

HW 8 Question 1

Monday, June 3, 2019 12:40 PM

$$x(t+1) = ax(t) + bx(t-1) + w(t)$$

Part a

$$[x(t)]_{t=0}^N \text{ known}$$

$$\min_{a, b} \sum_{t=1}^{N-1} (\bar{x}(t+1) - x(t+1))^2$$

$$\begin{aligned} \bar{x}(t+1) &= E(ax(t) + bx(t-1) + w(t)) \\ &= ax(t) + bx(t-1) \end{aligned}$$

$$\min_{a, b} \sum_{t=1}^{N-1} (x(t+1) - ax(t) - bx(t-1))$$

$$\underbrace{\begin{bmatrix} x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} x_1 & x_0 \\ x_2 & x_1 \\ \vdots & \vdots \\ x_{N-1} & x_{N-2} \end{bmatrix}}_H \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{\theta} + \underbrace{w}_{N(0, \sigma^2)}$$

$$\min_{a, b} \|Y - H\theta\|$$

\Rightarrow if H is full rank

$$\theta^* = (H^T H^{-1}) H^T y$$

if H is not full rank,
we can use MATLAB's left division

$$\theta^* = H \setminus y$$

Parts b and c

Monday, June 3, 2019 7:00 PM

Part b

See code

N = 10 : Least squares estimate of (a,b) is mean (0.92, -0.14) with std dev (0.34, 0.40)

N = 100 : Least squares estimate of (a,b) is mean (1.01, -0.13) with std dev (0.11, 0.11)

N = 1000 : Least squares estimate of (a,b) is mean (1.00, -0.10) with std dev (0.03, 0.03)

We can see that as the amount of data used increases (i.e. N increases),
our estimates for a and b get better.

The mean gets closer to the true value and the standard deviation decreases.

This is in accordance with what we expect which is that more data will
result in a better estimate.

True values of a and b are (1.00, -0.10)

Part c

See code

N = 10 : Least squares estimate of (a,b) is mean (0.98, -0.20) with std dev (0.85, 1.08)

N = 100 : Least squares estimate of (a,b) is mean (0.83, -0.01) with std dev (0.65, 0.87)

N = 1000 : Least squares estimate of (a,b) is mean (0.99, -0.22) with std dev (0.47, 0.36)

These estimates are worse than the estimates from part b.

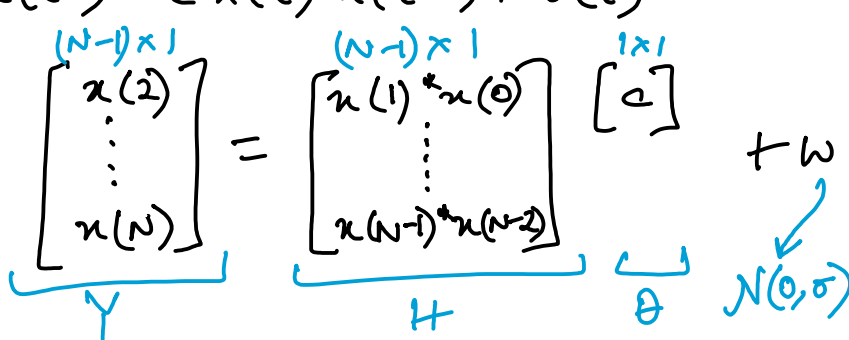
The mean is further from the true values and the standard deviation is higher.

This makes sense because the noise is no longer independent between each sample,
which adversely affects the quality of our estimate.

True values of a and b are (1.00, -0.10)

Part d

Monday, June 3, 2019 6:21 PM

$$x(t+1) = c x(t) x(t-1) + w(t)$$


Matrix representation of the equation above:

$\begin{bmatrix} x(2) \\ \vdots \\ x(N) \end{bmatrix} = \begin{bmatrix} x(1) * x(0) \\ \vdots \\ x(N-1) * x(N-2) \end{bmatrix} \begin{bmatrix} c \end{bmatrix} + w$

Dimensions: $(N-1) \times 1$ for Y , $(N-1) \times 1$ for H , and 1×1 for θ . The noise term w is noted as $N(0, \sigma)$.

$$\min_{\theta} \|Y - H\theta\|$$

$$\theta^* = H \backslash Y \quad \text{still applies!}$$

The system is linear in θ and we can compute the matrices Y & H from known data so the linear least squares approach still works.

See code:

N = 10 : Least squares estimate of c is mean 0.07 with std dev 0.44

N = 100 : Least squares estimate of c is mean 0.09 with std dev 0.09

N = 1000 : Least squares estimate of c is mean 0.10 with std dev 0.03

Again, we see that having more data makes our estimates better (closer to true value of $c = 0.10$ and smaller standard deviation).

Question 1 code

Contents

- [AA 203 HW 8 Question 1](#)
- [Part b](#)
- [Part c](#)
- [Part d](#)

AA 203 HW 8 Question 1

Somrita Banerjee

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

Part b

```
x0 = 1;
x1 = 0.5;
a_true = 1;
b_true = -0.1;
num_trials = 100;
N_values = [10, 100, 1000];
for N = N_values
    theta_estimates = zeros(2, num_trials);
    for nt=1:num_trials
        x = zeros(N+1, 1);
        for i=0:N
            if i == 0
                x(i+1) = x0;
            elseif i == 1
                x(i+1) = x1;
            else
                x(i+1) = a_true*x(i) + b_true*x(i-1) + normrnd(0,1);
            end
        end
        Y = x(3:end);
        H = [x(2:end-1) x(1:end-2)];
        theta = H\Y;
        theta_estimates(:,nt) = theta;
    end
    lse = mean(theta_estimates,2);
    stddev = std(theta_estimates,0,2);
    fprintf('N = %d : Least squares estimate of (a,b) is mean (%.2f, %.2f) with std dev (%.2f, %.2f)\n',N, lse(1),lse(2), stddev(1), stddev(2));
end
fprintf('We can see that as the amount of data used increases (i.e. N increases),\n our estimates for a and b get better. \n The mean gets closer to the true value
```

```
N = 10 : Least squares estimate of (a,b) is mean (0.92, -0.14) with std dev (0.34, 0.40)
N = 100 : Least squares estimate of (a,b) is mean (1.01, -0.13) with std dev (0.11, 0.11)
N = 1000 : Least squares estimate of (a,b) is mean (1.00, -0.10) with std dev (0.03, 0.03)
We can see that as the amount of data used increases (i.e. N increases),
our estimates for a and b get better.
The mean gets closer to the true value and the standard deviation decreases.
This is in accordance with what we expect which is that more data will
result in a better estimate.
True values of a and b are (1.00, -0.10)
```

Part c

```
for N = N_values
    theta_estimates = zeros(2, num_trials);
    for nt=1:num_trials
        x = zeros(N+1, 1);
        noise = zeros(N+1,1);
        for i=0:N
            if i == 0
                x(i+1) = x0;
            elseif i == 1
                x(i+1) = x1;
            else
                if i == 2
                    noise(i) = normrnd(0,1);
                else
                    noise(i) = normrnd(0, sqrt(2*abs(noise(i-1)))));
                end
                x(i+1) = a_true*x(i) + b_true*x(i-1) + noise(i);
            end
        end
        Y = x(3:end);
        H = [x(2:end-1) x(1:end-2)];
        theta = H\Y;
        theta_estimates(:,nt) = theta;
```

```

end
lse = mean(theta_estimates,2);
stddev = std(theta_estimates,0,2);
fprintf('N = %d : Least squares estimate of (a,b) is mean (%.2f, %.2f) with std dev (%.2f, %.2f)\n',N, lse(1),lse(2), stddev(1), stddev(2));
end
fprintf('These estimates are worse than the estimates from part b. \n The mean is further from the true values and the standard deviation is higher. \n This makes s

```

```

N = 10 : Least squares estimate of (a,b) is mean (0.92, -0.16) with std dev (0.49, 0.46)
N = 100 : Least squares estimate of (a,b) is mean (0.99, -0.10) with std dev (0.18, 0.16)
N = 1000 : Least squares estimate of (a,b) is mean (0.98, -0.09) with std dev (0.06, 0.05)
These estimates are worse than the estimates from part b.
The mean is further from the true values and the standard deviation is higher.
This makes sense because the noise is no longer independent between each sample,
which adversely affects the quality of our estimate.
True values of a and b are (1.00, -0.10)

```

Part d

```

x0 = 1;
x1 = 0.5;
c_true = 0.1;
num_trials = 100;
N_values = [10, 100, 1000];
fprintf('Sample independent noise\n')
for N = N_values
    theta_estimates = zeros(1, num_trials);
    for nt=1:num_trials
        x = zeros(N+1, 1);
        for i=0:N
            if i == 0
                x(i+1) = x0;
            elseif i == 1
                x(i+1) = x1;
            else
                x(i+1) = c_true*x(i)*x(i-1) + normrnd(0,1);
            end
        end
        Y = x(3:end);
        H = [x(2:end-1).*x(1:end-2)];
        theta = H\Y;
        theta_estimates(:,nt) = theta;
    end
    lse = mean(theta_estimates,2);
    stddev = std(theta_estimates,0,2);
    fprintf('N = %d : Least squares estimate of c is mean %.2f with std dev %.2f\n',N, lse, stddev);
end

```

```

Sample independent noise
N = 10 : Least squares estimate of c is mean 0.07 with std dev 0.44
N = 100 : Least squares estimate of c is mean 0.09 with std dev 0.09
N = 1000 : Least squares estimate of c is mean 0.10 with std dev 0.03

```

Question 2

Monday, June 3, 2019 7:30 PM

$$\dot{y}(t) + a y(t) = b u(t) \quad a, b \text{ unknown}$$

$$\dot{y}_m(t) + a_m y_m(t) = b_m r(t)$$

Part a

$$u(t) = k_r(t) r(t) + k_y(t) y(t)$$

Dropping dep on t for clarity

$$\Rightarrow \dot{y} + a y = b u = b (k_r r + k_y y)$$

$$\Rightarrow \dot{y} + (a - b k_y) y = b k_r r$$

To match reference model, we choose

$$a - b k_y = a_m \quad \text{and} \quad b k_r = b_m$$

$$\Rightarrow \boxed{k_y^* = \frac{a - a_m}{b} \quad \text{and} \quad k_r^* = \frac{b_m}{b}}$$

Part b

$$e(t) := y(t) - y_m(t)$$

$$\delta_r(t) := k_r(t) - k_r^*$$

$$\delta_y(t) := k_y(t) - k_y^*$$

$$\dot{e}(t) = \dot{y}(t) - \dot{y}_m(t)$$

$$= b u(t) - a y(t) - b_m r(t) + a_m y_m(t)$$

$$= b (k_r(t) r(t) + k_y(t) y(t)) - a y(t)$$

$$\begin{aligned} & \xrightarrow{\delta_r + k_r^*} + a_m y_m(t) \xrightarrow{\delta_y + k_y^*} - b_m r(t) \\ & \quad y_m = y - e \end{aligned}$$

(dropping dep on t for clarity)

$$\begin{aligned} \Rightarrow \dot{e} &= b \left((\delta_r + \frac{b_m}{b}) r + (\delta_y + \frac{a - a_m}{b}) y \right) - a y \\ & \quad + a_m y - a_m e - b_m r \\ &= b \delta_r r + \cancel{b_m r} + b \delta_y y + \cancel{(a - a_m) y} \\ & \quad - \cancel{a y} + \cancel{a_m y} - a_m e - \cancel{b_m r} \end{aligned}$$

$$\Rightarrow \boxed{\dot{e}(t) + a_m e(t) = b \delta_r(t) r(t) + b \delta_y(t) y(t)}$$

Part c

$$\text{Let } x := (e, \delta_r, \delta_y)$$

$$V(x) = \frac{1}{2} e^2 + \frac{|b|}{2r} (\delta_r^2 + \delta_y^2)$$

$$\Rightarrow \dot{V} = e \dot{e} + \frac{|b|}{2r} (2\delta_r \dot{\delta}_r + 2\delta_y \dot{\delta}_y)$$

$$\delta_r = k_r(t) - k_r^* \Rightarrow \dot{\delta}_r = \dot{k}_r = -\text{sgn}(b) \text{rel}(t) r(t)$$

$$\delta_y = k_y(t) - k_y^* \Rightarrow \dot{\delta}_y = \dot{k}_y = -\text{sgn}(b) \text{rel}(t) y(t)$$

$$\begin{aligned} \Rightarrow \dot{V} &= e(-a_m e + b\delta_r r + b\delta_y y) \\ &\quad + \frac{|b|}{r} (\delta_r (-\text{sgn}(b) \text{rel}(t) r) + \delta_y (-\text{sgn}(b) \text{rel}(t) y)) \\ &= -a_m e^2 + e b (\delta_r r + \delta_y y) \\ &\quad - \frac{|b|}{r} \text{sgn}(b) e (\delta_r r + \delta_y y) \end{aligned}$$

$$|b| \text{sgn}(b) = b \quad \forall b$$

$$\Rightarrow \dot{V} = -a_m e^2 + \cancel{e b (\delta_r r + \delta_y y)} - \cancel{e b (\delta_r r + \delta_y y)}$$

$$\Rightarrow \boxed{\dot{V} = -a_m e^2}$$

Consider Lyapunov Thm

① if $x := (e, \delta_r, \delta_y)$

then if $x=0$ ($e=0, \delta_r=0, \delta_y=0$)

$$\dot{e} = -a_m e + b\delta_r r + b\delta_y y = 0$$

$$\dot{\delta}_r = \dot{k}_r = -\text{sgn}(b) \text{rel}(t) r = 0$$

$$\dot{\delta}_y = \dot{k}_y = -\text{sgn}(b) \text{rel}(t) y = 0$$

$$\therefore \dot{x} = f(x, t) = 0 \quad @ \quad x=0$$

② $V(x) = \frac{1}{2} e^2 + \frac{|b|}{2r} (\delta_r^2 + \delta_y^2)$

is clearly continuous & differentiable

③ $V(x)$ is a sum of squares so it is positive definite

$$V(x) > 0 \quad \forall x \neq 0$$

$$V(x) = 0 \quad \forall x = 0$$

$$\begin{aligned}
 \textcircled{4} \quad \dot{V}(u) &= -a_m e^2 \quad a_m > 0 \\
 &\Rightarrow \dot{V}(u) < 0 \quad \forall e \neq 0 \\
 \text{and } \dot{V}(u) &= 0 \quad \forall e = 0 \\
 &\Rightarrow \dot{V}(u) \leq 0 \quad \forall u \neq 0 \quad \left[\begin{array}{l} \text{possible } e=0 \\ \delta_r \neq 0 \text{ or } e_y \neq 0 \end{array} \right] \\
 \text{and } \dot{V}(u) &= 0 \quad \forall u = 0
 \end{aligned}$$

$\therefore \dot{V}(u)$ is negative semi-definite.

So, we can say $u = (e, \delta_r, \delta_y) = 0$ is a stable point in the sense of Lyapunov.
 So $\|e(t)\|, \|\delta_r(t)\|, \|\delta_y(t)\|$ remain bounded $\forall t \in [0, \infty)$.

Consider Barbalat's Lemma.

$$\text{Let } g(t) = \dot{V}(u) = -a_m e(t)^2$$

① $g(t)$ is clearly differentiable $\dot{g} = -2a_m \dot{e}(t)$

② $g(t)$ has a finite limit as $t \rightarrow \infty$
 because $e(t)$ is bounded $\forall t$
 $\Rightarrow -a_m e(t)^2$ is bounded $\forall t$

③ $\dot{g}(t) = -2a_m \dot{e}(t)$ is uniformly continuous
 because

$$\ddot{g}(t) = -2a_m \ddot{e}(t) \text{ is bounded}$$

because

$$\begin{aligned}
 \ddot{e}(t) &= -a_m \dot{e} + b \delta_r \dot{r} + b \delta_r \ddot{r} + b \delta_y \dot{y} + b \delta_y \ddot{y} \\
 &= -a_m (-a_m e + b \delta_r r + b \delta_y y) + b (-\text{sgn}(b) \delta_r r) r \\
 &\quad + b \delta_r \dot{r} + b (-\text{sgn}(b) \delta_y y) y + b \delta_y \dot{y}
 \end{aligned}$$

Using Lyapunov, we showed e, δ_y, δ_r are bounded
 we know \dot{r}, \dot{y} are bounded (given to us)

$\therefore \ddot{e}(t)$ is bounded $\Rightarrow \ddot{g}(t)$ is bounded

Using Barbalat's, we can say

$$\dot{g}(t) = -2a_m \dot{e}(t) \rightarrow 0 \text{ as } t \rightarrow \infty$$

$$a_m > 0 \quad \therefore \boxed{e(t) \rightarrow 0 \text{ as } t \rightarrow \infty}$$

Part d

$$\delta = 2$$

Case 1: $r(t) = 4$ Case 2: $r(t) = 4 \sin(3t)$

$$\dot{y}_m(t) + 4y_m(t) = 4r(t) \quad \delta, y_m, k_r, k_y$$
$$\dot{y}(t) - y(t) = 3u(t) = 3(k_r(t)r(t) + k_y(t)y(t))$$

We know $\text{sgn}(b) = 1$ (b is positive)

$$\dot{k}_r(t) = -\text{sgn}(b) \delta e(t) r(t)$$
$$= -\delta (y(t) - y_m(t)) r(t)$$

$$\dot{k}_y(t) = -\text{sgn}(b) \delta e(t) y(t)$$
$$= -\delta (y(t) - y_m(t)) y(t)$$

Now we can define

$$x = \begin{bmatrix} y(t) \\ y_m(t) \\ k_r(t) \\ k_y(t) \end{bmatrix} \quad \text{and} \quad \dot{x} = \begin{bmatrix} \dot{y}(t) \\ \dot{y}_m(t) \\ \dot{k}_r(t) \\ \dot{k}_y(t) \end{bmatrix}$$

Question 2 part d code

Contents

- [AA 203 HW 8 Question 2](#)
- [Part d](#)

AA 203 HW 8 Question 2

Somrita Banerjee

```
clc
clear all
close all
```

Part d

x = [y ym kr ky]

```
tspan = [0 10];
a = -1; b = 3; am = 4; bm = 4;
krStar = bm/b;
kyStar = (a-am)/b;
xInit = [0 0 0 0];
options=odeset('RelTol',1e-3,'AbsTol',1e-6);

[t1,x1] = ode45(@(t,x) MRAC(t,x,1), tspan, xInit, options);
[t2,x2] = ode45(@(t,x) MRAC(t,x,2), tspan, xInit, options);

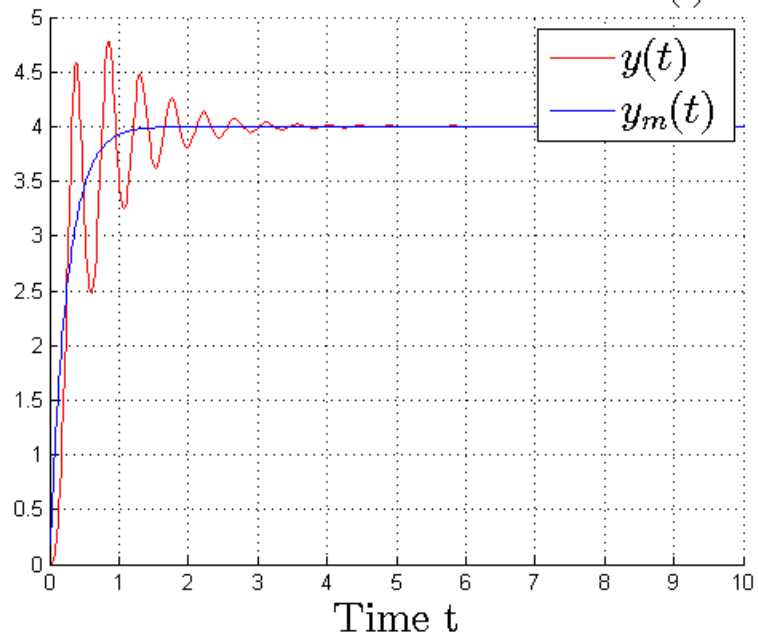
figure
hold on
plot(t1, x1(:,1), 'r')
plot(t1, x1(:,2), 'b')
legend({'$y(t)$','$y_m(t)$'}, 'Interpreter','latex','FontSize',20)
xlabel('Time t','Interpreter','latex','FontSize',20)
title('Evolution of true and reference model with $r(t) = 4$', 'Interpreter','latex','FontSize',15)
grid on

figure
hold on
plot(t1, x1(:,3), 'r')
plot(t1, krStar*ones(size(t1)), '--r')
plot(t1, x1(:,4), 'b')
plot(t1, kyStar*ones(size(t1)), '--b')
legend({'$k_r(t)$','$k_r^*$','$k_y(t)$','$k_y^*$'}, 'Interpreter','latex','FontSize',20)
xlabel('Time t','Interpreter','latex','FontSize',20)
title('Evolution of feedback gains with $r(t) = 4$', 'Interpreter','latex','FontSize',15)
grid on

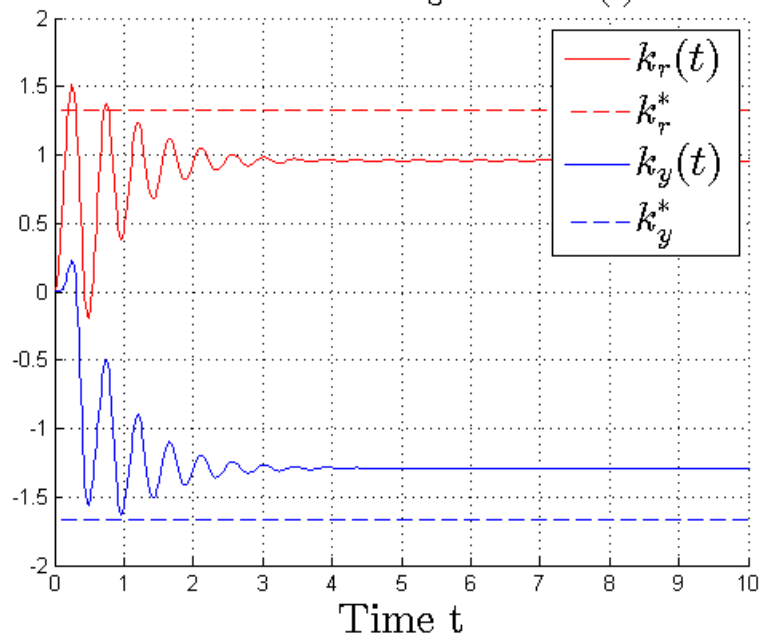
figure
hold on
plot(t2, x2(:,1), 'r')
plot(t2, x2(:,2), 'b')
legend({'$y(t)$','$y_m(t)$'}, 'Interpreter','latex','FontSize',20)
xlabel('Time t','Interpreter','latex','FontSize',20)
title('Evolution of true and reference model with $r(t) = 4\sin(3t)$', 'Interpreter','latex','FontSize',15)
grid on

figure
hold on
plot(t2, x2(:,3), 'r')
plot(t2, krStar*ones(size(t2)), '--r')
plot(t2, x2(:,4), 'b')
plot(t2, kyStar*ones(size(t2)), '--b')
legend({'$k_r(t)$','$k_r^*$','$k_y(t)$','$k_y^*$'}, 'Interpreter','latex','FontSize',20)
xlabel('Time t','Interpreter','latex','FontSize',20)
title('Evolution of feedback gains with $r(t) = 4\sin(3t)$', 'Interpreter','latex','FontSize',15)
grid on
```

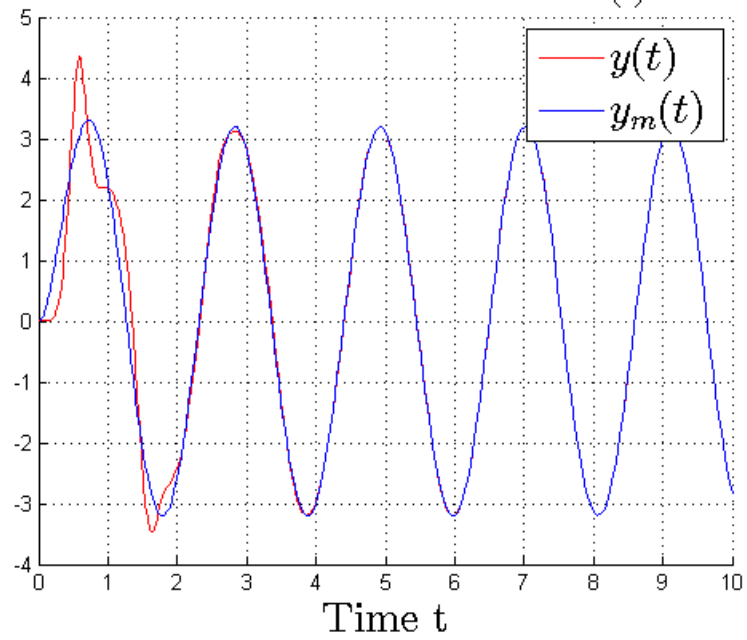
Evolution of true and reference model with $r(t) = 4$



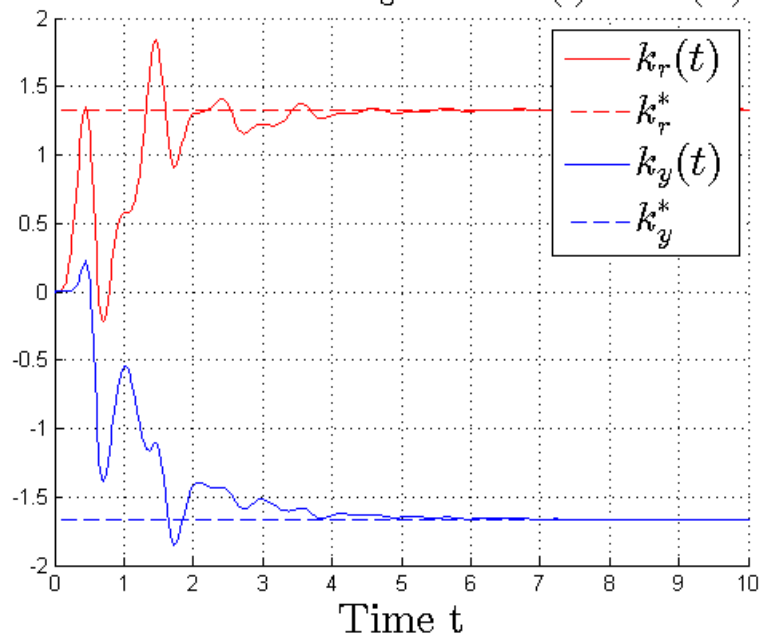
Evolution of feedback gains with $r(t) = 4$



Evolution of true and reference model with $r(t) = 4 \sin(3t)$



Evolution of feedback gains with $r(t) = 4 \sin(3t)$



```
function dxdt=MRAC(t,x,caseNum)
% x = [y ym kr ky]
if caseNum == 1
    r = 4;
elseif caseNum == 2
    r = 4*sin(3*t);
else
    fprintf('Error case number not recognized.\n');
end

y = x(1);
ym = x(2);
kr = x(3);
ky = x(4);

a = -1; b = 3; am = 4; bm = 4; gamma = 2;

u = kr * r + ky * y;
e = y - ym;

ydot = -a*y + b*u;
ymdot = -am*ym + bm*r;
krdot = -gamma * e * r;
kydot = -gamma * e * y;

dxdt = [ydot; ymdot; krdot; kydot];
end
```

Question 3

Contents

- [AA 203 HW 8 Question 3](#)
- [Part a - Plain LQR](#)
- [Part b- Certainty equivalent adaptive LQR controller](#)
- [Part c- adaptive LQR controller with white noise](#)

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

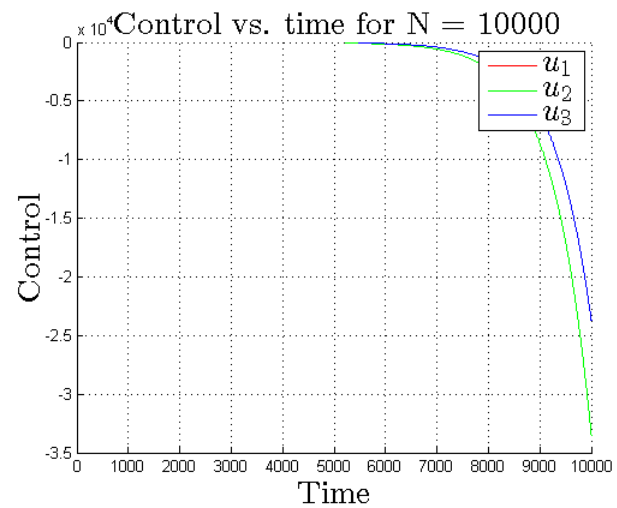
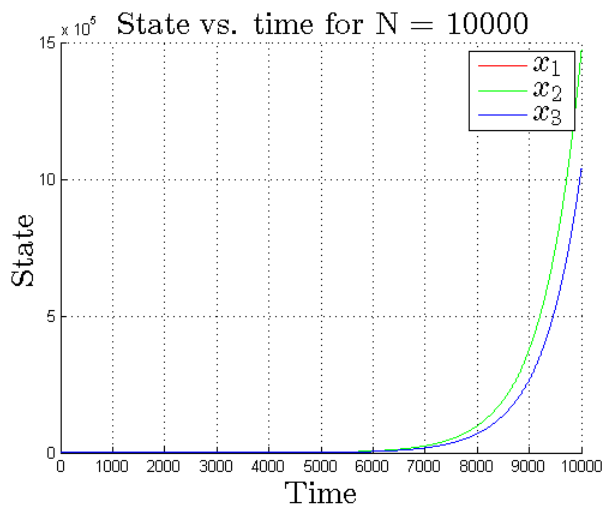
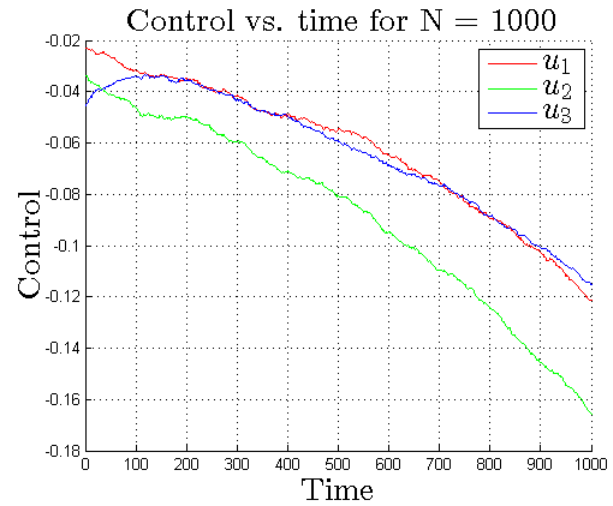
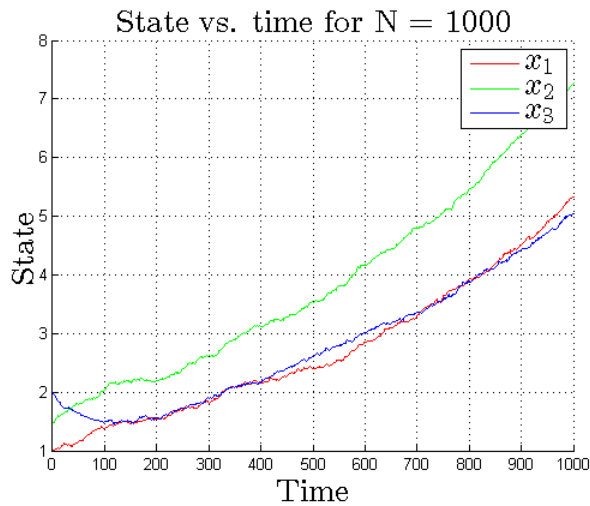
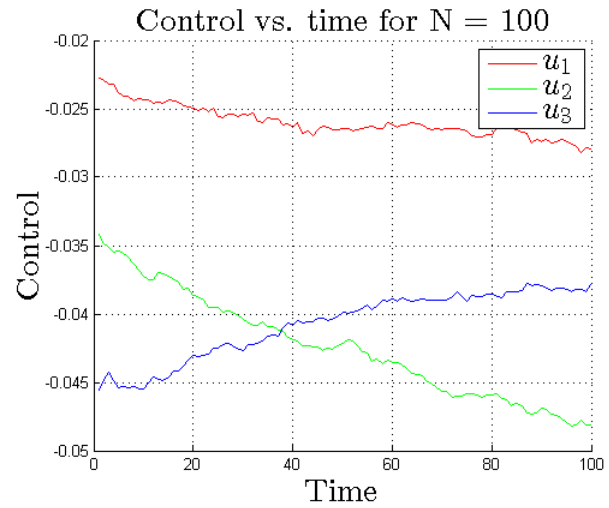
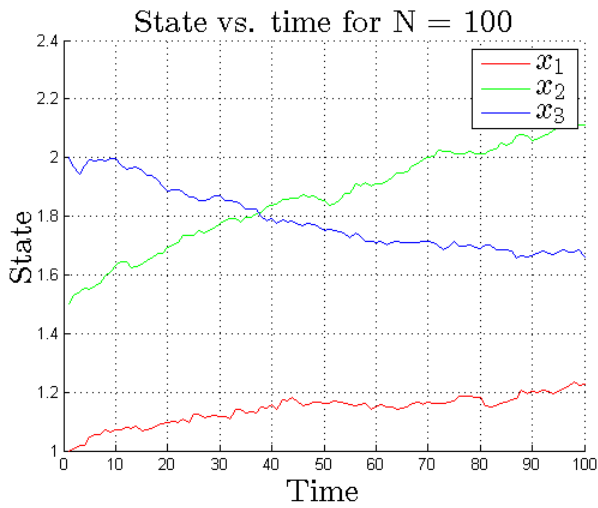
    fprintf('Cost for N = %d is %.2f \n',N, cost)

    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

```

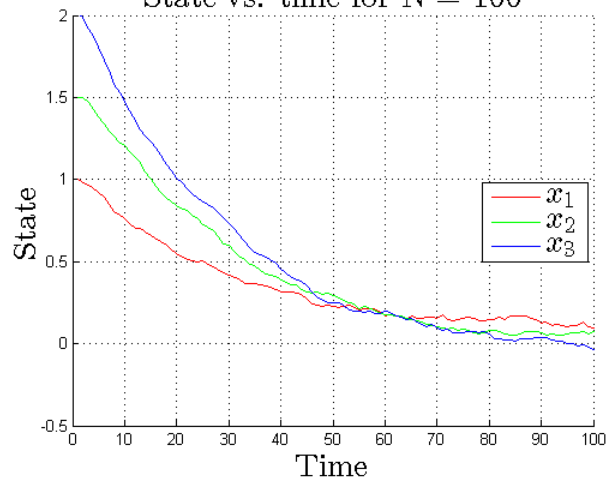
```

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

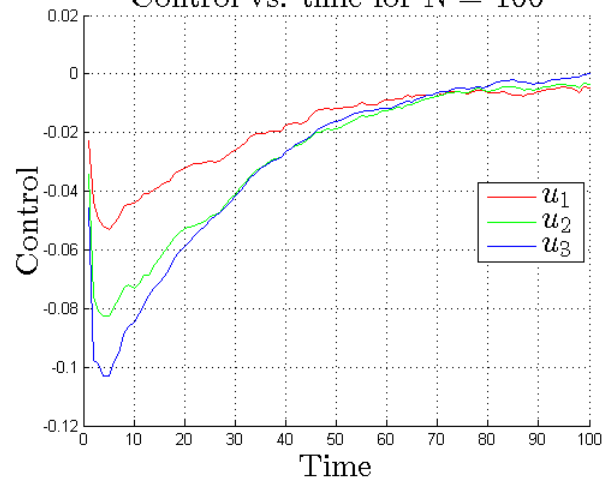
```

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.

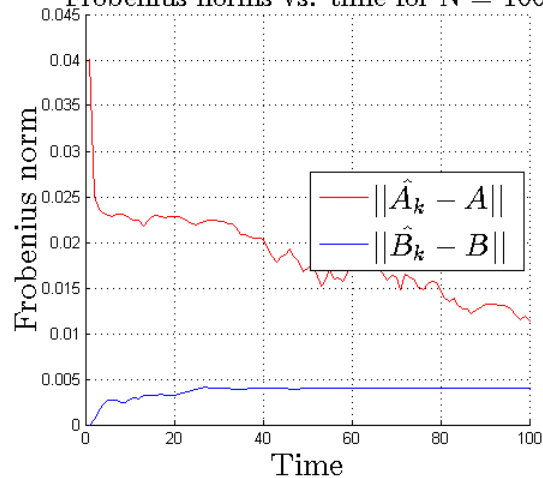
State vs. time for $N = 100$



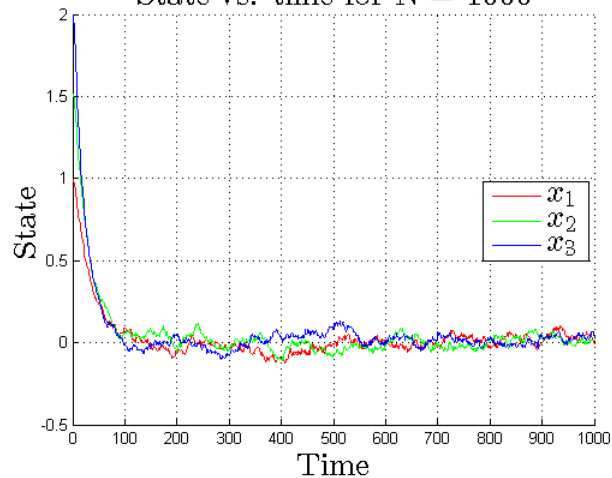
Control vs. time for $N = 100$



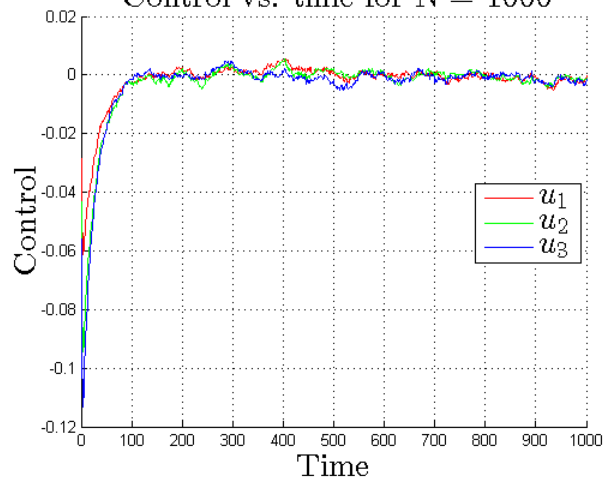
Frobenius norms vs. time for $N = 100$



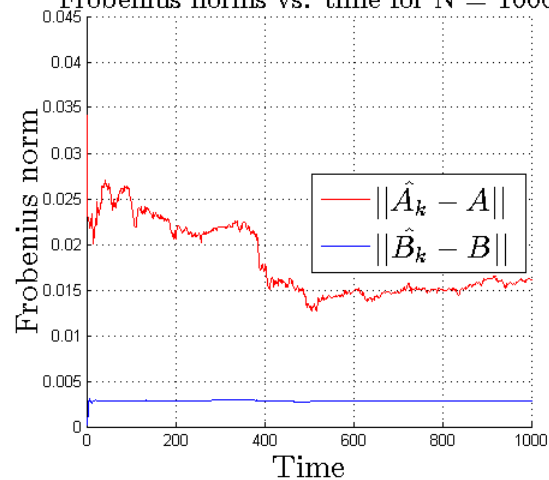
State vs. time for $N = 1000$

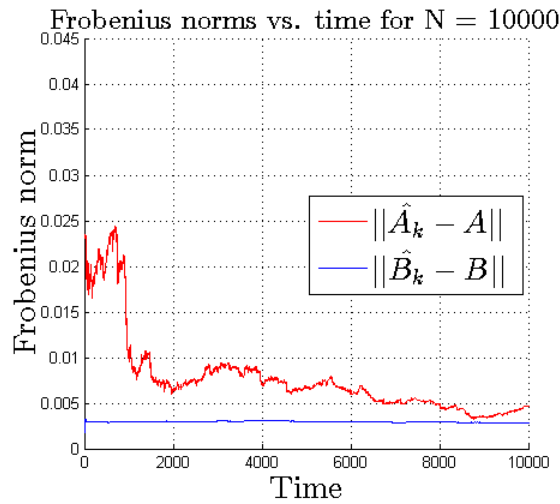
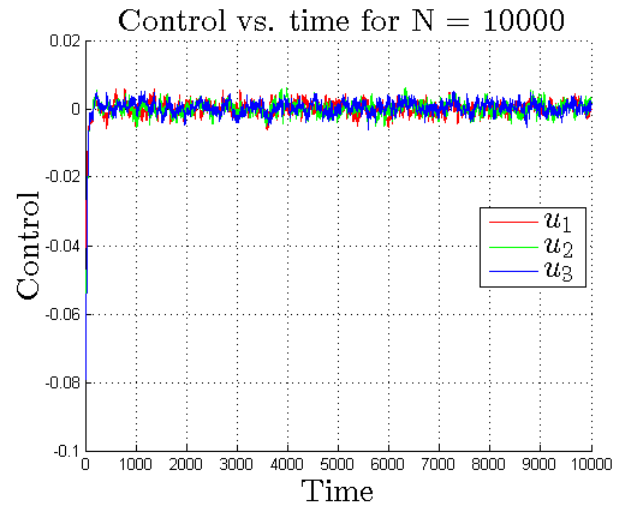
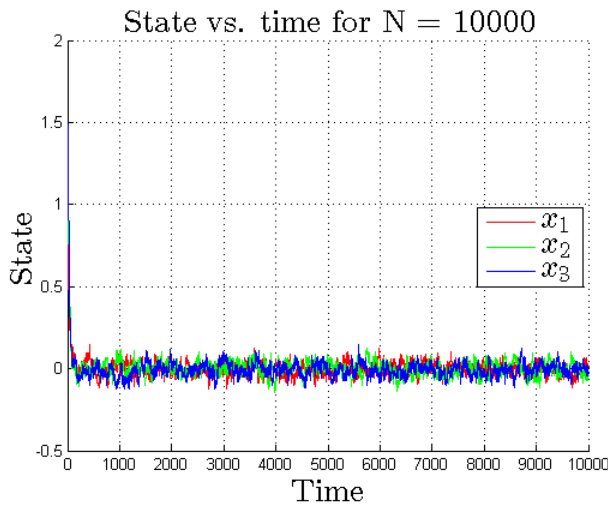


Control vs. time for $N = 1000$



Frobenius norms vs. time for $N = 1000$





Part c- adaptive LQR controller with white noise

```

N_vals = [100 1000 10000];
white_noise_stddev_vals = [0.000001 0.001 0.01 1];
for white_noise_stddev = white_noise_stddev_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(6,6,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_stddev, [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);
        end
    end
end

```

```

        % 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_stddev, 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_stddev);
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_stddev);
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_stddev);
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

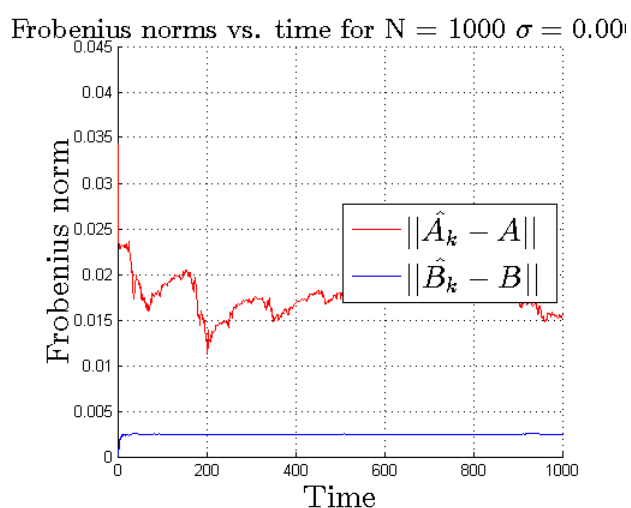
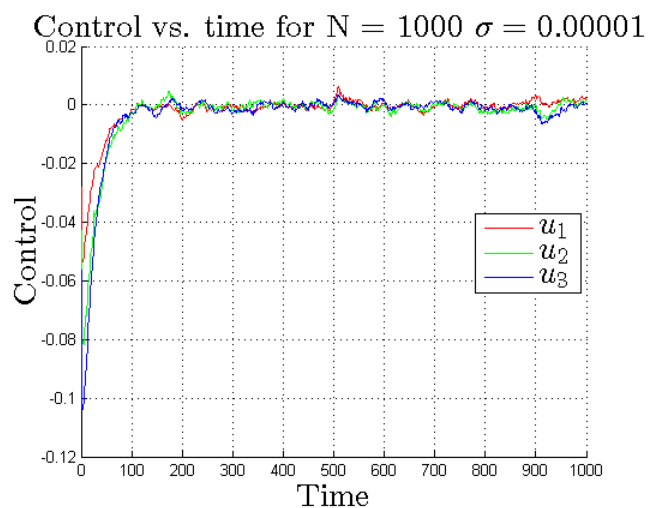
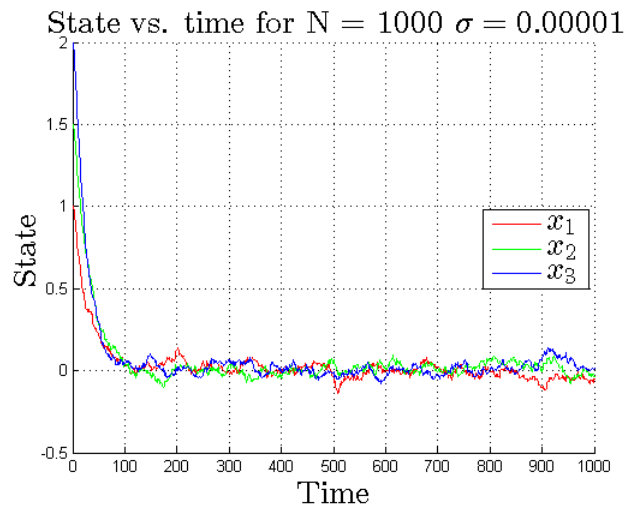
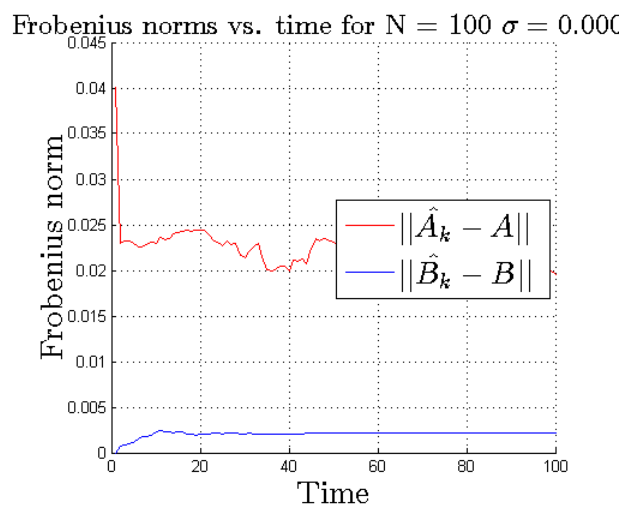
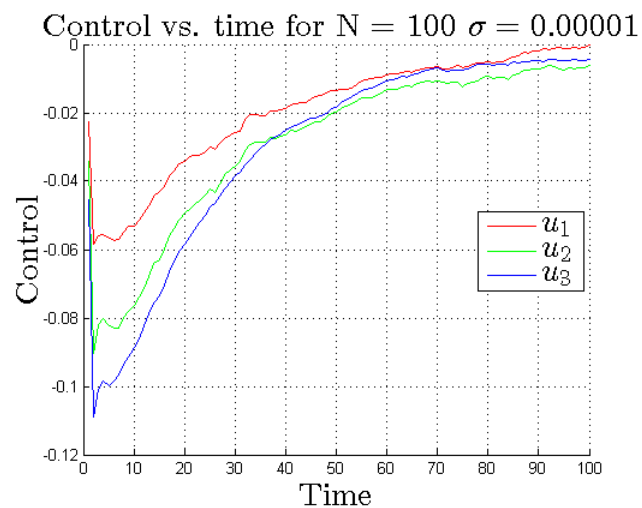
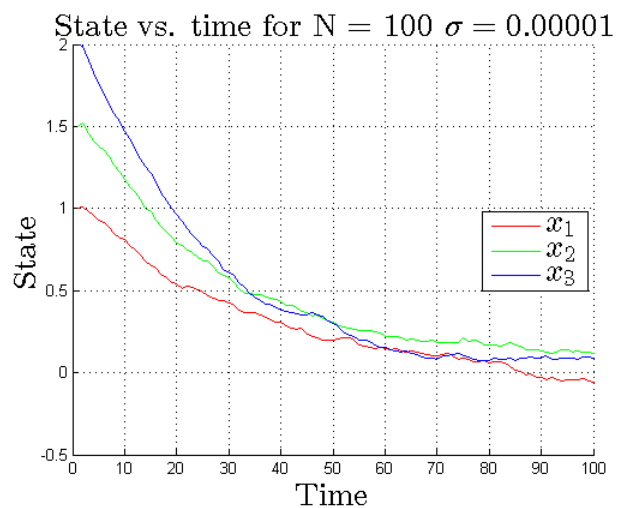
```

```

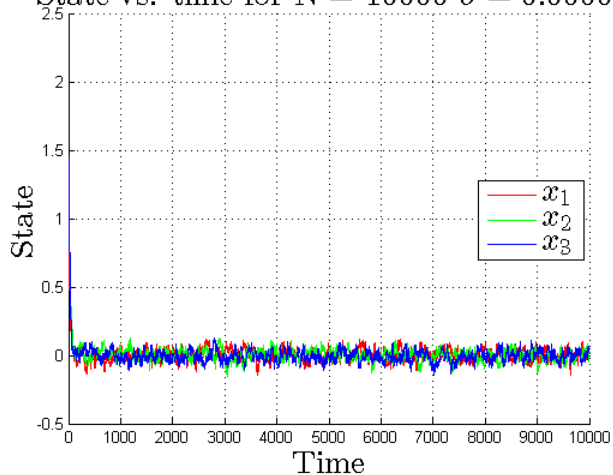
Cost for N = 100, white noise std dev = 0.00001 is 492.18
Cost for N = 1000, white noise std dev = 0.00001 is 482.81
Cost for N = 10000, white noise std dev = 0.00001 is 635.98
Cost for N = 100, white noise std dev = 0.00100 is 471.72
Cost for N = 1000, white noise std dev = 0.00100 is 495.86
Cost for 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

```

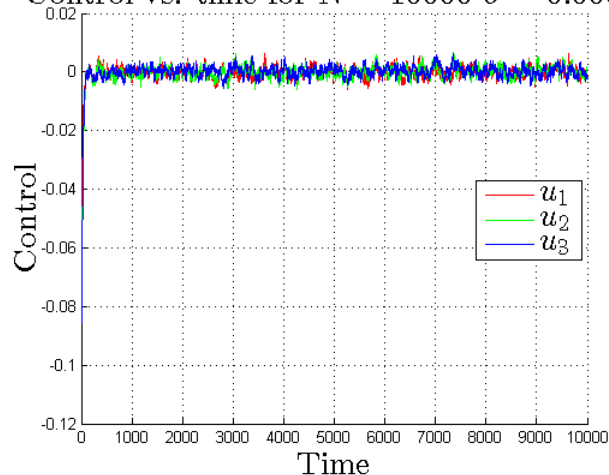
We notice a few interesting things after introducing white noise to the control. The first is that when the white noise standard deviation is very small, the performance is similar to the certainty equivalent LQR controller (just like we'd expect). As the white noise gets larger, the control input gets 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).



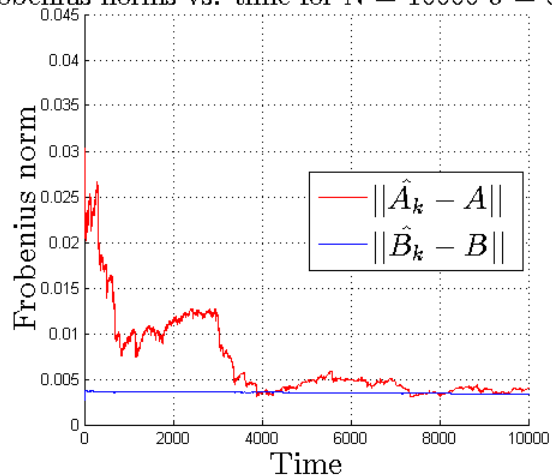
State vs. time for $N = 10000$ $\sigma = 0.00001$



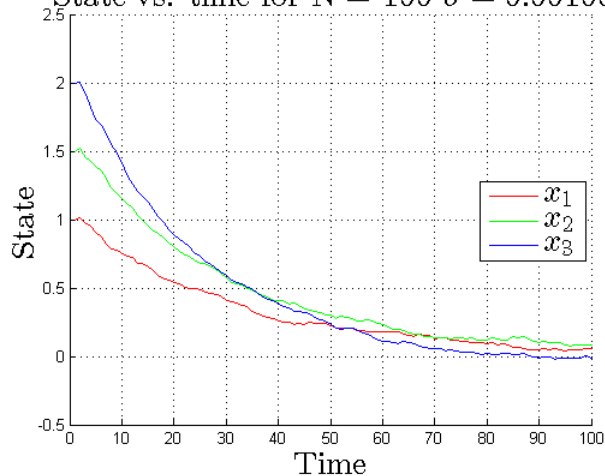
Control vs. time for $N = 10000$ $\sigma = 0.00001$



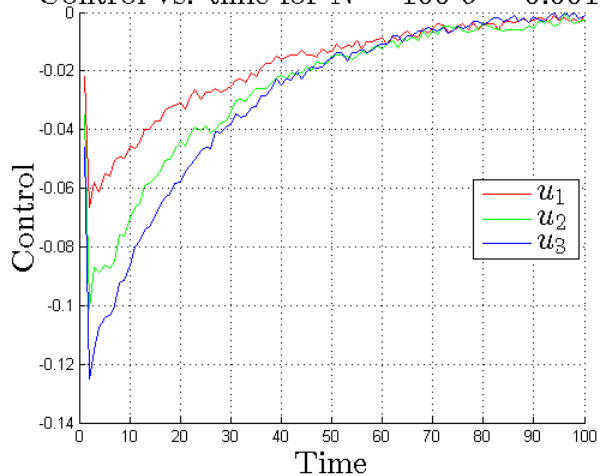
Frobenius norms vs. time for $N = 10000$ $\sigma = 0.00$



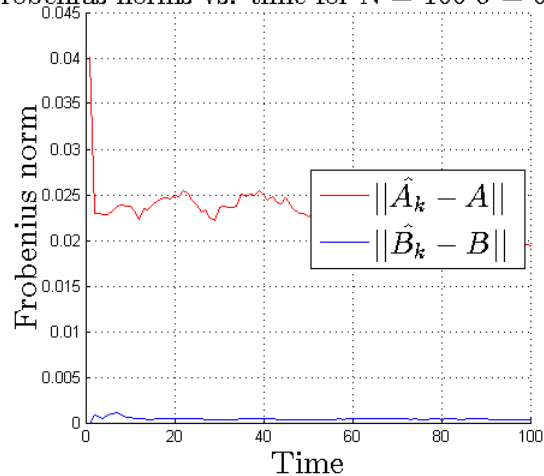
State vs. time for $N = 100$ $\sigma = 0.00100$



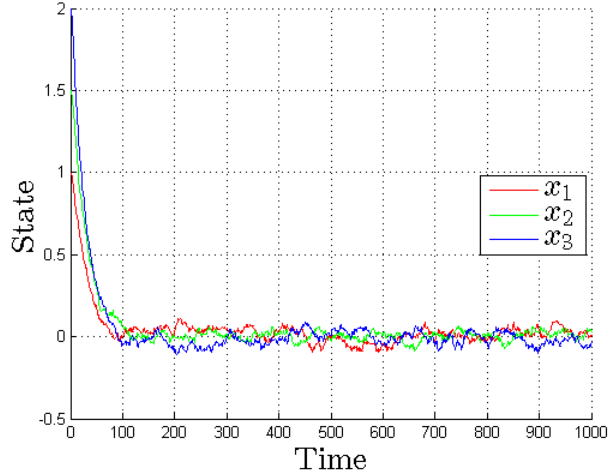
Control vs. time for $N = 100$ $\sigma = 0.00100$



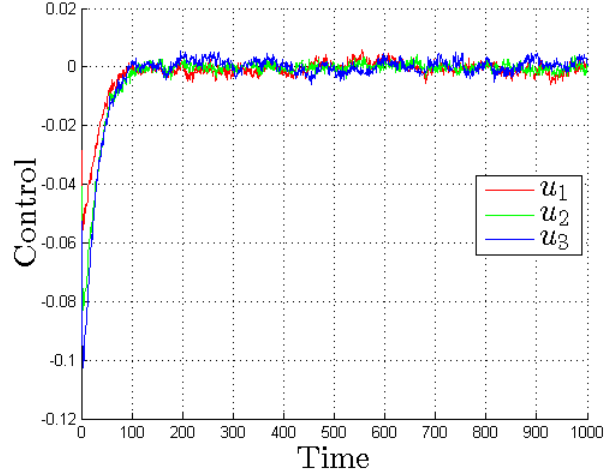
Frobenius norms vs. time for $N = 100$ $\sigma = 0.001$



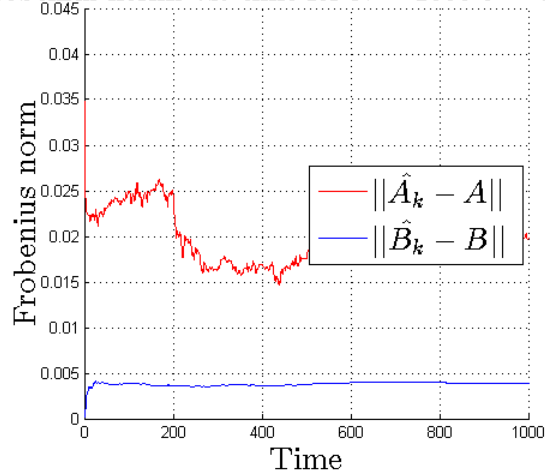
State vs. time for $N = 1000$ $\sigma = 0.00100$



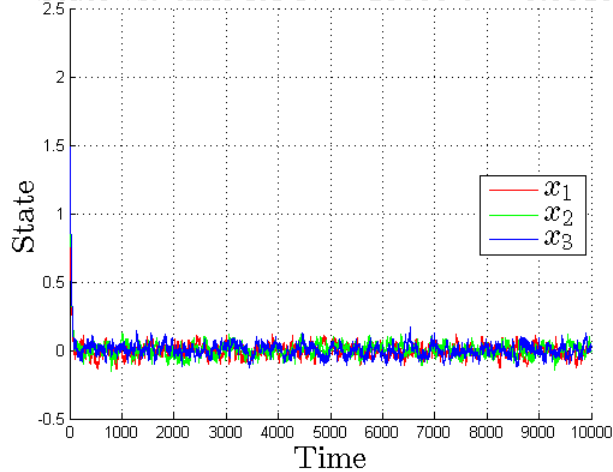
Control vs. time for $N = 1000$ $\sigma = 0.00100$



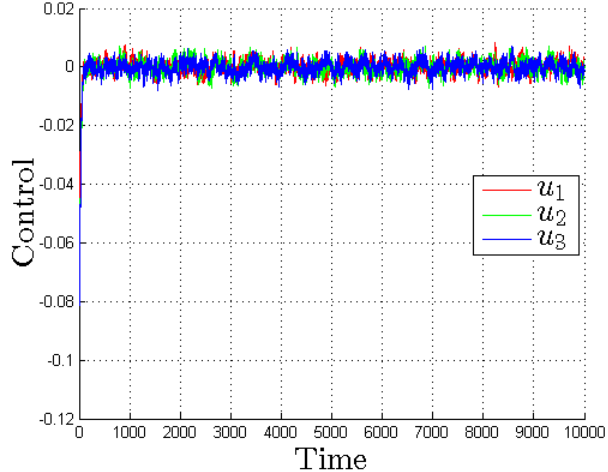
Frobenius norms vs. time for $N = 1000$ $\sigma = 0.00$



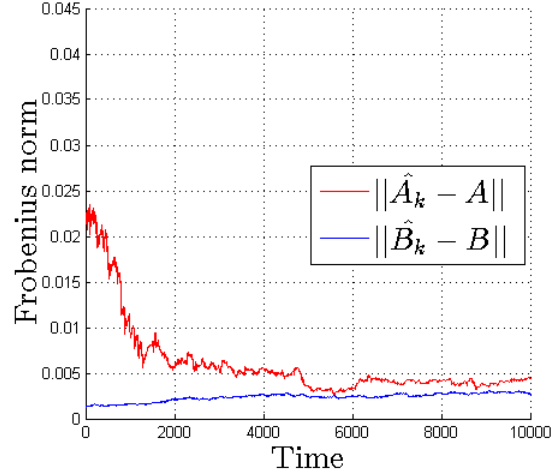
State vs. time for $N = 10000$ $\sigma = 0.00100$

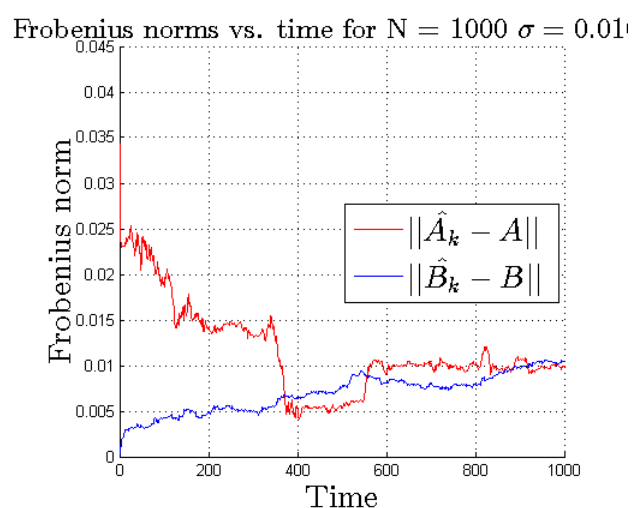
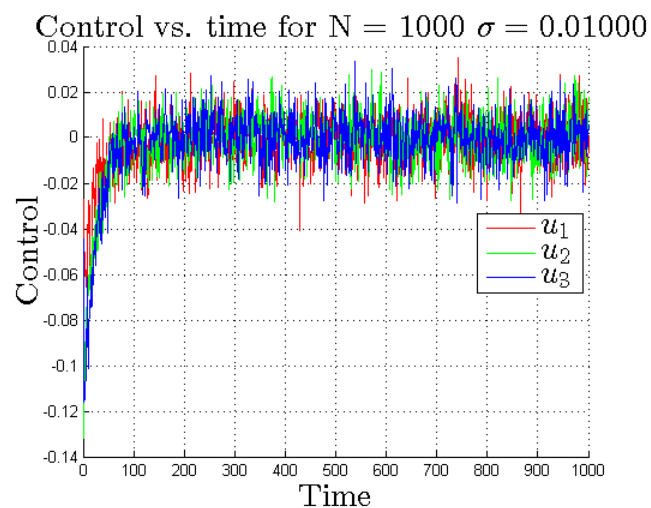
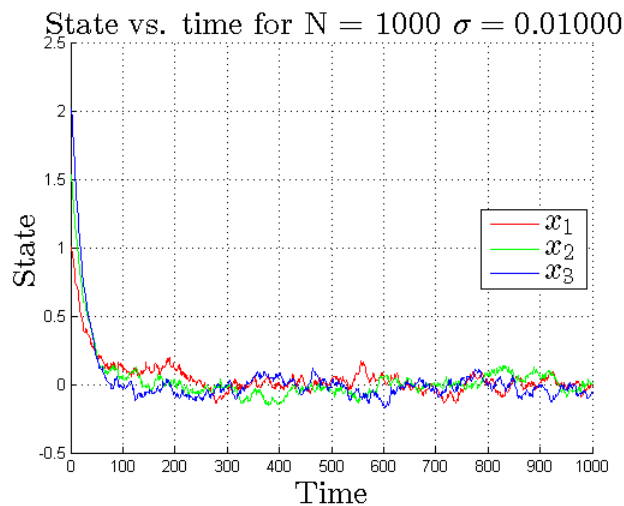
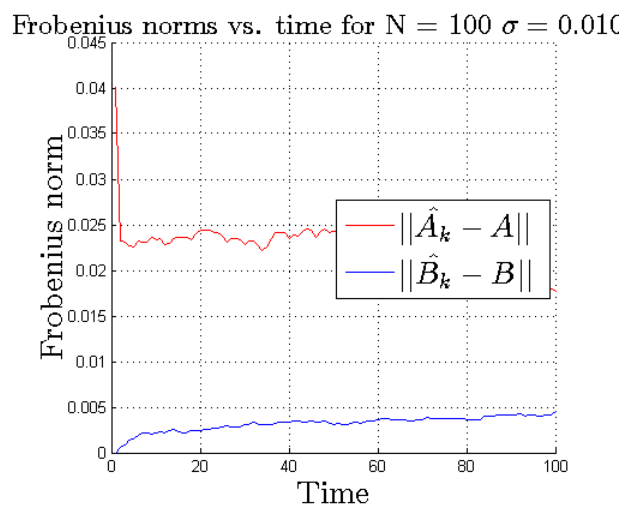
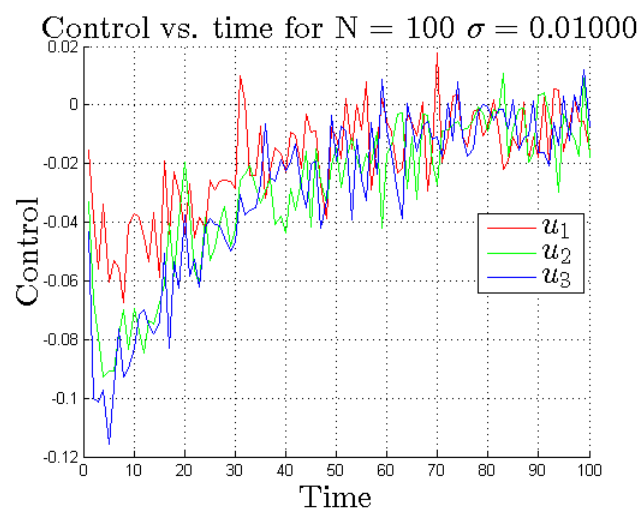
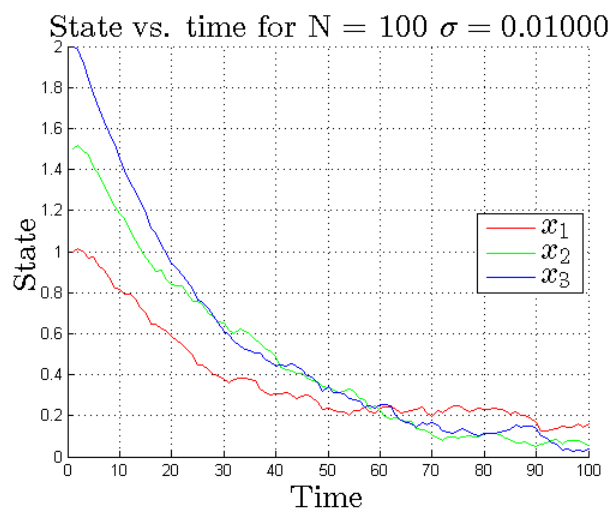


Control vs. time for $N = 10000$ $\sigma = 0.00100$

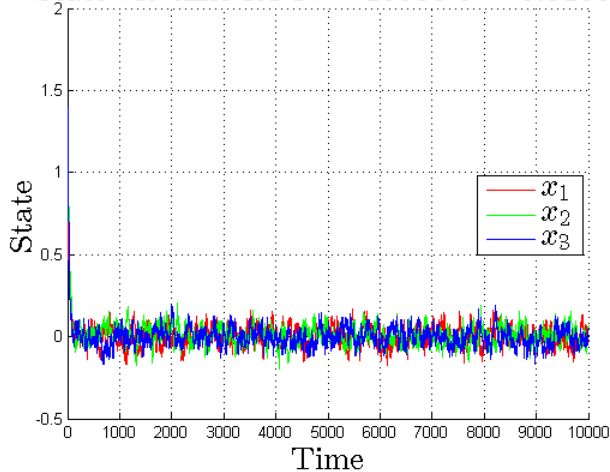


Frobenius norms vs. time for $N = 10000$ $\sigma = 0.00$

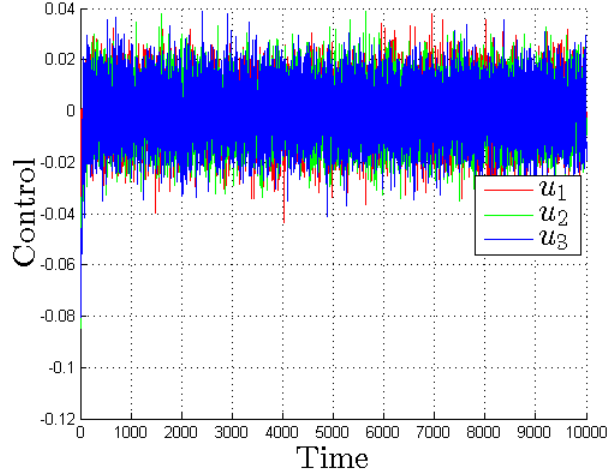




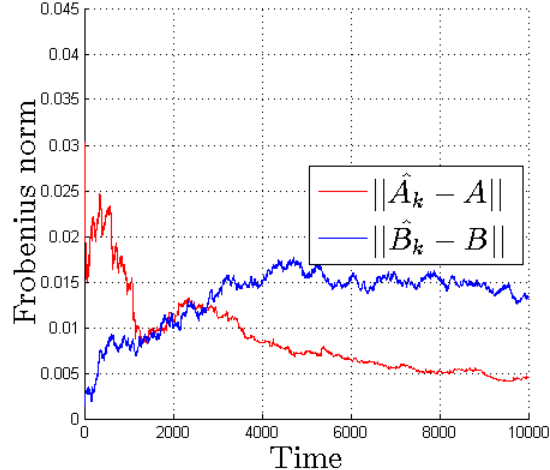
State vs. time for $N = 10000$ $\sigma = 0.01000$



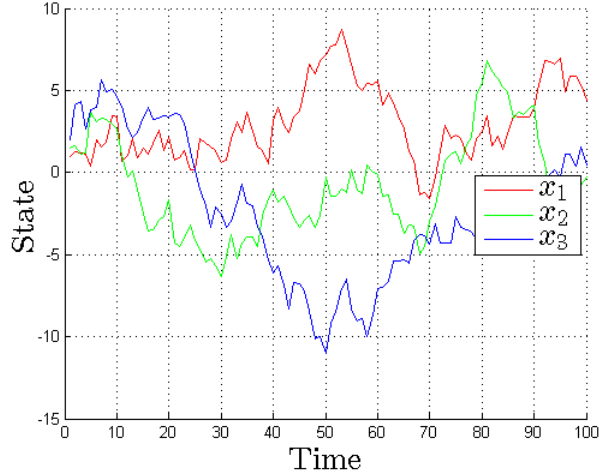
Control vs. time for $N = 10000$ $\sigma = 0.01000$



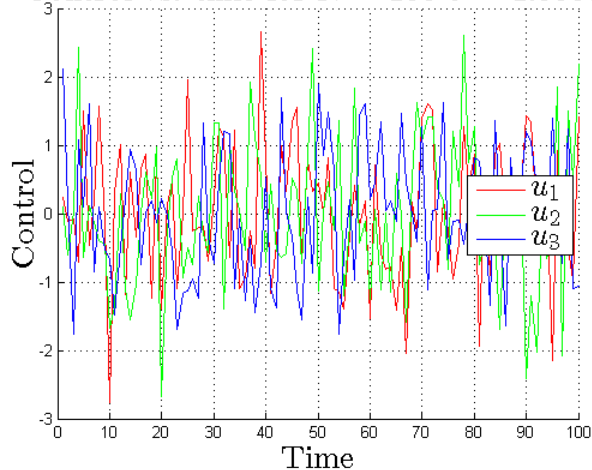
Frobenius norms vs. time for $N = 10000$ $\sigma = 0.01$



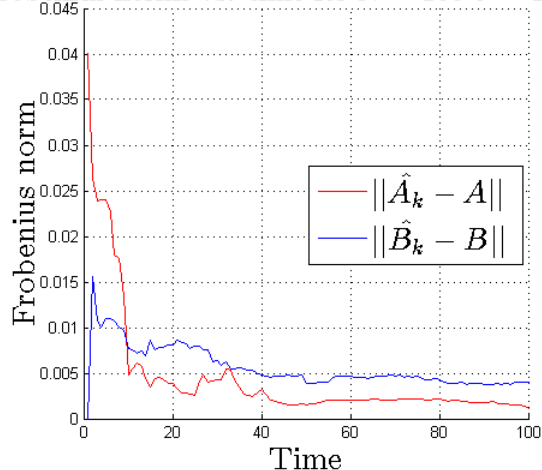
State vs. time for $N = 100$ $\sigma = 1.00000$

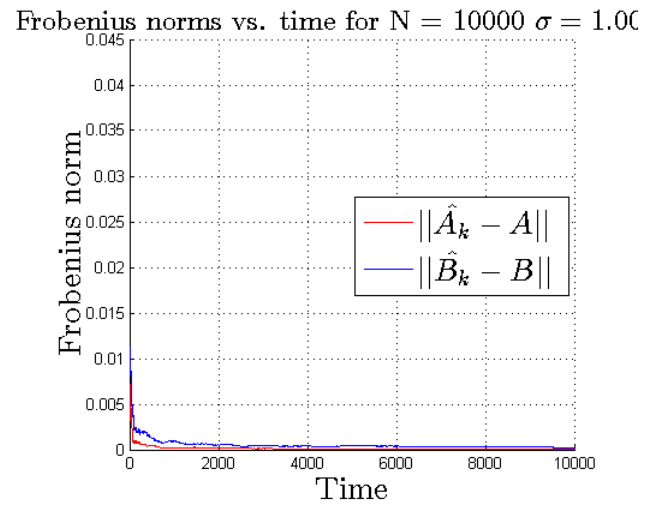
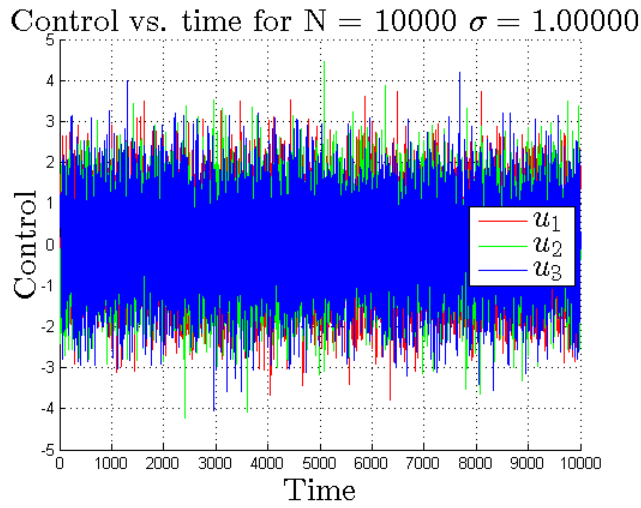
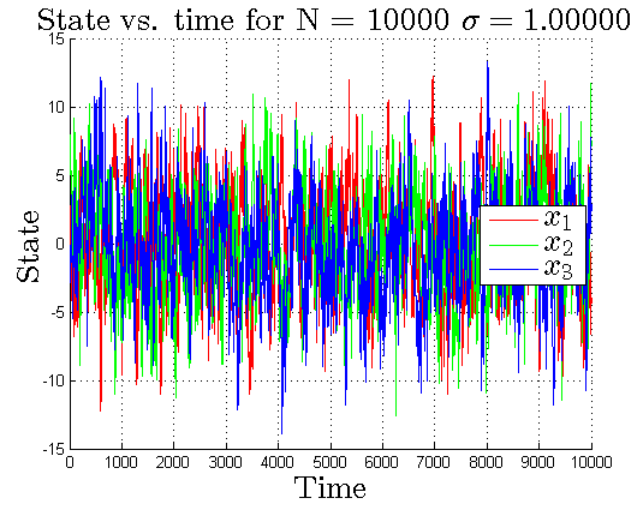
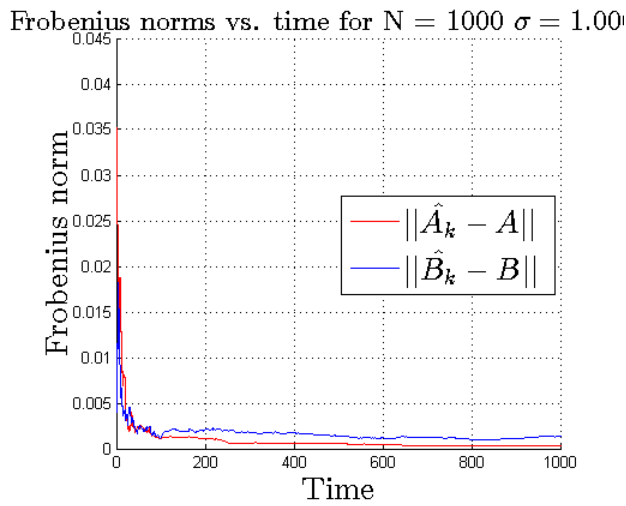
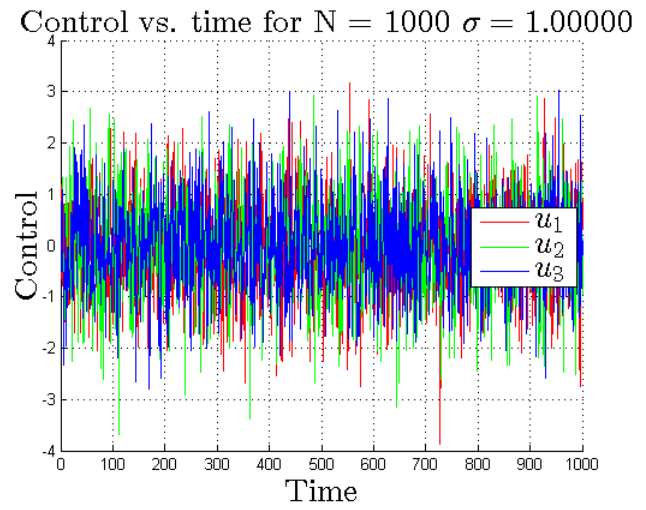
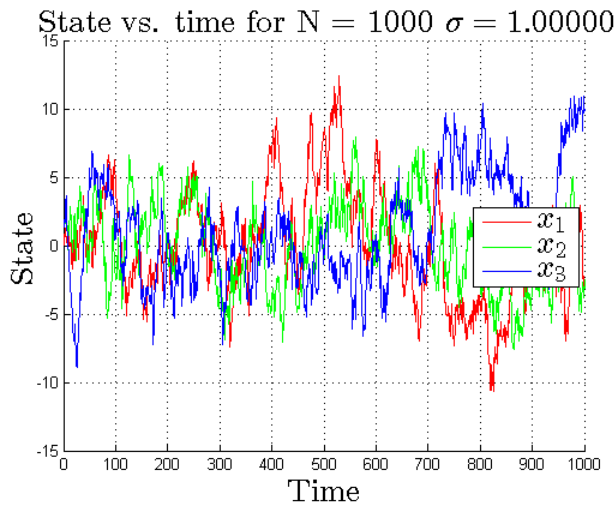


Control vs. time for $N = 100$ $\sigma = 1.00000$



Frobenius norms vs. time for $N = 100$ $\sigma = 1.000$





Question 3 Part d

Tuesday, June 4, 2019 12:13 PM

Papers:

1. M. Abeille and A. Lazaric. Thompson Sampling for Linear-Quadratic Control Problems. In AISTATS, 2017.
2. Yasin Abbasi-Yadkori and Csaba Szepesvári. Bayesian optimal control of smoothly parameterized systems. In Proceedings of the Conference on Uncertainty in Artificial Intelligence, 2015

Overview

One exploration technique taken in this paper by Abeille and Lazaric is to use Thompson sampling for LQR learning problems. The problem is divided into a set of episodes. In each episode, the design matrix (matrix of controls and states up to that point) and the regularized least-squares (RLS) estimate of the parameters (θ) are computed. The new RLS estimate $\hat{\theta}_t$ is not based on a Bayesian structure or a Gaussian prior assumption but instead, this method samples a perturbed RLS-estimate, while also checking that the new estimate is admissible.

```
while  $\tilde{\theta}_t \notin S$  do
  Sample  $\eta_t \sim \mathcal{D}^{\text{TS}}$ 
  Compute  $\tilde{\theta}_t = \hat{\theta}_t + \beta_t(\delta')V_t^{-1/2}\eta_t$ 
end while
```

Every coordinate of the matrix η_t is a random sample drawn i.i.d. from $\mathcal{N}(0, 1)$.

The overarching idea of this Thompson sampling process is to sample the parameters $\tilde{\theta}_t$ in each episode such that the action chosen $u_t = K(\tilde{\theta}_t)x_t$ will maximize the expected reward given the sampled parameters, the action, and the current history of states and controls.

Pros

The main benefit of Thompson sampling is that computing the Bayesian optimal policy is computationally expensive. This algorithm is a lazy posterior sampling method that maintains a distribution over the unknown parameter and changes the policy only when the variance of the distribution is reduced sufficiently. This allows the algorithm to trade off performance for computational efficiency while still getting near Bayesian optimal results.

Cons

One of the major difficulties with Thompson sampling that is mentioned by the authors is that Thompson sampling has to solve a trade-off between frequently updating the policy to guarantee enough optimistic samples and minimizing the number of policy switches to limit the regret incurred at each change. Solving this tradeoff leads to a complexity bound of $O(T^{2/3})$ as compared to Optimism in the Face of Uncertainty approaches that have a smaller complexity bound of $O(T^{1/2})$.