Contents

AA 203 HW 8 Question 3

Somrita Banerjee

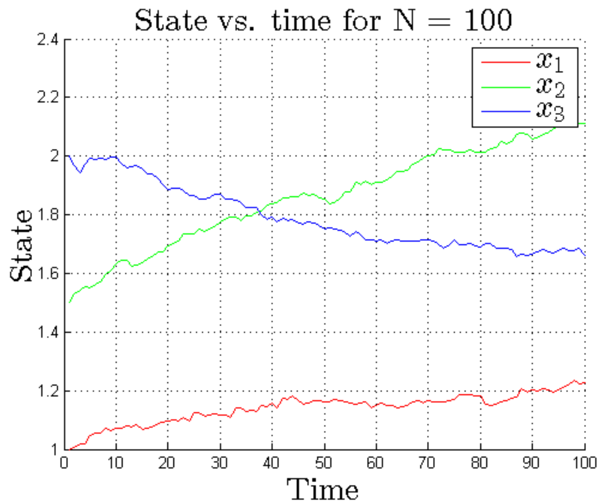
```
clc
clear all
close all
rng('default') % For reproducibility

A0 = [0.99 0 0;
    0 0.99 0;
    0 0 0.99];
B0 = [1 0 0;
    0 1 0;
    0 0 1];
A = [1.01 0.01 0;
    0.01 1.01 0.01;
    0 0.01 1.01];
B = B0;

noiseStdDev = 0.01;
Q = eye(3);
R = 1000*eye(3);

xInit = [1.0; 1.5; 2.0];
```

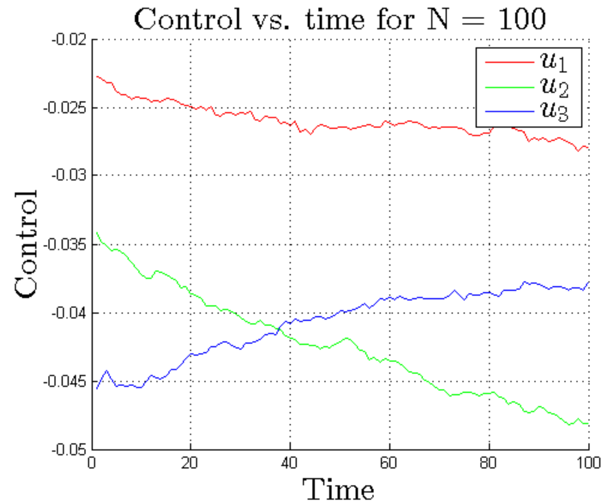
Part a - Plain LQR

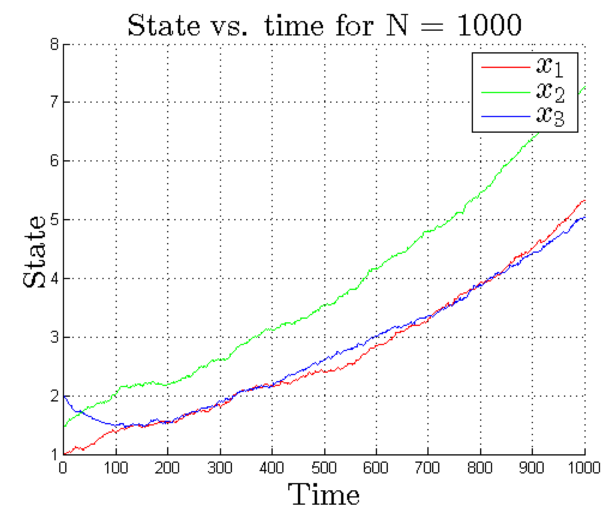
```
[K,S,e] = dlqr(A0,B0,Q,R);
N_vals = [100 1000 10000];
for N = N_vals
    x = zeros(3, N);
    u = zeros(3, N);
    cost = 0;
    for i = 1:N
        if i == 1
            x(:,i) = xInit;
        else
            noise = normrnd(0, noiseStdDev, [3,1]);
            x(:,i) = A*x(:,i-1) + B*u(:,i-1) + noise;
        end
        u(:,i) = -K*x(:,i);
        cost = cost + x(:,i)'*Q*x(:,i) + u(:,i)'*R*u(:,i);
    end

    fprintf('Cost for N = %d is %.2f \n',N, cost)

    figure
    hold on
    plot([1:N], x(1,:),'r')
    plot([1:N], x(2,:),'g')
    plot([1:N], x(3,:),'b')
    titl = sprintf('State vs. time for N = %d',N);
    legend({'$$x_1$$','$$x_2$$','$$x_3$$'},'Interpreter','latex','FontSize',20);
    xlabel('Time','Interpreter','latex','FontSize',20);
    ylabel('State','Interpreter','latex','FontSize',20);
    title(titl,'Interpreter','latex','FontSize',20);
    grid on

    figure
    hold on
    plot([1:N], u(1,:),'r')
    plot([1:N], u(2,:),'g')
    plot([1:N], u(3,:),'b')
    titl = sprintf('Control vs. time for N = %d',N);
    legend({'$$u_1$$','$$u_2$$','$$u_3$$'},'Interpreter','latex','FontSize',20);
    xlabel('Time','Interpreter','latex','FontSize',20);
    ylabel('Control','Interpreter','latex','FontSize',20);
    title(titl,'Interpreter','latex','FontSize',20);
    grid on
end
```

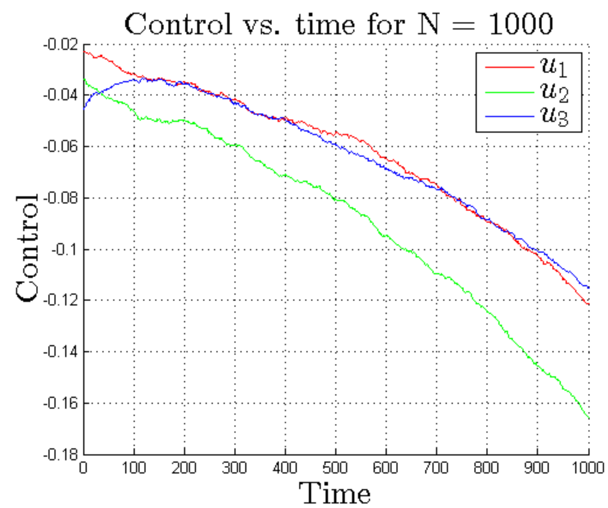
```
Cost for N = 100 is 1216.89
Cost for N = 1000 is 53255.71
Cost for N = 10000 is 2418241054959084.00
```

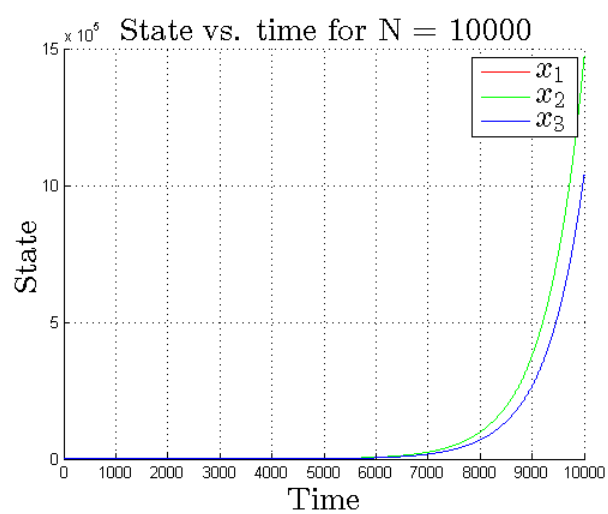
We see that our naive implementation of an LQR with the wrong parameters results in the cost increasing drastically as the number of time steps increases. The state and control also become unbounded.

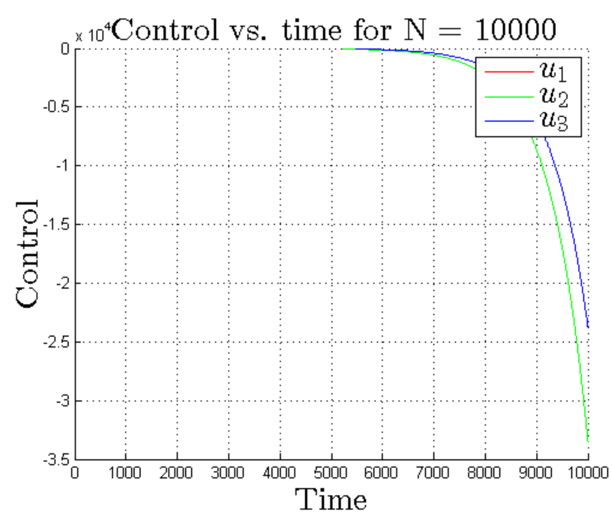












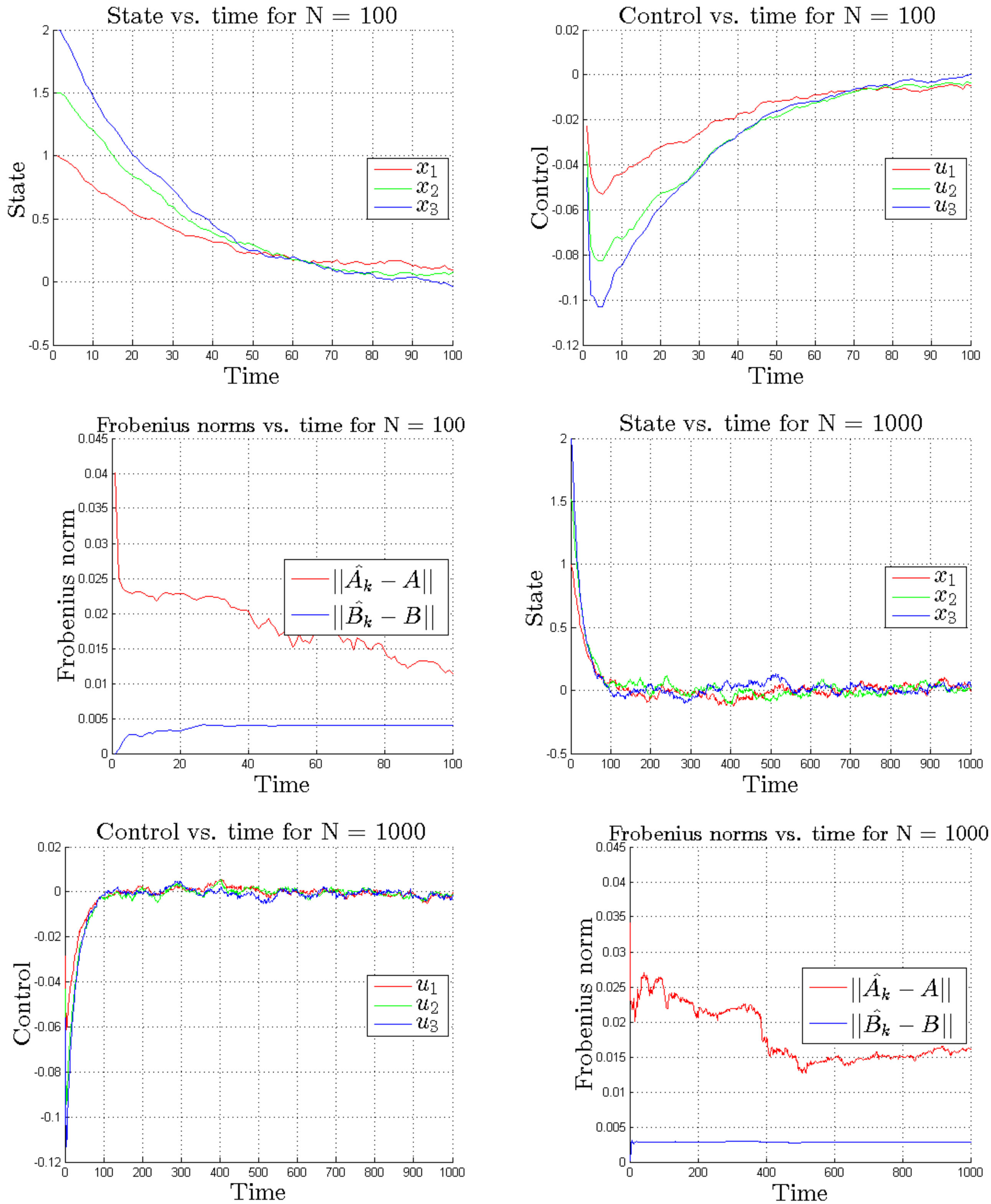
Part b- Certainty equivalent adaptive LQR controller

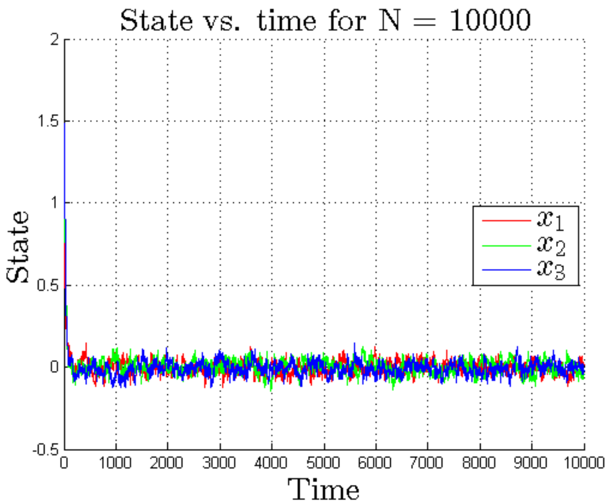
```
N_vals = [100 1000 10000];
for N = N_vals
    x = zeros(3, N);
    u = zeros(3, N);
    A_vals = zeros(3,3,N);
    B_vals = zeros(3,3,N);
    L_vals = zeros(6,6,N);
    Q_vals = zeros(6,3,N);
    A_fro_vals = zeros(1,N);
    B_fro_vals = zeros(1,N);
    cost = 0;
    for i = 1:N
        if i == 1
            x(:,i) = xInit;
            A_vals(:,:,i) = A0;
            B_vals(:,:,i) = B0;
```

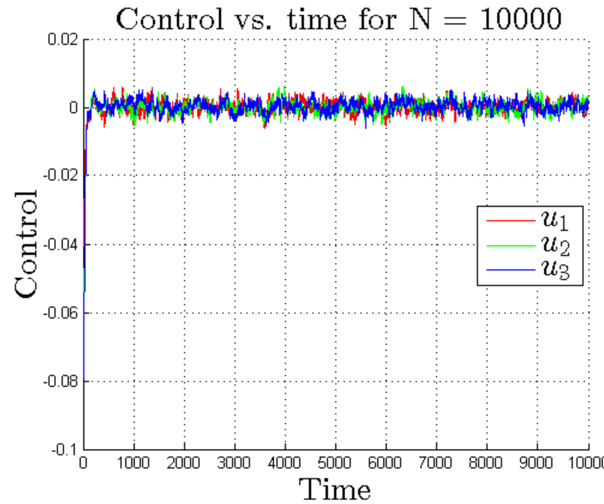
```
            L_vals(:,:,i) = eye(6);
            Q_vals(:,:,i) = [A0' B0']';
        else
            noise = normrnd(0, noiseStdDev, [3,1]);
            xk = x(:,i-1);
            uk = u(:,i-1);
            Ak = A_vals(:,:,i-1);
            Bk = B_vals(:,:,i-1);
            x(:,i) = A*xk + B*uk + noise;

            % Update L, Q, A, B vals for ith
            xbar = [xk' uk']';
            Lk = L_vals(:,:,i-1);
            Qk = Q_vals(:,:,i-1);
            Lknext = Lk - (1/(1+xbar'*Lk*xbar))*(Lk*xbar)*(Lk*xbar)';
            Qknext = xbar*x(:,i)' + Qk;
            LQnext = (Lknext*Qknext)';
            L_vals(:,:,i) = Lknext;
            Q_vals(:,:,i) = Qknext;
            A_vals(:,:,i) = LQnext(:,1:3);
            B_vals(:,:,i) = LQnext(:,4:6);
        end
        [K,~,~] = dlqr(A_vals(:,:,i),B_vals(:,:,i),Q,R);
        u(:,i) = -K*x(:,i);
        cost = cost + x(:,i)'*Q*x(:,i) + u(:,i)'*R*u(:,i);

        % Store frobenius norms
        A_fro_vals(i) = norm(A_vals(:,:,i)-A,'fro');
        B_fro_vals(i) = norm(B_vals(:,:,i)-B,'fro');
    end

    fprintf('Cost for N = %d is %.2f \n',N, cost)

    figure
    hold on
    plot([1:N], x(1,:),'r')
    plot([1:N], x(2,:),'g')
    plot([1:N], x(3,:),'b')
    titl = sprintf('State vs. time for N = %d',N);
    legend({'$$x_1$$','$$x_2$$','$$x_3$$'},'Interpreter','latex','FontSize',20,'Location','east');
    xlabel('Time','Interpreter','latex','FontSize',20);
    ylabel('State','Interpreter','latex','FontSize',20);
    title(titl,'Interpreter','latex','FontSize',20);
    grid on

    figure
    hold on
    plot([1:N], u(1,:),'r')
    plot([1:N], u(2,:),'g')
    plot([1:N], u(3,:),'b')
    titl = sprintf('Control vs. time for N = %d',N);
    legend({'$$u_1$$','$$u_2$$','$$u_3$$'},'Interpreter','latex','FontSize',20,'Location','east');
    xlabel('Time','Interpreter','latex','FontSize',20);
    ylabel('Control','Interpreter','latex','FontSize',20);
    title(titl,'Interpreter','latex','FontSize',20);
    grid on

    figure
    hold on
    plot([1:N],A_fro_vals,'r')
    plot([1:N],B_fro_vals,'b')
    titl = sprintf('Frobenius norms vs. time for N = %d',N);
    legend({'$$||\hat{A_k}-A||$$','$$||\hat{B_k}-B||$$'},'Interpreter','latex','FontSize',20,'Location','east');
    xlabel('Time','Interpreter','latex','FontSize',20);
    ylabel('Frobenius norm','Interpreter','latex','FontSize',20);
    title(titl,'Interpreter','latex','FontSize',18);
    grid on
end
```

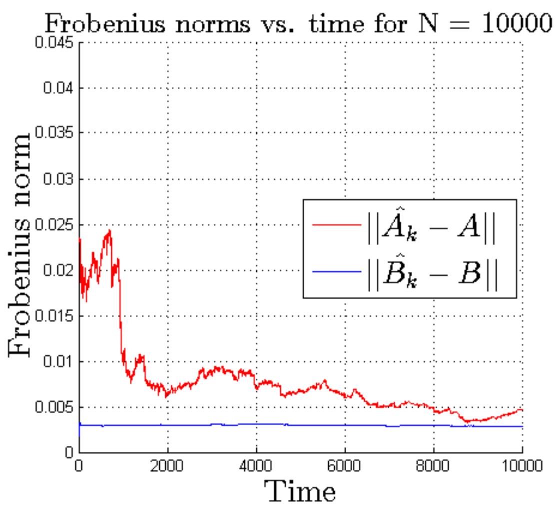
```
Cost for N = 100 is 478.99
Cost for N = 1000 is 499.20
Cost for N = 10000 is 601.38
```

After introducing a certainty-equivalent adaptive LQR controller, we see that in just a few timesteps, the control converges to 0 and the state is brought to 0. The cost is much lower than with the naive LQR controller. However, the system identification is not perfect because the Frobenius norm of the difference between the estimated parameter matrices and the true parameter matrices does not converge to 0.


State vs. time for N = 100
State
Time
x_1
x_2
x_3
Control vs. time for N = 100
Control
Time
u_1
u_2
u_3
Frobenius norms vs. time for N = 100
Frobenius norm
Time
$||\hat{A}_k - A||$
$||\hat{B}_k - B||$
State vs. time for N = 1000
State
Time
x_1
x_2
x_3
Control vs. time for N = 1000
Control
Time
u_1
u_2
u_3
Frobenius norms vs. time for N = 1000
Frobenius norm
Time
$||\hat{A}_k - A||$
$||\hat{B}_k - B||$







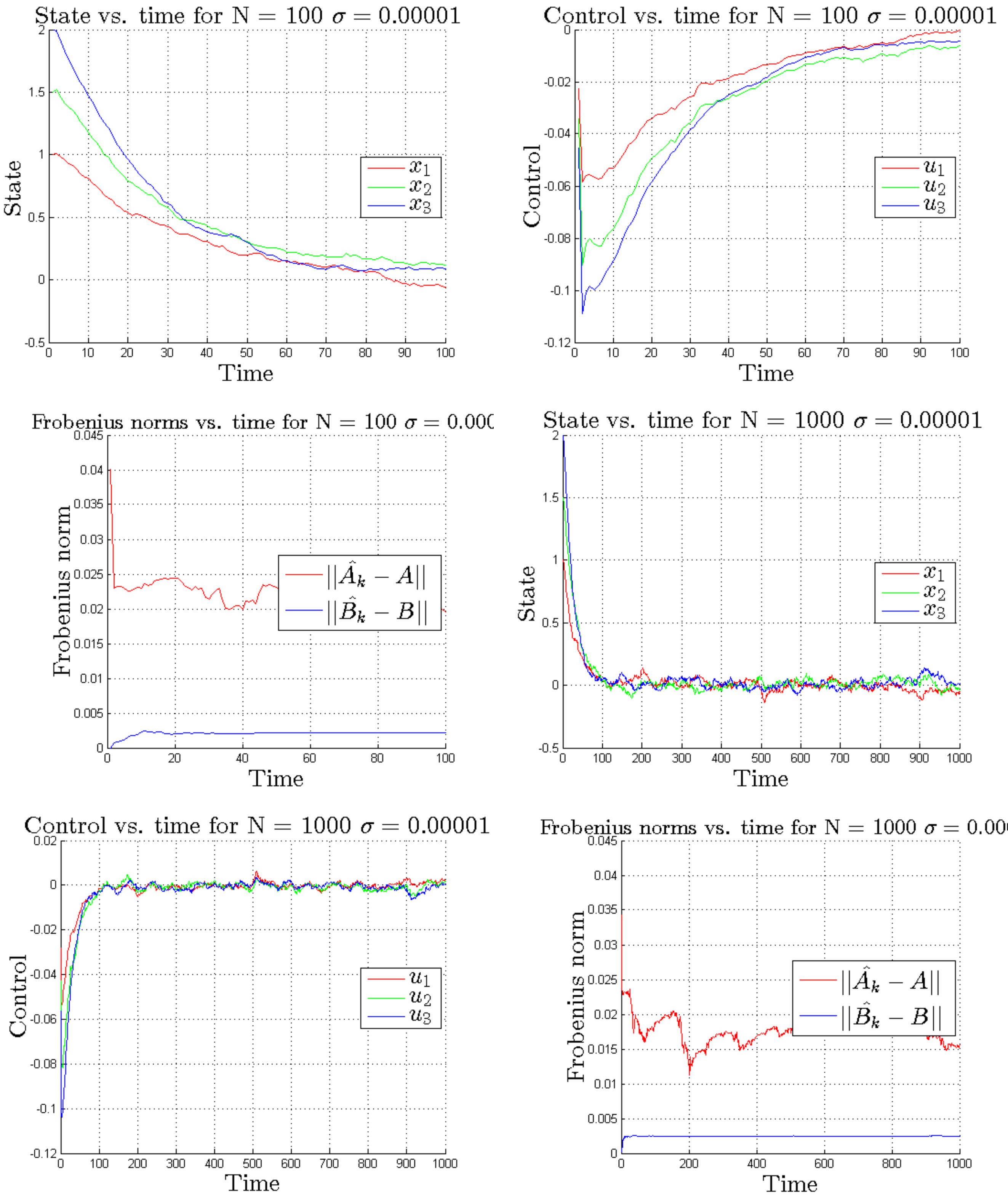
Part c- adaptive LQR controller with white noise

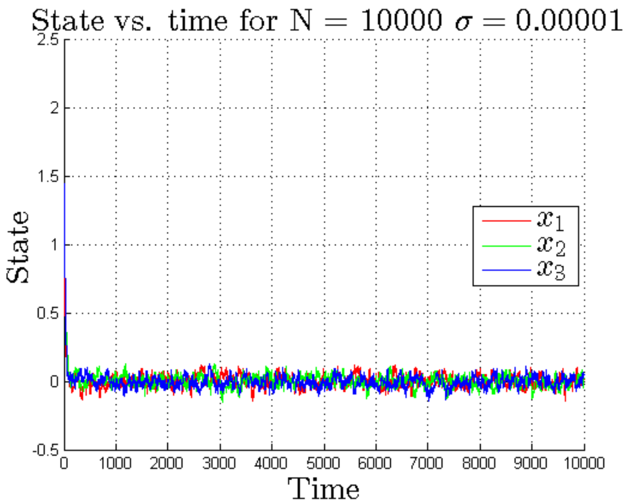
```
N_vals = [100 1000 10000];
white_noise_stdev_vals = [0.00001 0.001 0.01 1];
for white_noise_stdev = white_noise_stdev_vals
    for N = N_vals
        x = zeros(3, N);
        u = zeros(3, N);
        A_vals = zeros(3,3,N);
        B_vals = zeros(3,3,N);
        L_vals = zeros(6,6,N);
        Q_vals = zeros(6,3,N);
        A_fro_vals = zeros(1,N);
        B_fro_vals = zeros(1,N);
        cost = 0;
        for i = 1:N
            if i == 1
                x(:,i) = xInit;
                A_vals(:,:,i) = A0;
                B_vals(:,:,i) = B0;
                L_vals(:,:,i) = eye(6);
                Q_vals(:,:,i) = [A0' B0']';
            else
                noise = normrnd(0, noiseStdDev, [3,1]);
                xk = x(:,i-1);
                uk = u(:,i-1);
                Ak = A_vals(:,:,i-1);
                Bk = B_vals(:,:,i-1);
                x(:,i) = A*xk + B*uk + noise;

                % Update L, Q, A, B vals for ith
                xbar = [xk' uk']';
                Lk = L_vals(:,:,i-1);
                Qk = Q_vals(:,:,i-1);
                Lknext = Lk - (1/(1+xbar'*Lk*xbar))*(Lk*xbar)*(Lk*xbar)';
                Qknext = xbar*x(:,i)' + Qk;
                LQnext = (Lknext*Qknext)';
                L_vals(:,:,i) = Lknext;
                Q_vals(:,:,i) = Qknext;
                A_vals(:,:,i) = LQnext(:,1:3);
                B_vals(:,:,i) = LQnext(:,4:6);
            end
            white_noise = normrnd(0, white_noise_stdev, [3,1]);
            [K,~,~] = dlqr(A_vals(:,:,i),B_vals(:,:,i),Q,R);
            u(:,i) = -K*x(:,i) + white_noise;
            cost = cost + x(:,i)'*Q*x(:,i) + u(:,i)'*R*u(:,i);
```

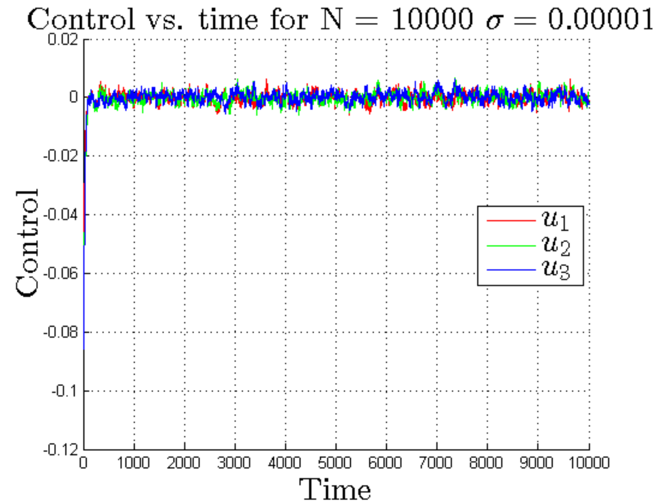
```
            % Store frobenius norms
            A_fro_vals(i) = norm(A_vals(:,:,i)-A,'fro');
            B_fro_vals(i) = norm(B_vals(:,:,i)-B,'fro');
        end

        fprintf('Cost for N = %d, white noise std dev = %.5f is %.2f \n',N, white_noise_stdev, cost)

        figure
        hold on
        plot([1:N], x(1,:),'r')
        plot([1:N], x(2,:),'g')
        plot([1:N], x(3,:),'b')
        titl = sprintf('State vs. time for N = %d $\\sigma = %.5f$',N, white_noise_stdev);
        legend({'$$x_1$$','$$x_2$$','$$x_3$$'},'Interpreter','latex','FontSize',20,'Location','east');
        xlabel('Time','Interpreter','latex','FontSize',20);
        ylabel('State','Interpreter','latex','FontSize',20);
        title(titl,'Interpreter','latex','FontSize',20);
        grid on

        figure
        hold on
        plot([1:N], u(1,:),'r')
        plot([1:N], u(2,:),'g')
        plot([1:N], u(3,:),'b')
        titl = sprintf('Control vs. time for N = %d $\\sigma = %.5f$',N, white_noise_stdev);
        legend({'$$u_1$$','$$u_2$$','$$u_3$$'},'Interpreter','latex','FontSize',20,'Location','east');
        xlabel('Time','Interpreter','latex','FontSize',20);
        ylabel('Control','Interpreter','latex','FontSize',20);
        title(titl,'Interpreter','latex','FontSize',20);
        grid on

        figure
        hold on
        plot([1:N],A_fro_vals,'r')
        plot([1:N],B_fro_vals,'b')
        titl = sprintf('Frobenius norms vs. time for N = %d $\\sigma = %.5f$',N, white_noise_stdev);
        legend({'$$||\hat{A_k}-A||$$','$$||\hat{B_k}-B||$$'},'Interpreter','latex','FontSize',20,'Location','east');
        xlabel('Time','Interpreter','latex','FontSize',20);
        ylabel('Frobenius norm','Interpreter','latex','FontSize',20);
        title(titl,'Interpreter','latex','FontSize',18);
        grid on
    end
end
```

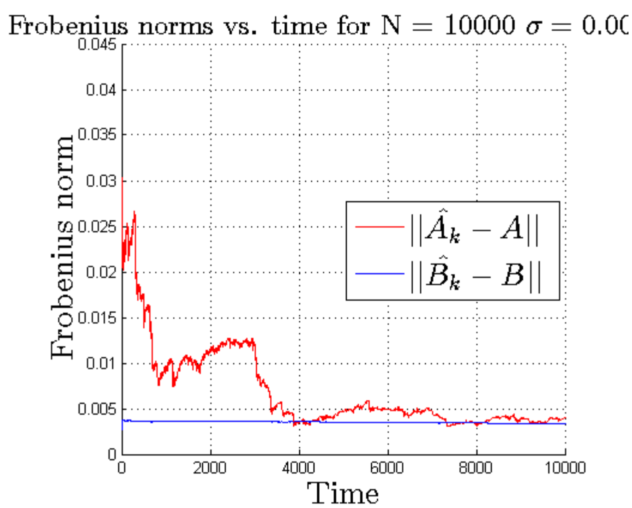
```
Cost for N = 100, white noise std dev = 0.00001 is 492.18
Cost for N = 1000, white noise std dev = 0.00001 is 482.81
Cost for N = 10000, white noise std dev = 0.00001 is 635.98
Cost for N = 100, white noise std dev = 0.00100 is 471.72
Cost for N = 1000, white noise std dev = 0.00100 is 495.86
Cost for N = 10000, white noise std dev = 0.00100 is 620.78
Cost for N = 100, white noise std dev = 0.01000 is 509.27
Cost for N = 1000, white noise std dev = 0.01000 is 777.46
Cost for N = 10000, white noise std dev = 0.01000 is 3776.37
Cost for N = 100, white noise std dev = 1.00000 is 310598.40
Cost for N = 1000, white noise std dev = 1.00000 is 3097063.33
Cost for N = 10000, white noise std dev = 1.00000 is 31337434.89
```

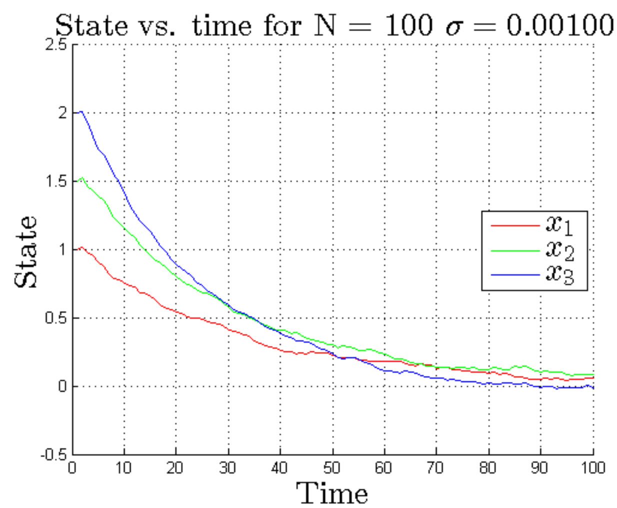
We notice a few interesting things after introducing white noise to the control. The first is that when the white noise standard deviation is very small, the performance is similar to the certainty equivalent LQR controller (just like we'd expect). As the white noise gets larger, the control input gets more jittery, and the cost increases (without blowing up), but the controller also does more exploration so the system identification gets better. At large standard deviation, with sigma =1, we see that the system identification becomes perfect, i.e. the Frobenius norm of the difference between the estimated parameter matrix and the true matrix converges to 0 (see last plot).

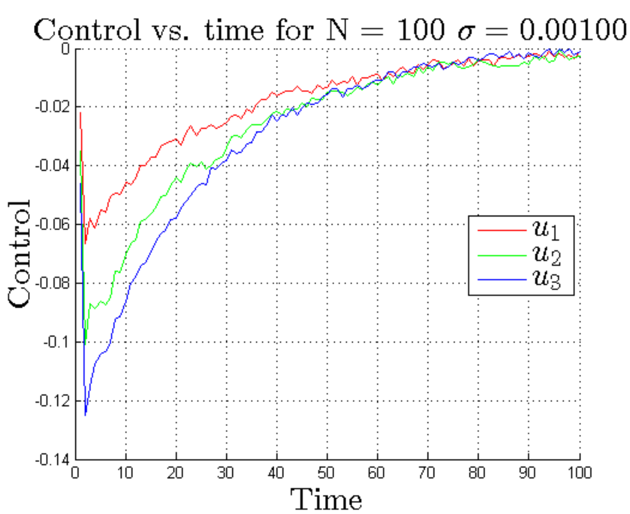

State vs. time for N = 100 σ = 0.00001
State
Time
x1
x2
x3
Control vs. time for N = 100 σ = 0.00001
Control
Time
u1
u2
u3
Frobenius norms vs. time for N = 100 σ = 0.000
Frobenius norm
Time
||Âk − A||
||B̂k − B||
State vs. time for N = 1000 σ = 0.00001
State
Time
x1
x2
x3
Control vs. time for N = 1000 σ = 0.00001
Control
Time
u1
u2
u3
Frobenius norms vs. time for N = 1000 σ = 0.00
Frobenius norm
Time
||Âk − A||
||B̂k − B||

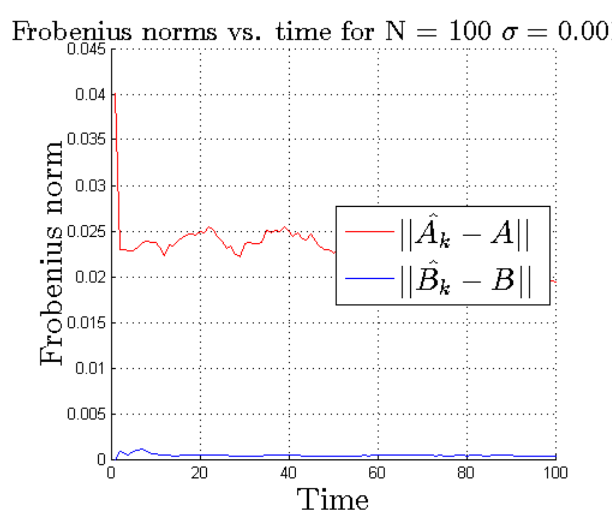

State vs. time for N = 10000 $\sigma = 0.00001$
State
Time
x_1
x_2
x_3

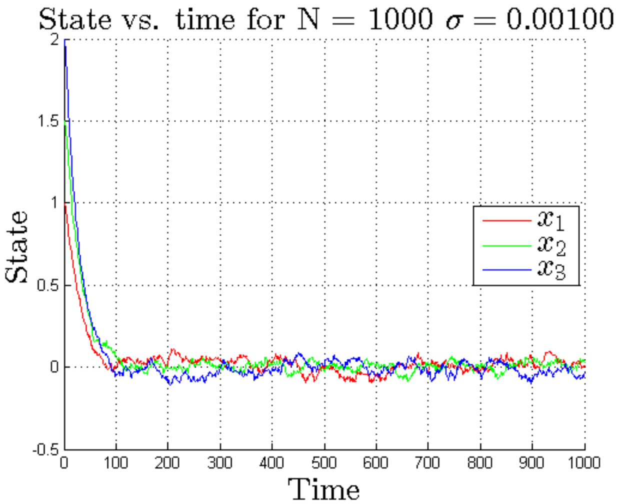

Control vs. time for N = 10000 $\sigma = 0.00001$
Control
Time
u_1
u_2
u_3

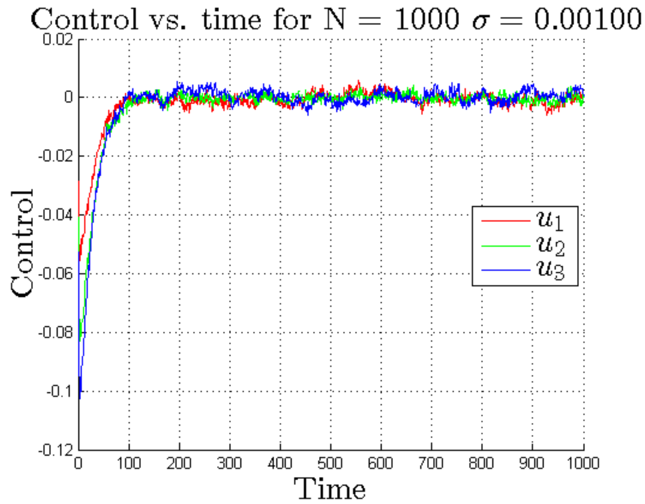

Frobenius norms vs. time for N = 10000 σ = 0.0
Frobenius norm
Time
$||\hat{A}_k - A||$
$||\hat{B}_k - B||$

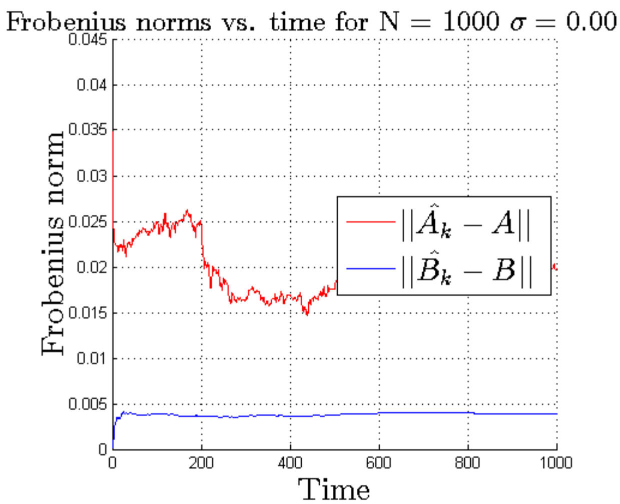

State vs. time for N = 100 $\sigma = 0.00100$
State
Time
x_1
x_2
x_3

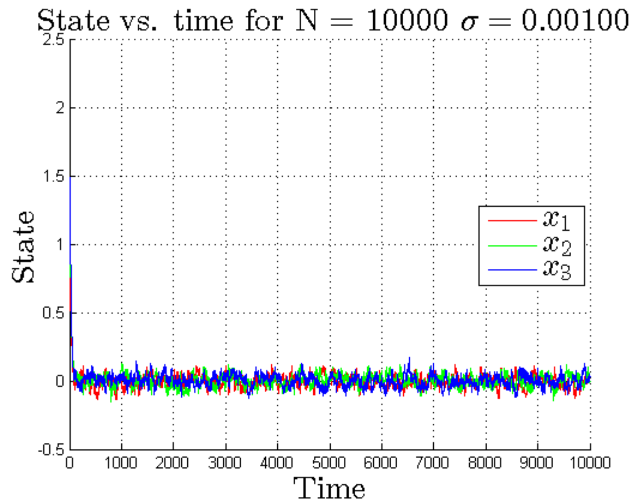

Control vs. time for N = 100 $\sigma = 0.00100$
Control
Time
u_1
u_2
u_3

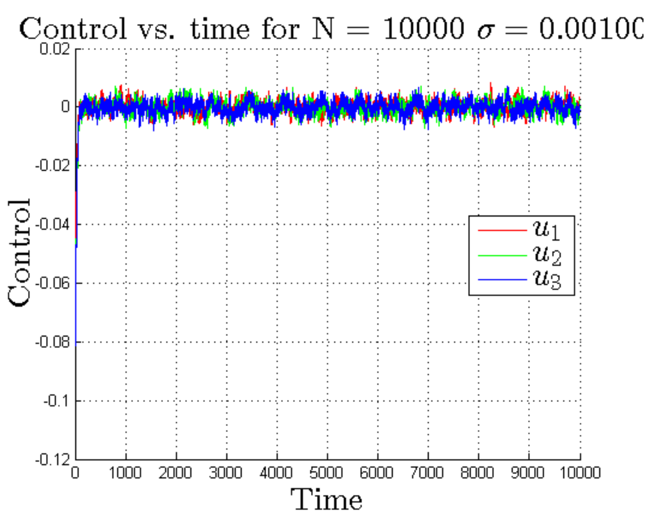

Frobenius norms vs. time for N = 100 $\sigma = 0.001$
Frobenius norm
Time
$||\hat{A}_k - A||$
$||\hat{B}_k - B||$

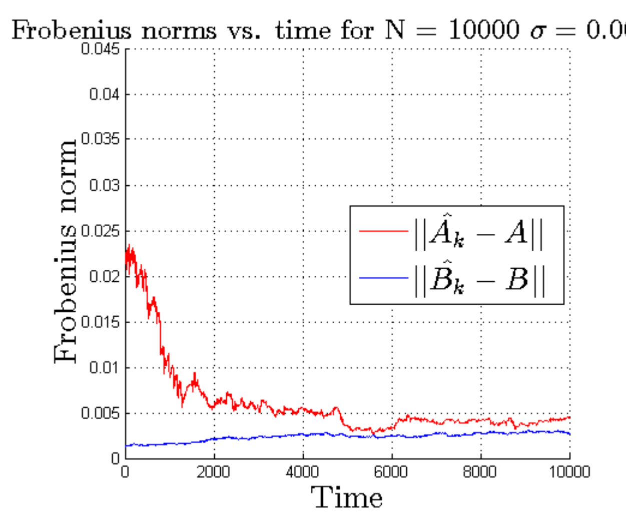

State vs. time for N = 1000 $\sigma = 0.00100$
State
Time
2
1.5
1
0.5
0
-0.5
0 100 200 300 400 500 600 700 800 900 1000
x_1
x_2
x_3

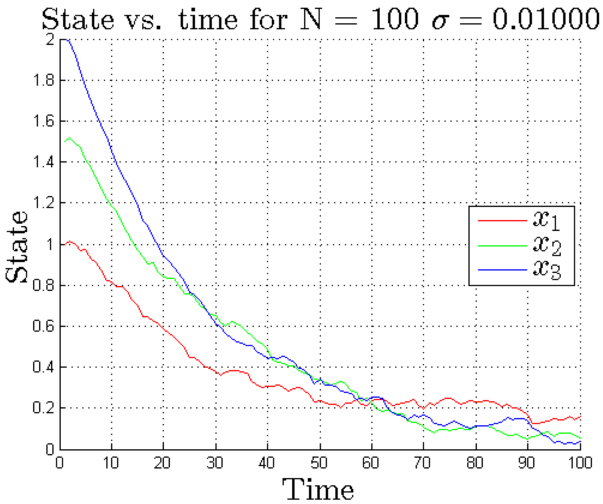

Control vs. time for N = 1000 $\sigma = 0.00100$
Control
Time
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
-0.12
0 100 200 300 400 500 600 700 800 900 1000
u_1
u_2
u_3

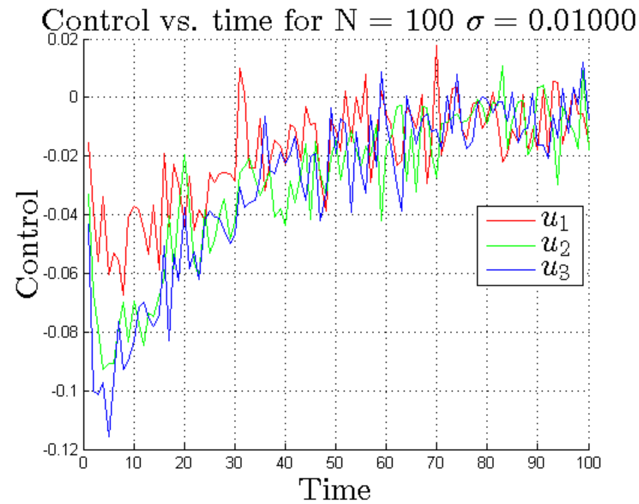

Frobenius norms vs. time for N = 1000 $\sigma = 0.00$
Frobenius norm
Time
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0 200 400 600 800 1000
$||\hat{A}_k - A||$
$||\hat{B}_k - B||$

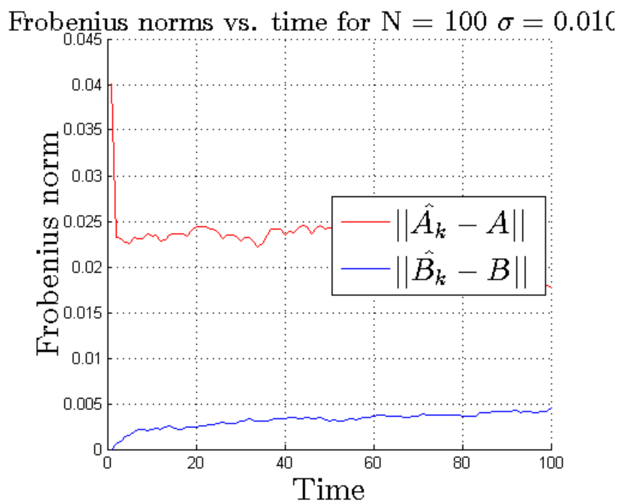

State vs. time for N = 10000 $\sigma = 0.00100$
State
Time
2.5
2
1.5
1
0.5
0
-0.5
0 1000 2000 3000 4000 5000 6000 7000 8000 9000 10000
x_1
x_2
x_3

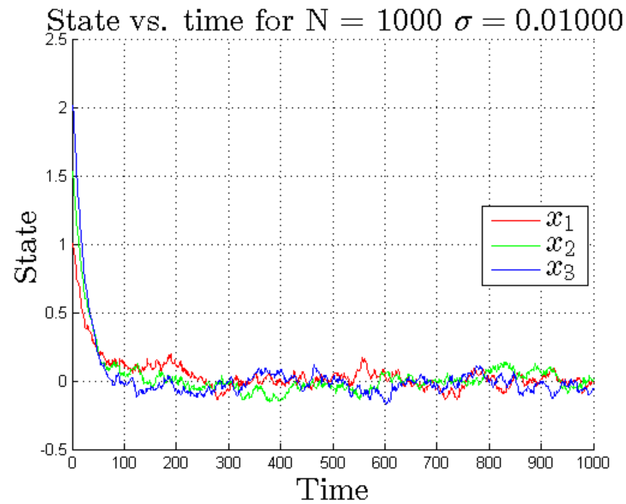

Control vs. time for N = 10000 $\sigma = 0.0010($
Control
Time
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
-0.12
0 1000 2000 3000 4000 5000 6000 7000 8000 9000 10000
u_1
u_2
u_3

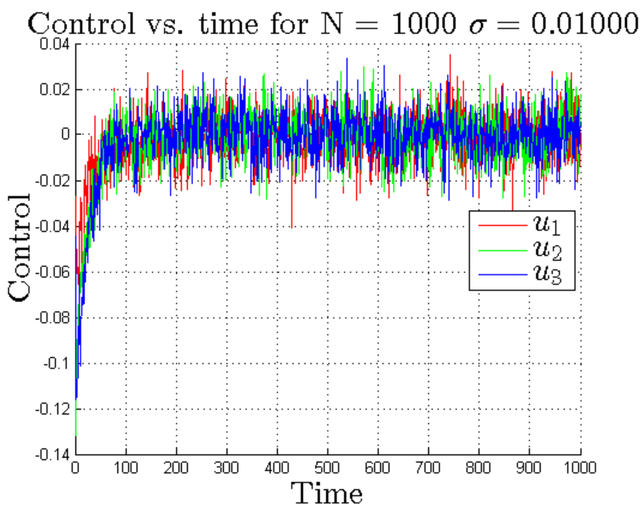

Frobenius norms vs. time for N = 10000 $\sigma = 0.0($
Frobenius norm
Time
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0 2000 4000 6000 8000 10000
$||\hat{A}_k - A||$
$||\hat{B}_k - B||$

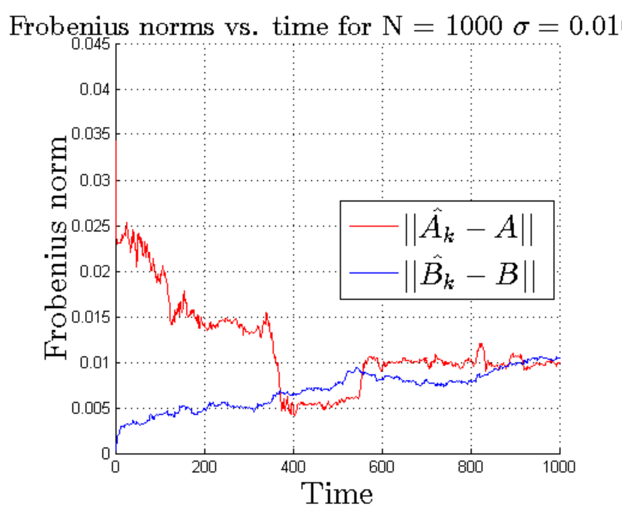

State vs. time for N = 100 $\sigma = 0.01000$
State
Time
x_1
x_2
x_3

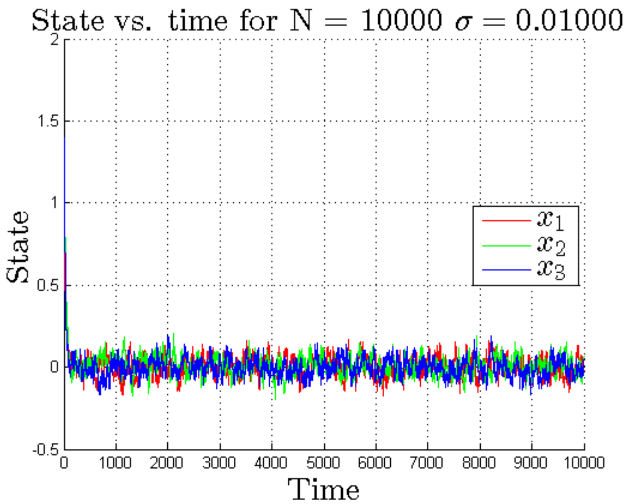

Control vs. time for N = 100 $\sigma = 0.01000$
Control
Time
u_1
u_2
u_3

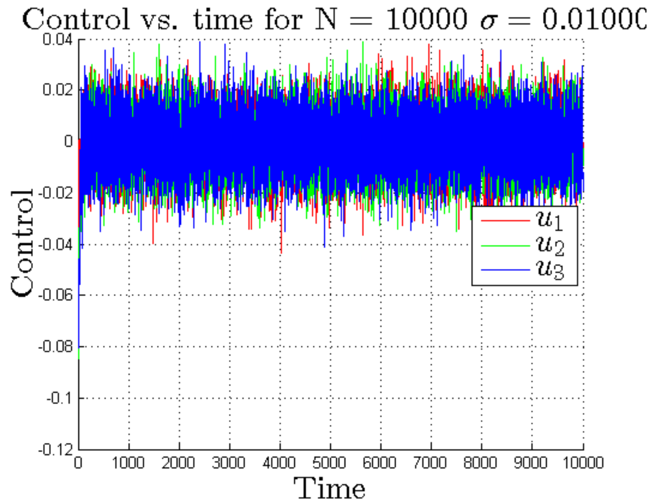

Frobenius norms vs. time for N = 100 $\sigma = 0.010$
Frobenius norm
Time
$||\hat{A}_k - A||$
$||\hat{B}_k - B||$

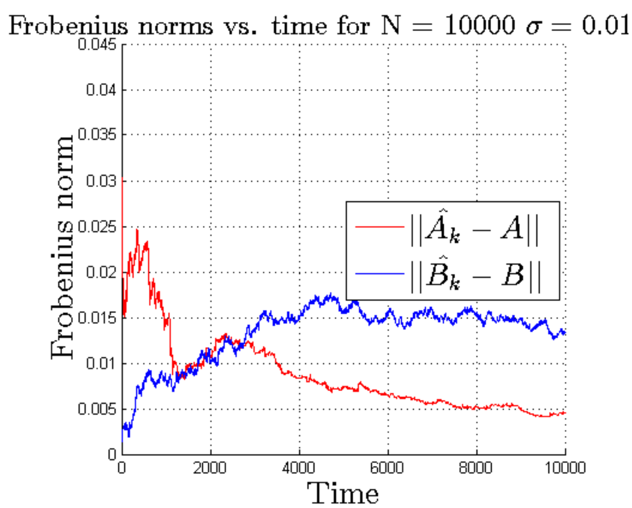

State vs. time for N = 1000 $\sigma = 0.01000$
State
Time
x_1
x_2
x_3

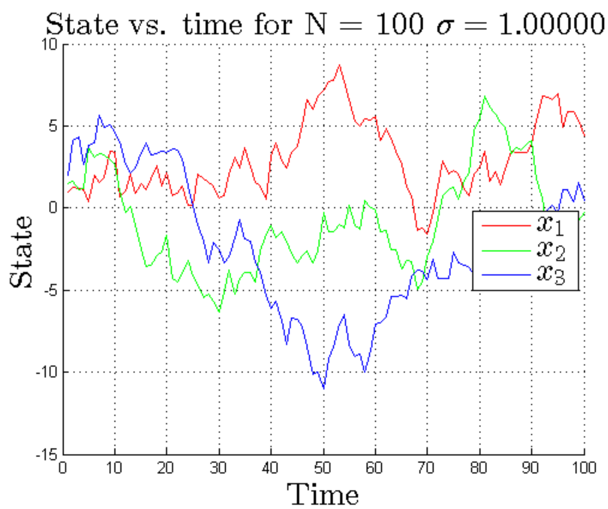

Control vs. time for N = 1000 $\sigma = 0.01000$
Control
Time
u_1
u_2
u_3

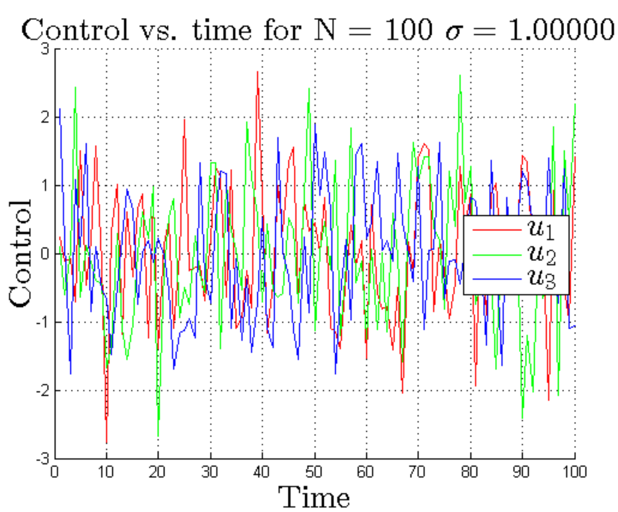

Frobenius norms vs. time for N = 1000 $\sigma = 0.01$
Frobenius norm
Time
$||\hat{A}_k - A||$
$||\hat{B}_k - B||$

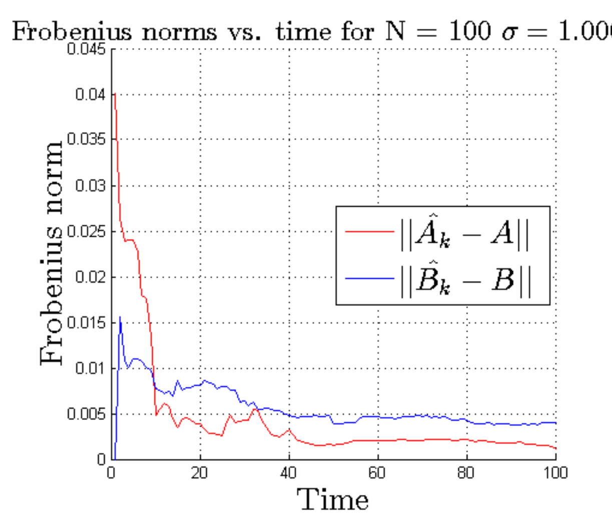

State vs. time for N = 10000 $\sigma = 0.01000$
State
Time
x_1
x_2
x_3

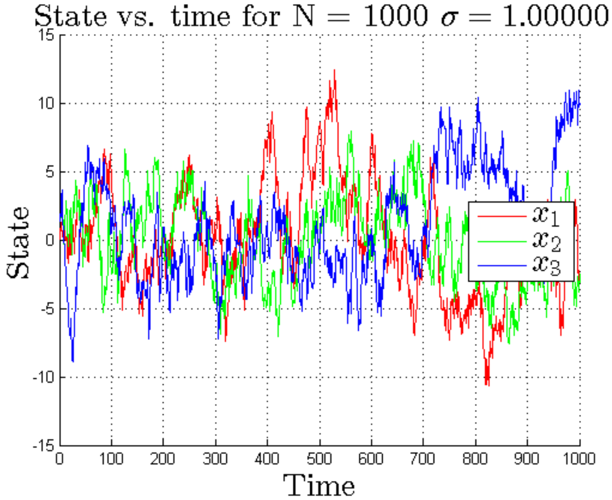

Control vs. time for N = 10000 $\sigma = 0.0100$
Control
Time
u_1
u_2
u_3

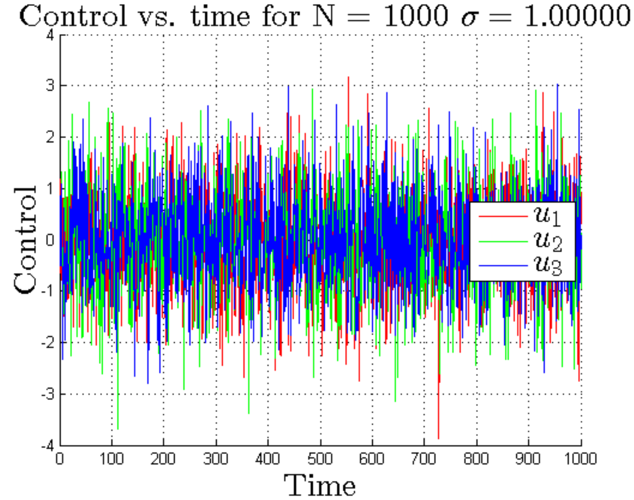

Frobenius norms vs. time for N = 10000 $\sigma = 0.01$
Frobenius norm
Time
$||\hat{A}_k - A||$
$||\hat{B}_k - B||$

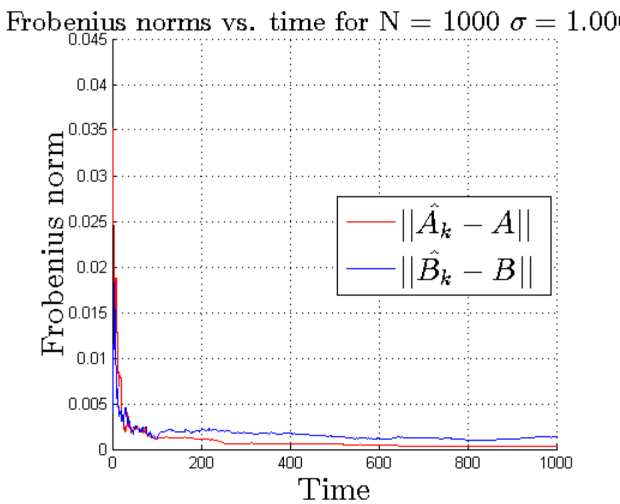

State vs. time for N = 100 $\sigma = 1.00000$
State
Time
x_1
x_2
x_3

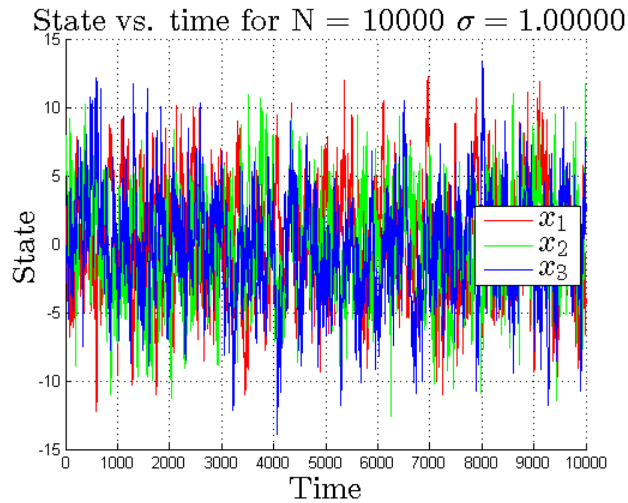

Control vs. time for N = 100 $\sigma = 1.00000$
Control
Time
u_1
u_2
u_3

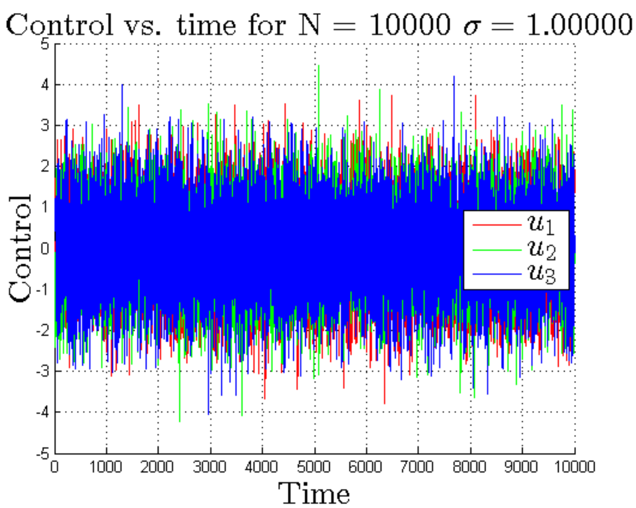

Frobenius norms vs. time for N = 100 $\sigma = 1.00$
Frobenius norm
Time
$||\hat{A}_k - A||$
$||\hat{B}_k - B||$

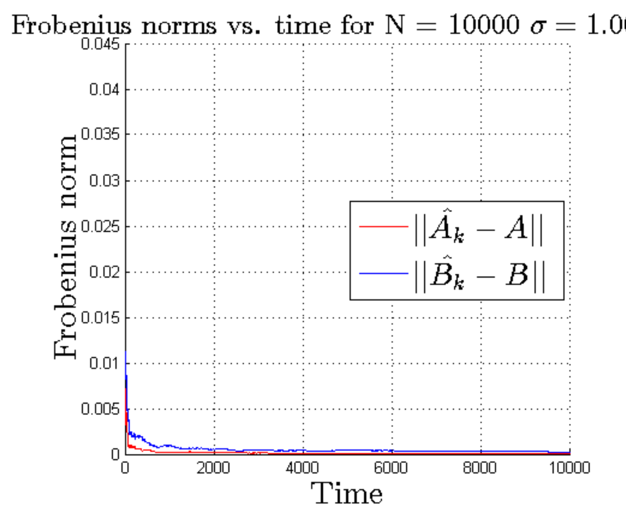

State vs. time for N = 1000 σ = 1.00000
State
Time
x_1
x_2
x_3


Control vs. time for N = 1000 σ = 1.00000
Control
Time
u_1
u_2
u_3


Frobenius norms vs. time for N = 1000 σ = 1.00
Frobenius norm
Time
$||\hat{A}_k - A||$
$||\hat{B}_k - B||$


State vs. time for N = 10000 σ = 1.00000
State
Time
x_1
x_2
x_3


Control vs. time for N = 10000 σ = 1.00000
Control
Time
u_1
u_2
u_3


Frobenius norms vs. time for N = 10000 σ = 1.0
Frobenius norm
Time
$||\hat{A}_k - A||$
$||\hat{B}_k - B||$