mpc); end**

**function mpc = uncstr(mpc,kf) x\_cur = kf.x\_po(:, end); x\_prev = kf.x\_po(:, end-1); del\_x = x\_cur - x\_prev;**

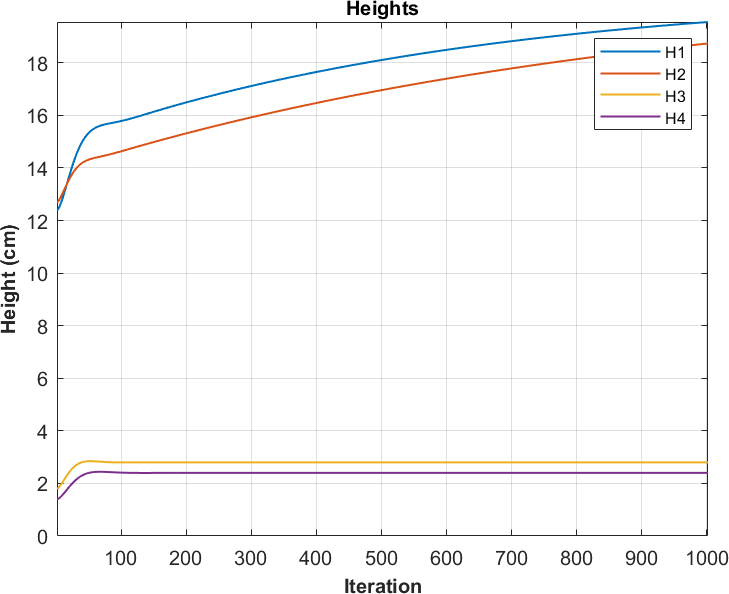
**mpc.x = [del\_x; kf.Zest(:,end)];**

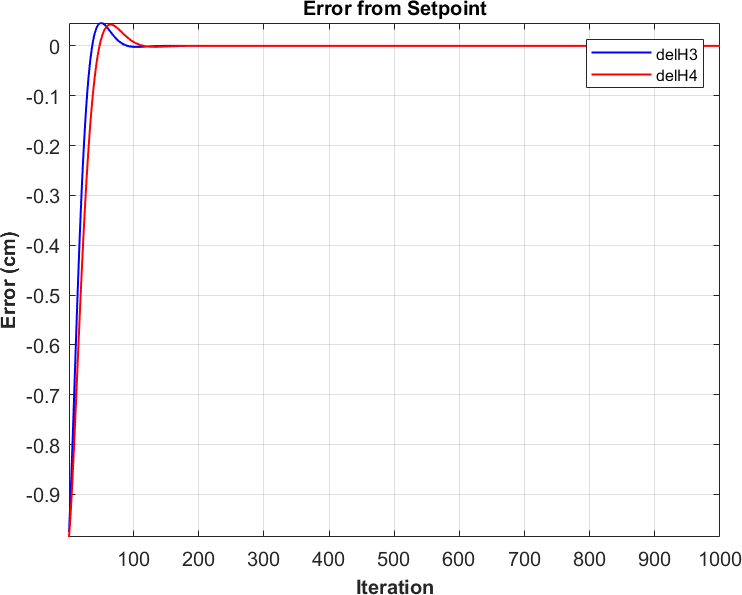
**delta = (mpc.phi' \* mpc.phi + mpc.R)\ (mpc.phi' \* (mpc.reference - mpc.F\*mpc.x));**

**mpc.U(:, end+1) = delta(1:2) + mpc.U(:,end); end**

**function kf = plant(mpc,kf)**

**kf.Y(:,end+1) = kf.Hd \* ( kf.Ad \* kf.x\_po(:,end) + kf.Bd \* mpc.U(:,end)); end**





**b) So the Heuristics made for the given problem because:**

**• The reduce of height is pretty slow process, and hence prediction horizon should be long enough to have**

**enough samples to control the prediction.**

**• It is better to have control horizon half as long as the prediction horizon.**

**• Both should be proportional to the number of samples taken across the time, but at the same time, they**

**shouldn't be so high to make computation capability less.**

**• Weight matrix Q is made to optimize are to priotize the h1 and h2, whereas R is given to minimize the**

**control effort.**

**c) Poles won't move anywhere significantly, since the gain Matrix which makes the closed loop remains constant throughout the process**

**% Closed-loop stability analysis for unconstrained MPC system**

**% Initialize variables for eigenvalue computation**

**mpc.Acl\_first = kf.Ad - kf.Bd \* mpc.U(:, 1);**

**%Closed-loop A at first move**

**mpc.Acl\_stable = kf.Ad - kf.Bd \* mpc.U(:, end); % Closed-loop A at**

**stabilization**

**% Eigenvalues of the closed-loop system at the first move**

**eig\_first = eig(mpc.Acl\_first);**

**% Eigenvalues of the closed-loop system near set-point stabilization**

**eig\_stable = eig(mpc.Acl\_stable);**

**figure;**

**subplot(1, 2, 1);**

**scatter(real(eig\_first), imag(eig\_first), 'filled');**

**title('Poles at First Move');**

**xlabel('Real Part');**

**ylabel('Imaginary Part');**

**grid on;**

**subplot(1, 2, 2);**

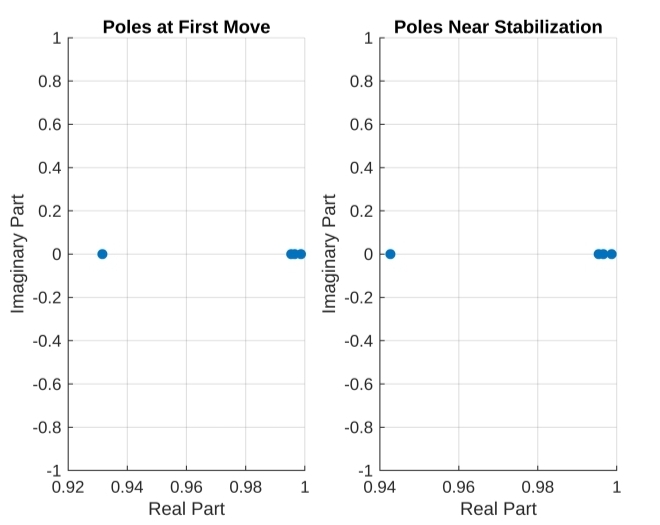
**scatter(real(eig\_stable), imag(eig\_stable), 'filled');**

**title('Poles Near Stabilization');**

**xlabel('Real Part');**

**ylabel('Imaginary Part');**

**grid on;**

****

# Part B

**for p = 1:mpc.Np-1**

**filler = mpc.C\*(mpc.A^p)\*mpc.B; for c = 1: p**

**if c+1<=mpc.Nc**

**filler = horzcat(filler, mpc.C\*(mpc.A^(p-c))\*mpc.B);**

**end**

**end**

**if (p+1<=mpc.Nc)**

**for c = 2:mpc.Nc-p**

**filler = horzcat(filler, zeros(dim));**

**end**

**end**

**mpc.phi = vertcat(mpc.phi, filler);**

**end**

**dim = size(eye(2));**

**mpc.C2 = horzcat(eye(2), zeros(dim(1), dim(2)\*(mpc.Nc-1))); mpc.C1 = eye(2);**

**for p = 1:mpc.Nc-1 filler = eye(2); for c = 1: p**

**if c+1<=mpc.Nc**

**filler = horzcat(filler, eye(2));**

**end**

**end**

**if (p+1<=mpc.Nc)**

**for c = 2:mpc.Nc-p**

**filler = horzcat(filler, zeros(dim));**

**end**

**end**

**mpc.C2 = vertcat(mpc.C2, filler); mpc.C1 = vertcat(mpc.C1, eye(2));**

**end**

**mpc.bcon = [zeros(2\*mpc.Nc, 1) + mpc.C1\*[kf.v1;kf.v2]; 20\*ones(2\*mpc.Nc, 1) - mpc.C1\*[kf.v1; kf.v2]; ones(2\*mpc.Nc,1); 5\*ones(2\*mpc.Nc,1)];**

**mpc.Acon = [-mpc.C2;mpc.C2; -eye(2\*mpc.Nc); eye(2\*mpc.Nc)]; iter = 500;**

**for i = 1:iter**

**mpc = cstr(mpc,kf); kf = plant(mpc,kf); kf = Kalman(kf,mpc);**

**end figure(1)**

**plot(kf.x\_po(1,:), 'LineWidth', 1); hold on**

**plot(kf.x\_po(2,:), 'LineWidth', 1);**

**plot(kf.x\_po(3,:), 'LineWidth', 1);**

**plot(kf.x\_po(4,:), 'LineWidth', 1); grid on**

**ax = gca; ax.FontSize = 10;**

**legend("H1","H2","H3","H4",'FontSize',6) xlabel("Iteration",'FontWeight','bold','FontSize',10) ylabel("Height (cm)",'FontWeight','bold','FontSize',10) title("Heights", 'FontWeight', 'bold','FontSize',10) axis tight**

**ylim([0 15]) hold off figure(2)**

**plot(kf.err(1,:), 'LineWidth', 2); hold on**

**plot(kf.err(2,:), 'LineWidth',2);**

**grid on ax = gca;**

**ax.FontSize = 10; legend("delH3","delH4",'FontSize',6) xlabel("Iteration",'FontWeight','bold','FontSize',10) ylabel("Error (cm)",'FontWeight','bold','FontSize',10)**

**title("Error from Setpoint", 'FontWeight', 'bold','FontSize',10) axis tight**

**hold off**

**function kf = priori(kf, mpc) Xpost = kf.x\_po(:,end);**

**Xpr= kf.Ad\*Xpost + kf.Bd \* mpc.U(:,end); kf.P\_pr = kf.Ad \* kf.P\_pr \* kf.Ad' + kf.Q ; kf.x\_pr(:,end+1) = Xpr;**

**end**

**function kf = Kappa(kf)**

**dr = kf.Hd \* kf.P\_pr \* kf.Hd' + kf.R; nr = kf.P\_pr \* kf.Hd' ;**

**kf.kappa = nr / dr; end**

**function kf = posterior(kf,mpc)**

**kf.x\_po(:,end+1) = kf.x\_pr(:,end) + kf.kappa \* ( kf.Y(:,end) - kf.Hd \* kf.x\_pr(:,end) ) ;**

**kf.P\_pr = (eye(4) - kf.kappa\*kf.Hd)\*kf.P\_pr; kf.Zest(:,end+1) = kf.kc \* kf.x\_po(2:3,end); kf.err(:,end+1) = kf.x\_po(2:3,end) - [13.8;2.4];**

**end**

**function kf = Kalman(kf,mpc) kf = priori(kf,mpc);**

**kf = Kappa(kf);**

**kf = posterior(kf,mpc); end**

**function mpc = uncstr(mpc,kf) x\_cur = kf.x\_po(:, end); x\_prev = kf.x\_po(:, end-1); del\_x = x\_cur - x\_prev;**

**mpc.x = [del\_x; kf.Zest(:,end)];**

**delta = (mpc.phi' \* mpc.phi + mpc.R)\ (mpc.phi' \* (mpc.reference - mpc.F\*mpc.x));**

**mpc.U(:, end+1) = delta(1:2) + mpc.U(:,end); end**

**function mpc = cstr(mpc,kf) x\_cur = kf.x\_po(:, end); x\_prev = kf.x\_po(:, end-1); del\_x = x\_cur - x\_prev;**

**mpc.x = [del\_x; kf.Zest(:,end)]; H = mpc.phi' \* mpc.phi + mpc.R ;**

**F = mpc.phi' \* (mpc.reference -mpc.F \* mpc.x ) ; [mopc,delta] = mp\_controller(mpc);**

**mpc.U(:, end+1) = delta(1:2) + mpc.U(:,end); end**

**function [mopc, u\_opt] = mp\_controller(mpc) cfun = @(x) (mpc.reference-mpc.F\*mpc.x)'\*...**

**(mpc.reference-mpc.F\*mpc.x) + ... x'\*mpc.R\*x;**

**u0 = repmat(mpc.U(:,end), mpc.Nc, 1);**

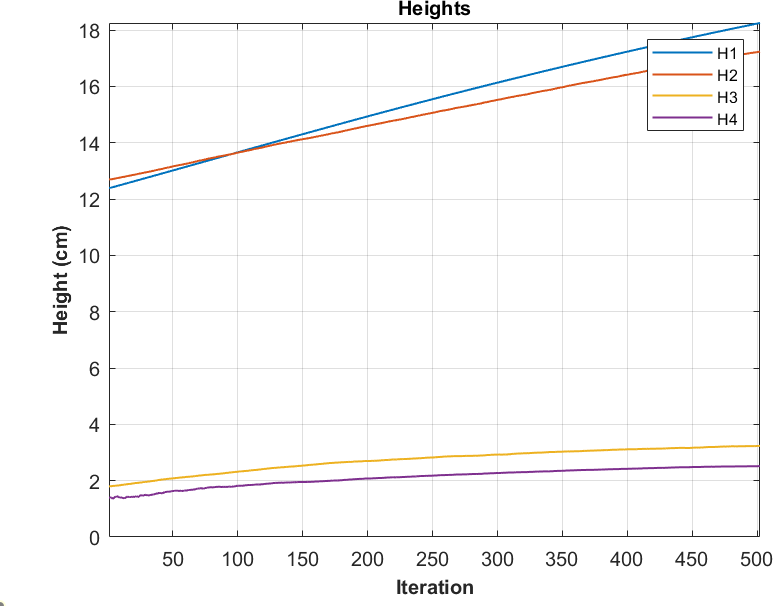
**x = fmincon(cfun, u0 , mpc.Acon, mpc.bcon); u\_opt = x;**

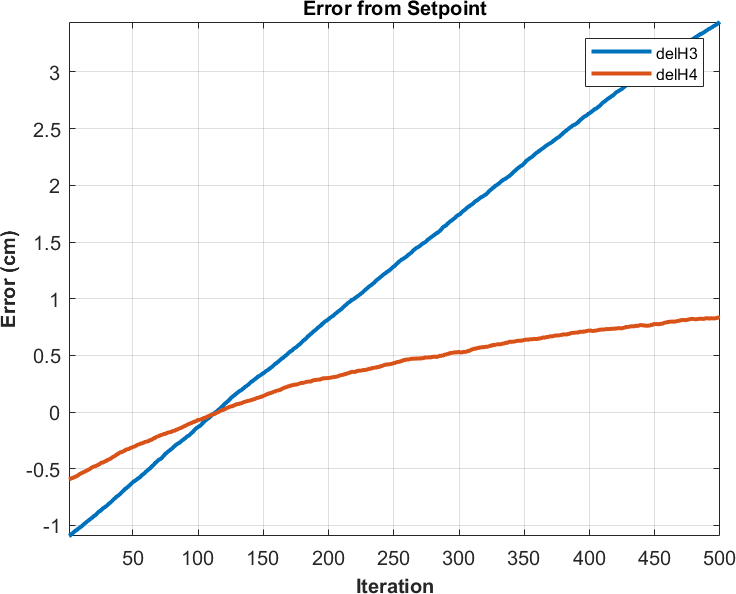
**mopc = mpc; end**

**function kf = plant(mpc,kf)**

**kf.Y(:,end+1) = kf.Hd \* ( kf.Ad \* kf.x\_po(:,end) + kf.Bd \* mpc.U(:,end)); kf.Y(:,end) = awgn(kf.Y(:,end),50);**

**end**





# Part B

b) Finding an optimal values for measurement and process noise goes very crucial for this because:

• Low values of Q, or High values of R can actually reduce the effect of the MPC controller on the output

values, since it gives the fact that model is very much reliable.

• Thus making the response very smooth and slow, which might cause some lag in tracking set point

changes as well.

• Higher Q, or Lower R makes the control input more responsive for small changes in control input, causing

more changes in model input which inturn affects the control input in the opposite direction, making it

oscillatory.

# Part C - a

**clc; clear; close all;**

**% initializing the kf structure which will carry all our parameters and**

**% solutions global kf; global g; global mpc;**

**% parameters kf = struct();**

**kf.A1 = 28; %(cm^2) kf.A2 = 32;**

**kf.A3 = 28;**

**kf.A4 = 32;**

**kf.A = [kf.A1, kf.A2, kf.A3, kf.A4]; kf.a1 = 0.071; kf.a3 = 0.071; %(cm^2) kf.a2 = 0.057; kf.a4 = 0.057;**

**kf.a = [kf.a1, kf.a2, kf.a3, kf.a4]; kf.kc = 0.5; % (V/cm)**

**g = 981; %(cm/s^2)**

**kf.gamma1 = 0.7; kf.gamma2 = 0.6; kf.k1 = 3.33; kf.k2 = 3.35; %[cm^3/Vs] kf.kc = 1; % [V/cm]**

**kf.kc = 1; % [V/cm]**

**kf.v1 = 3; kf.v2 = 3; % (V) kf.U = [kf.v1; kf.v2];**

**kf.h0 = [12.4; 12.7; 1.8; 1.4];**

**kf.P\_pr = 100000\*eye(4); kf.Q = 10\*eye(4); kf.R = 10\*eye(2); T = [];**

**kf.x\_pr = [];**

**kf.err=[];**

**% desired outputs for plotting**

**% Finding the value for the term T for all the Elements for j = 1:4**

**T(j) = (kf.A(j)/kf.a(j))\*sqrt(2\*kf.h0(j)/g) ; end**

**% Initializing the Control Input Matrix, State Matrix and Output Matrix kf.Ac = [ -1/T(1), 0, kf.A3/(kf.A1\*T(3)), 0 ; 0, -1/T(2), 0,**

**kf.A4/(kf.A2\*T(4)); 0, 0, -1/T(3), 0; 0, 0, 0, -1/T(4)];**

**kf.Bc = [kf.gamma1\*kf.k1/kf.A1 0 ; 0 kf.gamma2\*kf.k2/ kf.A2; 0 (1 -**

**kf.gamma2)\*kf.k2/kf.A3; (1-kf.gamma1)\*kf.k1/kf.A4 0];**

**kf.Dc = 0;**

**kf.Hcc = [0 kf.kc 0 0; 0 0 kf.kc 0];**

**kf.Hc = [kf.kc 0 0 0; 0 0 0 kf.kc];**

**% Initializing the array containing posterior values of the States, with**

**% first val being h0 kf.x\_po(:,1) = kf.h0;**

**kf.x\_po(:,2) = kf.h0;**

**kf.Y = zeros(2,1); kf.Zest = zeros(2,1);**

**% changing the Matrices to discrete domain based on step of 0.1 state\_space = ss(kf.Ac, kf.Bc, kf.Hc, kf.Dc); state\_space\_discrete = c2d(state\_space, 0.1);**

**kf.Ad = state\_space\_discrete.A; kf.Bd = state\_space\_discrete.B; kf.Hd = state\_space\_discrete.C; kf.Dd = state\_space\_discrete.D; mpc = struct();**

**mpc.Np = 14; mpc.Nc = 3 ;**

**mpc.A = [kf.Ad, zeros(4,2);kf.Hcc\*kf.Ad, eye(2)]; mpc.B = [kf.Bd; kf.Hcc\*kf.Bd];**

**mpc.C = [zeros(2,4), eye(2)]; mpc.F = mpc.C\*mpc.A; mpc.reference = [13.7; 2.8]; mpc.R = 0.01\*eye(2\*mpc.Nc); mpc.U = [kf.v1; kf.v2]; mpc.F = mpc.C\*mpc.A;**

**dims1 = size(kf.Hcc);**

**mpc.q = dims1(1); dims2 = size(kf.Bd); mpc.m = dims2(2); mpc.b = [];**

**for n = 2:mpc.Np**

**mpc.F = vertcat(mpc.F, mpc.C\*mpc.A^n); mpc.reference = vertcat(mpc.reference, [13.7; 2.8]);**

**end**

**dim = size(mpc.C\*mpc.B);**

**mpc.phi = horzcat(mpc.C\*mpc.B, zeros(dim(1), dim(2)\*(mpc.Nc-1))); for p = 1:mpc.Np-1**

**filler = mpc.C\*(mpc.A^p)\*mpc.B; for c = 1: p**

**if c+1<=mpc.Nc**

**filler = horzcat(filler, mpc.C\*(mpc.A^(p-c))\*mpc.B);**

**end**

**end**

**if (p+1<=mpc.Nc)**

**for c = 2:mpc.Nc-p**

**filler = horzcat(filler, zeros(dim));**

**end**

**end**

**mpc.phi = vertcat(mpc.phi, filler);**

**end**

**dim = size(eye(2));**

**mpc.C2 = horzcat(eye(2), zeros(dim(1), dim(2)\*(mpc.Nc-1))); mpc.C1 = eye(2);**

**for p = 1:mpc.Nc-1 filler = eye(2); for c = 1: p**

**if c+1<=mpc.Nc**

**filler = horzcat(filler, eye(2));**

**end**

**end**

**if (p+1<=mpc.Nc)**

**for c = 2:mpc.Nc-p**

**filler = horzcat(filler, zeros(dim));**

**end**

**end**

**mpc.C2 = vertcat(mpc.C2, filler); mpc.C1 = vertcat(mpc.C1, eye(2));**

**end**

**mpc.bcon = [zeros(2\*mpc.Nc, 1) + mpc.C1\*[kf.v1;kf.v2]; 20\*ones(2\*mpc.Nc, 1) - mpc.C1\*[kf.v1; kf.v2]; ones(2\*mpc.Nc,1); 5\*ones(2\*mpc.Nc,1)];**

**mpc.Acon = [-mpc.C2;mpc.C2; -eye(2\*mpc.Nc); eye(2\*mpc.Nc)]; iter = 500;**

**for i = 1:iter**

**mpc = cstr(mpc,kf);**

**kf = plant(mpc,kf); kf = Kalman(kf,mpc);**

**end figure(1)**

**plot(kf.x\_po(1,:), 'LineWidth', 2); hold on**

**plot(kf.x\_po(2,:), 'LineWidth', 2);**

**plot(kf.x\_po(3,:), 'LineWidth', 2);**

**plot(kf.x\_po(4,:), 'LineWidth', 2); grid on**

**ax = gca; ax.FontSize = 10;**

**legend("H1","H2","H3","H4",'FontSize',6) xlabel("Iteration",'FontWeight','bold','FontSize',10) ylabel("Height (cm)",'FontWeight','bold','FontSize',10) title("Heights", 'FontWeight', 'bold','FontSize',10) axis tight**

**ylim([0 15]) hold off figure(2)**

**plot(kf.err(1,:), 'LineWidth', 2); hold on**

**plot(kf.err(2,:), 'LineWidth',2); grid on**

**ax = gca; ax.FontSize = 10;**

**legend("delH1","delH4",'FontSize',6) xlabel("Iteration",'FontWeight','bold','FontSize',10) ylabel("Error (cm)",'FontWeight','bold','FontSize',10) title("Error from Setpoint", 'FontWeight', 'bold','FontSize',10) axis tight**

**hold off**

**function kf = priori(kf, mpc) Xpost = kf.x\_po(:,end);**

**Xpr= kf.Ad\*Xpost + kf.Bd \* mpc.U(:,end); kf.P\_pr = kf.Ad \* kf.P\_pr \* kf.Ad' + kf.Q ; kf.x\_pr(:,end+1) = Xpr;**

**end**

**function kf = Kappa(kf)**

**dr = kf.Hd \* kf.P\_pr \* kf.Hd' + kf.R; nr = kf.P\_pr \* kf.Hd' ;**

**kf.kappa = nr / dr; end**

**function kf = posterior(kf,mpc)**

**kf.x\_po(:,end+1) = kf.x\_pr(:,end) + kf.kappa \* ( kf.Y(:,end) - kf.Hd \* kf.x\_pr(:,end) ) ;**

**kf.P\_pr = (eye(4) - kf.kappa\*kf.Hd)\*kf.P\_pr;**

**kf.Zest(:,end+1) = [ kf.kc \* kf.x\_po(2,end) ; kf.kc\*kf.x\_po(3,end)];**

**kf.err(:,end+1) = [kf.x\_po(2,end);kf.x\_po(3,end)] - [13.7;2.8];**

**end**

**function kf = Kalman(kf,mpc) kf = priori(kf,mpc);**

**kf = Kappa(kf);**

**kf = posterior(kf,mpc); end**

**function mpc = uncstr(mpc,kf) x\_cur = kf.x\_po(:, end); x\_prev = kf.x\_po(:, end-1); del\_x = x\_cur - x\_prev;**

**mpc.x = [del\_x; kf.Zest(:,end)];**

**delta = (mpc.phi' \* mpc.phi + mpc.R)\ (mpc.phi' \* (mpc.reference - mpc.F\*mpc.x));**

**mpc.U(:, end+1) = delta(1:2) + mpc.U(:,end); end**

**function mpc = cstr(mpc,kf) x\_cur = kf.x\_po(:, end); x\_prev = kf.x\_po(:, end-1); del\_x = x\_cur - x\_prev;**

**mpc.x = [del\_x; kf.Zest(:,end)]; H = mpc.phi' \* mpc.phi + mpc.R ;**

**F = mpc.phi' \* (mpc.reference -mpc.F \* mpc.x ) ; [mopc,delta] = mp\_controller(mpc);**

**mpc.U(:, end+1) = delta(1:2) + mpc.U(:,end); end**

**function [mopc, u\_opt] = mp\_controller(mpc)**

**cfun = @(x) (mpc.reference-mpc.F\*mpc.x-mpc.phi\*x)'\*... (mpc.reference-mpc.F\*mpc.x-mpc.phi\*x) + ... x'\*mpc.R\*x;**

**u0 = repmat(mpc.U(:,end), mpc.Nc, 1);**

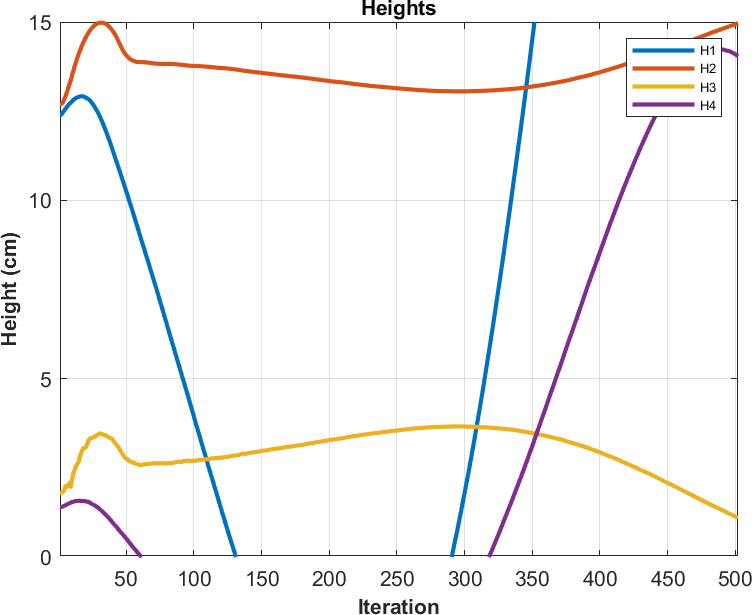
**x = fmincon(cfun, u0 , mpc.Acon, mpc.bcon); u\_opt = x;**

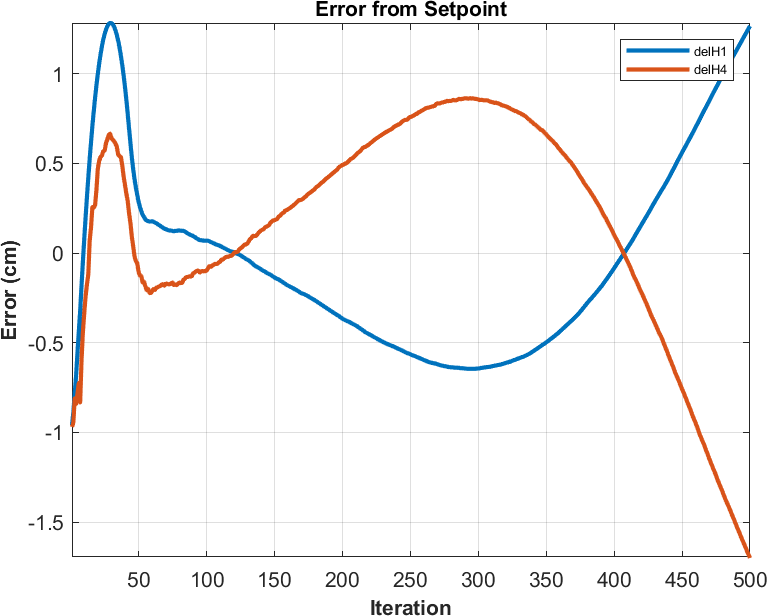
**mopc = mpc; end**

**function kf = plant(mpc,kf)**

**kf.Y(:,end+1) = kf.Hd \* ( kf.Ad \* kf.x\_po(:,end) + kf.Bd \* mpc.U(:,end)); kf.Y(:,end) = awgn(kf.Y(:,end),45);**

**end**





# Part C - b

**clc; clear; close all;**

**% initializing the kf structure which will carry all our parameters and**

**% solutions global kf; global g; global mpc;**

**% parameters kf = struct();**

**kf.A1 = 28; %(cm^2) kf.A2 = 32;**

**kf.A3 = 28;**

**kf.A4 = 32;**

**kf.A = [kf.A1, kf.A2, kf.A3, kf.A4]; kf.a1 = 0.071; kf.a3 = 0.071; %(cm^2) kf.a2 = 0.057; kf.a4 = 0.057;**

**kf.a = [kf.a1, kf.a2, kf.a3, kf.a4]; kf.kc = 0.5; % (V/cm)**

**g = 981; %(cm/s^2)**

**kf.gamma1 = 0.7; kf.gamma2 = 0.6; kf.k1 = 3.33; kf.k2 = 3.35; %[cm^3/Vs] kf.kc = 1; % [V/cm]**

**kf.kc = 1; % [V/cm]**

**kf.v1 = 3; kf.v2 = 3; % (V) kf.U = [kf.v1; kf.v2];**

**kf.h0 = [12.4; 12.7; 1.8; 1.4];**

**kf.P\_pr = 100000\*eye(4); kf.Q = 10\*eye(4); kf.R = 10\*eye(2); T = [];**

**kf.x\_pr = [];**

**kf.err=[];**

**% desired outputs for plotting**

**% Finding the value for the term T for all the Elements for j = 1:4**

**T(j) = (kf.A(j)/kf.a(j))\*sqrt(2\*kf.h0(j)/g) ; end**

**% Initializing the Control Input Matrix, State Matrix and Output Matrix kf.Ac = [ -1/T(1), 0, kf.A3/(kf.A1\*T(3)), 0 ; 0, -1/T(2), 0,**

**kf.A4/(kf.A2\*T(4)); 0, 0, -1/T(3), 0; 0, 0, 0, -1/T(4)];**

**kf.Bc = [kf.gamma1\*kf.k1/kf.A1 0 ; 0 kf.gamma2\*kf.k2/ kf.A2; 0 (1 -**

**kf.gamma2)\*kf.k2/kf.A3; (1-kf.gamma1)\*kf.k1/kf.A4 0];**

**kf.Dc = 0;**

**kf.Hcc = [kf.kc 0 0 0; 0 0 kf.kc 0];**

**kf.Hc = [0 kf.kc 0 0; 0 0 0 kf.kc];**

**% Initializing the array containing posterior values of the States, with**

**% first val being h0 kf.x\_po(:,1) = kf.h0;**

**kf.x\_po(:,2) = kf.h0;**

**kf.Y = zeros(2,1); kf.Zest = zeros(2,1);**

**% changing the Matrices to discrete domain based on step of 0.1 state\_space = ss(kf.Ac, kf.Bc, kf.Hc, kf.Dc); state\_space\_discrete = c2d(state\_space, 0.1);**

**kf.Ad = state\_space\_discrete.A; kf.Bd = state\_space\_discrete.B; kf.Hd = state\_space\_discrete.C; kf.Dd = state\_space\_discrete.D; mpc = struct();**

**mpc.Np = 50; mpc.Nc = 8 ;**

**mpc.A = [kf.Ad, zeros(4,2);kf.Hcc\*kf.Ad, eye(2)]; mpc.B = [kf.Bd; kf.Hcc\*kf.Bd];**

**mpc.C = [zeros(2,4), eye(2)]; mpc.F = mpc.C\*mpc.A; mpc.reference = [13.7; 2.8]; mpc.R = 0.1\*eye(2\*mpc.Nc); mpc.U = [kf.v1; kf.v2]; mpc.F = mpc.C\*mpc.A;**

**dims1 = size(kf.Hcc); mpc.q = dims1(1); dims2 = size(kf.Bd); mpc.m = dims2(2); mpc.b = [];**

**for n = 2:mpc.Np**

**mpc.F = vertcat(mpc.F, mpc.C\*mpc.A^n); mpc.reference = vertcat(mpc.reference, [13.7; 2.8]);**

**end**

**dim = size(mpc.C\*mpc.B);**

**mpc.phi = horzcat(mpc.C\*mpc.B, zeros(dim(1), dim(2)\*(mpc.Nc-1))); for p = 1:mpc.Np-1**

**filler = mpc.C\*(mpc.A^p)\*mpc.B; for c = 1: p**

**if c+1<=mpc.Nc**

**filler = horzcat(filler, mpc.C\*(mpc.A^(p-c))\*mpc.B);**

**end**

**end**

**if (p+1<=mpc.Nc)**

**for c = 2:mpc.Nc-p**

**filler = horzcat(filler, zeros(dim));**

**end**

**end**

**mpc.phi = vertcat(mpc.phi, filler); end**

**dim = size(eye(2));**

**mpc.C2 = horzcat(eye(2), zeros(dim(1), dim(2)\*(mpc.Nc-1))); mpc.C1 = eye(2);**

**for p = 1:mpc.Nc-1 filler = eye(2); for c = 1: p**

**if c+1<=mpc.Nc**

**filler = horzcat(filler, eye(2));**

**end**

**end**

**if (p+1<=mpc.Nc)**

**for c = 2:mpc.Nc-p**

**filler = horzcat(filler, zeros(dim));**

**end**

**end**

**mpc.C2 = vertcat(mpc.C2, filler); mpc.C1 = vertcat(mpc.C1, eye(2));**

**end**

**mpc.bcon = [zeros(2\*mpc.Nc, 1) + mpc.C1\*[kf.v1;kf.v2]; 20\*ones(2\*mpc.Nc, 1) - mpc.C1\*[kf.v1; kf.v2]; ones(2\*mpc.Nc,1); 5\*ones(2\*mpc.Nc,1)];**

**mpc.Acon = [-mpc.C2;mpc.C2; -eye(2\*mpc.Nc); eye(2\*mpc.Nc)]; iter = 1000;**

**for i = 1:iter**

**mpc = cstr(mpc,kf); kf = plant(mpc,kf); kf = Kalman(kf,mpc);**

**end figure(1)**

**plot(kf.x\_po(1,:), 'LineWidth', 2); hold on**

**plot(kf.x\_po(2,:), 'LineWidth', 2);**

**plot(kf.x\_po(3,:), 'LineWidth', 2);**

**plot(kf.x\_po(4,:), 'LineWidth', 2); grid on**

**ax = gca; ax.FontSize = 10;**

**legend("H1","H2","H3","H4",'FontSize',6) xlabel("Iteration",'FontWeight','bold','FontSize',10) ylabel("Height (cm)",'FontWeight','bold','FontSize',10) title("Heights", 'FontWeight', 'bold','FontSize',10) axis tight**

**ylim([0 15]) hold off figure(2)**

**plot(kf.err(1,:), 'LineWidth', 2); hold on**

**plot(kf.err(2,:), 'LineWidth',2); grid on**

**ax = gca;**

**ax.FontSize = 10; legend("delH1","delH3",'FontSize',6) xlabel("Iteration",'FontWeight','bold','FontSize',10) ylabel("Error (cm)",'FontWeight','bold','FontSize',10)**

**title("Error from Setpoint", 'FontWeight', 'bold','FontSize',10) axis tight**

**hold off**

**function kf = priori(kf, mpc) Xpost = kf.x\_po(:,end);**

**Xpr= kf.Ad\*Xpost + kf.Bd \* mpc.U(:,end); kf.P\_pr = kf.Ad \* kf.P\_pr \* kf.Ad' + kf.Q ; kf.x\_pr(:,end+1) = Xpr;**

**end**

**function kf = Kappa(kf)**

**dr = kf.Hd \* kf.P\_pr \* kf.Hd' + kf.R; nr = kf.P\_pr \* kf.Hd' ;**

**kf.kappa = nr / dr; end**

**function kf = posterior(kf,mpc)**

**kf.x\_po(:,end+1) = kf.x\_pr(:,end) + kf.kappa \* ( kf.Y(:,end) - kf.Hd \* kf.x\_pr(:,end) ) ;**

**kf.P\_pr = (eye(4) - kf.kappa\*kf.Hd)\*kf.P\_pr;**

**kf.Zest(:,end+1) = [kf.kc \* kf.x\_po(1,end);kf.kc \* kf.x\_po(3,end)];**

**kf.err(:,end+1) = [kf.x\_po(1,end);kf.x\_po(3,end)] - [13.7;2.8]; end**

**function kf = Kalman(kf,mpc) kf = priori(kf,mpc);**

**kf = Kappa(kf);**

**kf = posterior(kf,mpc); end**

**function mpc = uncstr(mpc,kf) x\_cur = kf.x\_po(:, end); x\_prev = kf.x\_po(:, end-1); del\_x = x\_cur - x\_prev;**

**mpc.x = [del\_x; kf.Zest(:,end)];**

**delta = (mpc.phi' \* mpc.phi + mpc.R)\ (mpc.phi' \* (mpc.reference - mpc.F\*mpc.x));**

**mpc.U(:, end+1) = delta(1:2) + mpc.U(:,end); end**

**function mpc = cstr(mpc,kf) x\_cur = kf.x\_po(:, end); x\_prev = kf.x\_po(:, end-1); del\_x = x\_cur - x\_prev;**

**mpc.x = [del\_x; kf.Zest(:,end)]; H = mpc.phi' \* mpc.phi + mpc.R ;**

**F = mpc.phi' \* (mpc.reference -mpc.F \* mpc.x ) ; [mopc,delta] = mp\_controller(mpc);**

**mpc.U(:, end+1) = delta(1:2) + mpc.U(:,end); end**

**function [mopc, u\_opt] = mp\_controller(mpc)**

**cfun = @(x) (mpc.reference-mpc.F\*mpc.x-mpc.phi\*x)'\*... (mpc.reference-mpc.F\*mpc.x-mpc.phi\*x) + ... x'\*mpc.R\*x;**

**u0 = repmat(mpc.U(:,end), mpc.Nc, 1);**

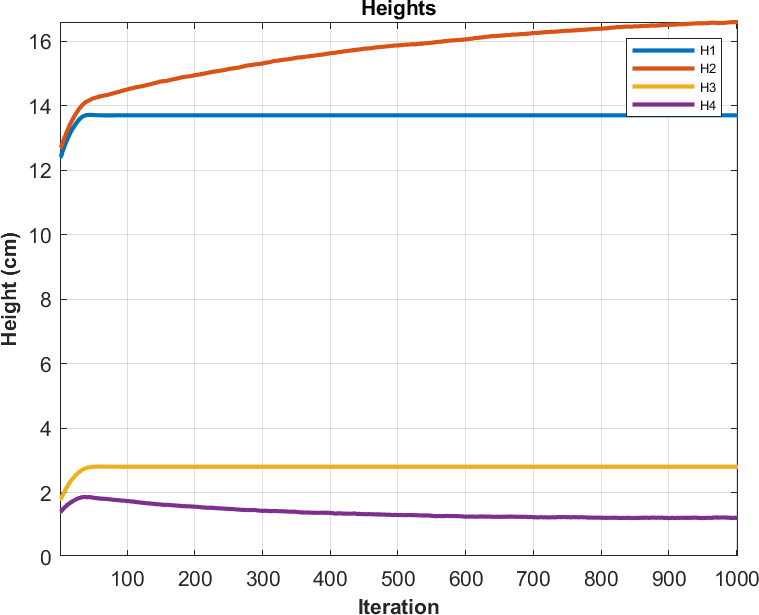
**x = fmincon(cfun, u0 , mpc.Acon, mpc.bcon); u\_opt = x;**

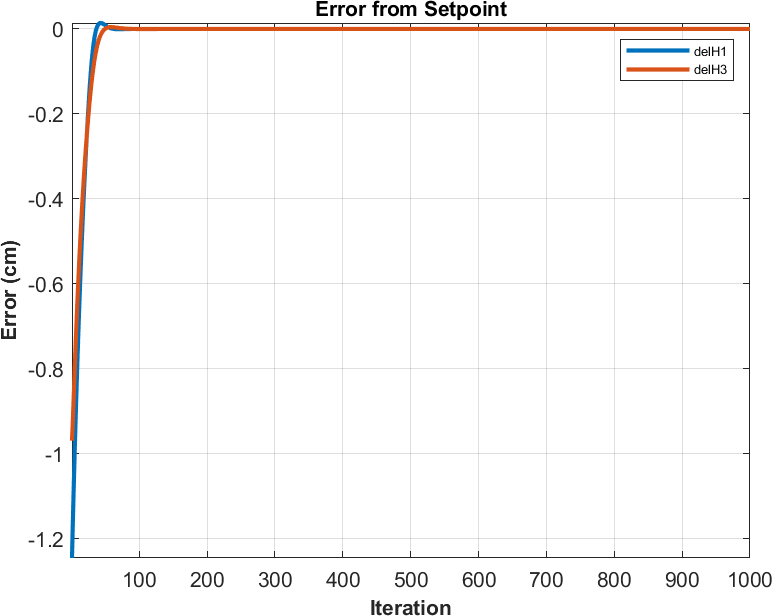
**mopc = mpc; end**

**function kf = plant(mpc,kf)**

**kf.Y(:,end+1) = kf.Hd \* ( kf.Ad \* kf.x\_po(:,end) + kf.Bd \* mpc.U(:,end)); kf.Y(:,end) = awgn(kf.Y(:,end),50);**

**end**





As we can see the second case reaches set point, whereas the first case could not reach the appropriate setpoints over there due to its tendency to focus on constraints more than moving towards the setpoint in first case.