EE363 homework 5 solutions

1. *One-step ahead prediction of an autoregressive time series.* We consider the following autoregressive (AR) system

$$p_{t+1} = \alpha p_t + \beta p_{t-1} + \gamma p_{t-2} + w_t, \qquad y_t = p_t + v_t.$$

Here p is the (scalar) time series we are interested in, and y is the scalar measurement available to us. The process noise w is IID zero mean Gaussian, with variance 1. The sensor noise v is IID Gaussian with zero mean and variance 0.01. Our job is to estimate p_{t+1}, based on knowledge of $y_0, \ldots, y_t$. We will use the parameter values

$$\alpha = 2.4, \qquad \beta = -2.17, \qquad \gamma = 0.712.$$

(a) Find the steady state covariance matrix Σ_x of the state

$$x_t = \begin{bmatrix} p_t \\ p_{t-1} \\ p_{t-2} \end{bmatrix}.$$

(b) Run three simulations of the system, starting from statistical steady state. Plot p_t for each of your three simulations.

(c) Find the steady-state Kalman filter for the estimation problem, and simulate it with the three realizations found in part (b). Plot the one-step ahead prediction error for the three realizations.

(d) Find the variance of the prediction error, *i.e.*, $\mathbf{E}(\hat{p}_t - p_t)^2$. Verify that this is consistent with the performance you observed in part (c).

Solution:

(a) We start by writing the state space equations for this system:

$$\begin{bmatrix} p_{t+1} \\ p_t \\ p_{t-1} \end{bmatrix} = \underbrace{\begin{bmatrix} \alpha & \beta & \gamma \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}}_{A} \begin{bmatrix} p_t \\ p_{t-1} \\ p_{t-2} \end{bmatrix} + \underbrace{\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}_{B} w_t,$$

and

$$y_t = \underbrace{\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}}_{C} \begin{bmatrix} p_t \\ p_{t-1} \\ p_{t-2} \end{bmatrix} + v_t.$$

The steady state covariance matrix Σ_x is found by solving the Lyapunov equation

$$\Sigma_x = A\Sigma_x A^T + B\sigma_w B^T,$$

which is done in Matlab by `Sigma_x = dlyap(A,B*B')`, giving the result

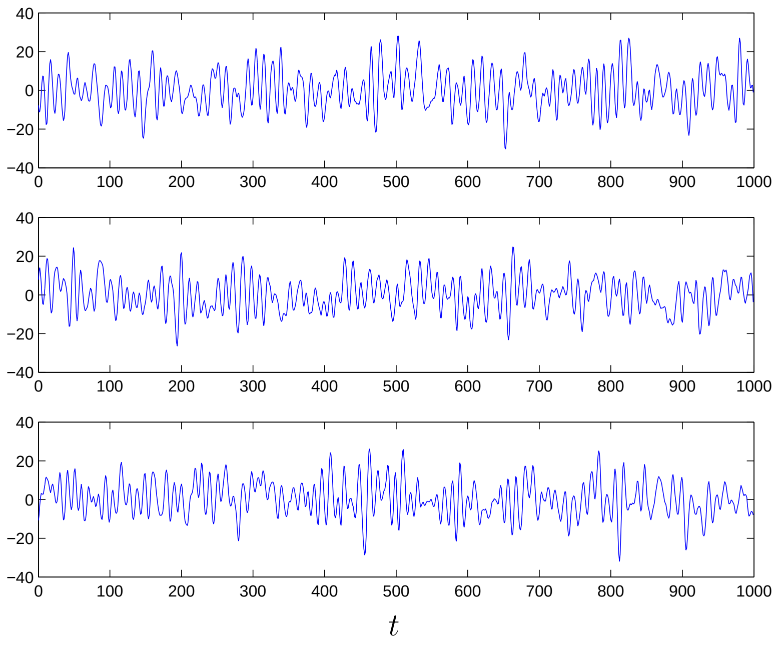
$$\Sigma_x = \begin{bmatrix} 82.2 & 73.7 & 50.8 \\ 73.7 & 82.2 & 73.7 \\ 50.8 & 73.7 & 82.2 \end{bmatrix}$$

(to two significant figures).

(b) A simple Matlab code that runs one simulation follows.

```
% starting from statistical steady state
X = [sqrtm(Sigma_x)*randn(3,1) zeros(3,N)];
sqrtW=sqrtm(W);
for t = 1:N
X(:,t+1) = A*X(:,t) + B*sqrtW*randn(1,1);
end
```

Running this three times gives the plot



(c) We can solve the DARE directly, or by iterating the Riccati recursion. We then calculate the observer gain matrix L. Below is a Matlab code that finds Σ_x both ways and then the observer gain L.

```
% using the dare
Sigma_dare = dare(A',C',B*W*B',V,zeros(3,1),eye(3));
```

```
% recursively
Sigma = X;

for(i = 1:N)
% measurement update
Sigma = Sigma - Sigma*(C')*inv(C*Sigma*(C') + V)*C*Sigma;

% time update
Sigma = A*Sigma*(A') + B*W*B';
end

%compare the two
Sigma - Sigma_dare

L = A*Sigma*C'*inv(C*Sigma*C' + V);
```

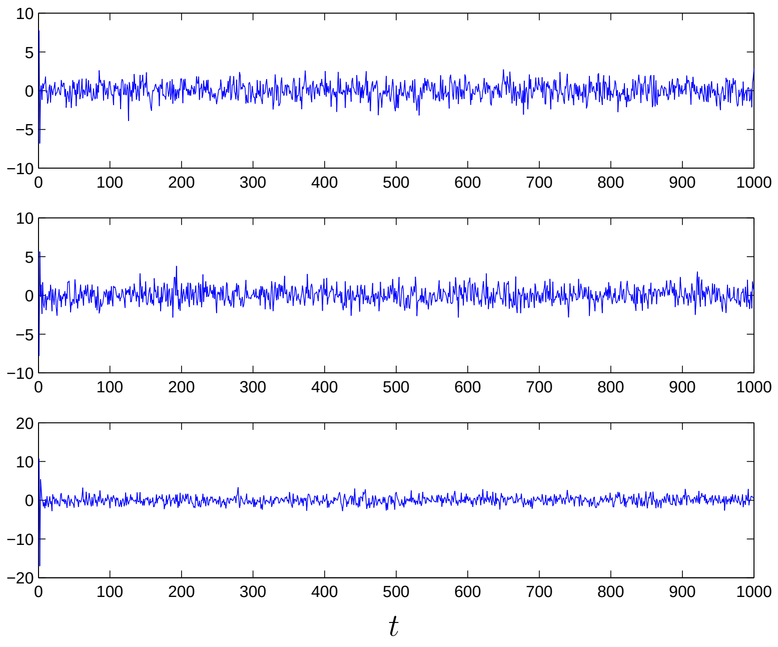
This gives the observer gain

$$L = \begin{bmatrix} 2.3206 \\ 0.9910 \\ 0.0209 \end{bmatrix}.$$

For each simulation we need to run our steady state Kalman filter, here is a Matlab code that does this for one simulation

```
xhat = zeros(3,N+1);

for(t = 1:N)
xhat(:,t+1) = A*xhat(:,t) + L*(Y(t) - C*xhat(:,t));
end
```

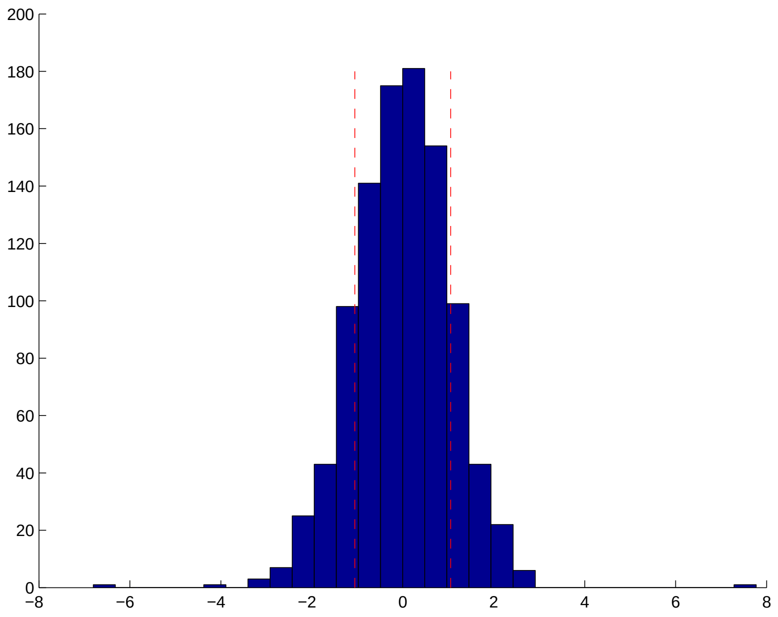
We plot the resulting prediction error $(\hat{x}_t)_1 - p_t$.



(d) We calculate the mean square prediction error using

$$\begin{aligned} \mathbf{E}\ e_t^2 &= \mathbf{E}\ (Cx_t + v_t - C\hat{x}_t)^2 = \mathbf{E}\ (Cx_t + v_t - C\hat{x}_t)^2 \\ &= \mathbf{E}\ (C\tilde{x}_t + v_t)^2 = C\Sigma_x C^T + V = 1.1 \end{aligned}$$

All our random variables are Gaussian, so the prediction error (which is a linear combination of our random variables) is Gaussian as well. Looking at a histogram of the prediction error we expect about 67% of the prediction errors to fall within one standard deviation (*i.e.*, $\sqrt{\mathbf{E}\ e_t^2}$) of our prediction error.



To ease comparision we have marked in $\pm\sqrt{\mathbf{E}\ e_t^2}$ on the histogram.

2. *Performance of Kalman filter when the system dynamics change.* We consider the Gauss-Markov system

$$x_{t+1} = Ax_t + w_t, \qquad y_t = Cx_t + v_t, \tag{1}$$

with v and w are zero mean, with covariance matrices $V > 0$ and $W \geq 0$, respectively. We'll call this system the *nominal system.*

We'll consider another Gauss-Markov sysem, which we call the *perturbed system*:

$$x_{t+1} = (A + \delta A)x_t + w_t, \qquad y_t = Cx_t + v_t, \tag{2}$$

where $\delta A \in \mathbf{R}^{n\times n}$. Here (for simplicity) C, V, and W are the same as for the nominal system; the only difference between the perturbed system and the nominal system is that the dynamics matrix is $A + \delta A$ instead of A.

In this problem we examine what happens when you design a Kalman filter for the nominal system (1), and use it for the perturbed system (2).

Let L denote the steady-state Kalman filter gain for the nominal system (1), *i.e.*, the steady-state Kalman filter for the nominal system is

$$\hat{x}_{t+1} = A\hat{x}_t + L(y_t - \hat{y}_t), \qquad \hat{y}_t = C\hat{x}_t. \tag{3}$$

(We'll assume that (C, A) is observable and (A, W) is controllable, so the steady-state Kalman filter gain L exists, is unique, and $A - LC$ is stable.)

Now suppose we use the filter (3) on the perturbed system (2).

We will consider the specific case

$$A = \begin{bmatrix} 1.8 & -1.4 & 0.5 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \qquad \delta A = \begin{bmatrix} 0.1 & -0.2 & 0.1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$C = [1\ 0\ 0], \qquad W = I, \qquad V = 0.01.$$

   (a) Find the steady-state value of $\mathbf{E}\,\|x_t\|^2$, for the nominal system, and also for the perturbed system.

   (b) Find the steady-state value of $\mathbf{E}\,\|\hat{x}_t - x_t\|^2$, where x is the state of the perturbed system, and $\hat{x}$ is the state of the Kalman filter (designed for the nominal system). (In other words, find the steady-state mean square value of the one step ahead prediction error, using the Kalman filter designed for the nominal system, but with the perturbed system.)

   Compare this to $\mathbf{Tr}\,\hat{\Sigma}$, where $\hat{\Sigma}$ is the steady-state one step ahead prediction error covariance, when the Kalman filter is run with the nominal system. ($\mathbf{Tr}\,\hat{\Sigma}$ gives the steady-state value of $\mathbf{E}\,\|\hat{x}_t - x_t\|^2$, when x evolves according to the nominal system.)

Solution:

(a) We first check that the nominal and perturbed systems are stable (otherwise talking about the steady-state of $\|x_t\|^2$ is meaningless). Both are verified to be stable (by computing the eigenvalues, for example.)

We find the steady state covariance matrix for the state of the nominal system by solving the Lyapunov equation $\Sigma = A\Sigma A^T + W$. The mean square value $\mathbf{E}\,\|x_t\|^2$ is then given by $\mathbf{Tr}\,\Sigma$. We repeat this for the perturbded system.

```
Sigma_xn = dlyap(A,W);
msXn = trace(Sigma_xn)
Sigma_xp = dlyap(A+dA,W);
msXp = trace(Sigma_xp)
```

which gives the values

$$\mathbf{E}\,\|x_t\|^2 = 74.1, \qquad \mathbf{E}\,\|x_t\|^2 = 97.88,$$

for the nominal and perturbed systems, respectively.

(b) We start by designing a steady state Kalman filter for the nominal system, for example by solving the Riccati equation

$$\hat{\Sigma} = A\hat{\Sigma}A^T + W - A\hat{\Sigma}C^T(C\hat{\Sigma}C^T + V)^{-1}C\hat{\Sigma}A^T,$$

and then we find the Kalman filter gain L by

$$L = A\hat{\Sigma}C^T(C\hat{\Sigma}C^T + V)^{-1}.$$

The dynamics of the Kalman filter, running with the perturbed system, are given by

$$\begin{aligned} x_{t+1} &= (A + \delta A)x_t + w_t \\ \hat{x}_{t+1} &= A\hat{x}_t + L(Cx_t + v_t - C\hat{x}_t) \\ &= LCx_t + (A - LC)\hat{x}_t + Lv_t. \end{aligned}$$

We can write this as one big linear $2n$-state system,

$$\begin{bmatrix} x_{t+1} \\ \hat{x}_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} A + \delta A & 0 \\ LC & A - LC \end{bmatrix}}_{\tilde{A}} \underbrace{\begin{bmatrix} x_t \\ \hat{x}_t \end{bmatrix}}_{\tilde{x}_t} + \underbrace{\begin{bmatrix} w_t \\ Lv_t \end{bmatrix}}_{\tilde{w}_t}.$$

We find $\tilde{\Sigma}$, the steady-state covariance matrix of $\tilde{x}_t$ by solving the Lyapunov equation

$$\tilde{\Sigma} = \tilde{A}\tilde{\Sigma}\tilde{A}^T + \tilde{W},$$

where the covariance matrix of $\tilde{w}_t$ is

$$\tilde{W} = \left[\begin{array}{cc} W & 0 \\ 0 & LVL^T \end{array} \right].$$

To find $\mathbf{E}\,\|\hat{x}_t - x_t\|^2$, we first find the covariance of $z = \hat{x} - x = [-I\ I]\tilde{x}$:

$$\begin{aligned} \mathbf{E}\,zz^T &= \mathbf{E}([-I\ I]\tilde{x})([-I\ I]\tilde{x})^T \\ &= [-I\ I]\,\mathbf{E}\,\tilde{x}\tilde{x}^T[-I\ I]^T \\ &= [-I\ I]\tilde{\Sigma}[-I\ I]. \end{aligned}$$

Taking the trace of this yields $\mathbf{E}\,\|\hat{x}_t - x_t\|^2$.
To get numerical values we use the matlab code below:

```
%kf for the system (A,C)
Sigmahat = dare(A',C',W,V,zeros(n,1),eye(n));
L = A*Sigmahat*(C')*inv(C*Sigmahat*(C') - V);

%the steady state covariance matrix of xtilde,
Xtildess = dlyap([(A+dA) zeros(n,n);(L*C) (A - L*C)],...
[W zeros(n,n);zeros(n,n) (L*V*(L'))]);

% we calculate the ss, ms errors
msprederr = trace([-eye(n) eye(n)]*Xtildess*[-eye(n);eye(n)]);
msprederrnomin = trace(Sigmahat)
```

which gives the numerical values

$$\mathbf{E}\,\|\hat{x}_t - x_t\|^2 = 7.2 \qquad \mathbf{Tr}\,\hat{\Sigma} = 6.37.$$

3. *Open-loop control.* We consider a linear dynamical system with n states and m inputs,

$$x_{t+1} = Ax_t + Bu_t + w_t, \quad t = 0, 1, \ldots,$$

where w_t are IID $\mathcal{N}(0, \Sigma_w)$, and $x_0 \sim \mathcal{N}(0, \Sigma_0)$ is independent of all w_t. The objective is

$$J = \mathbf{E}\left(\sum_{t=0}^{N-1} \left(x_t^T Q x_t + u_t^T R u_t \right) + x_N^T Q_f x_N \right)$$

where $Q \geq 0$, $Q_f \geq 0$, and $R > 0$.

In the standard stochastic control setup, we choose u_t as a function of the current state x_t, so we have $u_t = \phi_t(x_t)$, $t = 0, \ldots, N-1$. In *open-loop control*, we choose u_t as a function of the initial state x_0 only, so we have $u_t = \psi_t(x_0)$, $t = 0, \ldots, N-1$. Thus, in open-loop control, we must commit to an input sequence at time $t = 0$, based only on knowledge of the initial state x_0; in particular, there is no opportunity for recourse or changes in the input due to new observations. The open loop control problem is to choose the control functions $\psi_0, \dots, \psi_{N-1}$ that minimize the objective J.

In this exercise, you will derive explicit expressions for the optimal control functions $\psi_0^\star, \dots, \psi_{N-1}^\star$, for the open-loop control problem. The problem data are A, B, Σ_w, Q, Q_f, and R, and N.

Show that the optimal control functions are $\psi_0^\star(x_0) = K_0 x_0$, and

$$\psi_t^\star(x_0) = K_t(A + BK_{t-1}) \cdots (A + BK_0)x_0, \quad t = 1, \dots, N-1,$$

where

$$K_t = -(R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A, \quad t = 0, \dots, N-1,$$

and

$$P_t = Q + A^T P_{t+1} A - A^T P_{t+1} B (R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A, \quad t = N-1, \dots, 0,$$

with $P_N = Q_f$. In other words, we can solve the open-loop control problem by solving the deterministic LQR problem obtained by taking $w_0 = w_1 = \cdots = w_{N-1} = 0$.

Solution.

Let $V_0(z)$ denote the optimal value of the objective, starting from $t = 0$ at $x_0 = z$. Since we must commit to an input sequence given $x_0 = z$, we can express $V_0(z)$ as

$$V_0(z) = \min_{u_0, \dots, u_{N-1}} \mathbf{E} \left(\sum_{t=0}^{N-1} \left(x_t^T Q x_t + u_t^T R u_t \right) + x_N^T Q_f x_N \right),$$

subject to the system dynamics, and $x_0 = z$. Let's define $X = (x_1, \dots, x_N) \in \mathbf{R}^{nN}$, $U = (u_0, u_1, \dots, u_{N-1}) \in \mathbf{R}^{mN}$, and $W = (w_0, w_1, \dots, w_{N-1}) \in \mathbf{R}^{nN}$. We can write

$$X = Fx_0 + GU + HW,$$

where

$$F = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \quad G = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}, \quad H = \begin{bmatrix} I & 0 & \cdots & 0 \\ A & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1} & A^{N-2} & \cdots & I \end{bmatrix}.$$

Thus we have

$$\begin{aligned} V_0(z) &= \min_U \left\{ \mathbf{E} \left(z^T Q z + (Fz + GU + HW)^T \tilde{Q} (Fz + GU + HW) + U^T \tilde{R} U \right) \right\} \\ &= \min_U \left\{ z^T (Q + F^T \tilde{Q} F) z + U^T (\tilde{R} + G^T \tilde{Q} G) U + 2 z^T F^T \tilde{Q} G U + \mathbf{E}(W^T H^T \tilde{Q} H W) \right\} \\ &= \min_U \left\{ z^T (Q + F^T \tilde{Q} F) z + U^T (\tilde{R} + G^T \tilde{Q} G) U + 2 z^T F^T \tilde{Q} G U \right\} + \mathbf{E}(W^T H^T \tilde{Q} H W) \\ &= \min_U \left\{ z^T Q z + (Fz + GU)^T \tilde{Q} (Fz + GU) + U^T \tilde{R} U \right\} + \mathbf{E}(W^T H^T \tilde{Q} H W), \end{aligned}$$

where,

$$\tilde{Q} = \begin{bmatrix} Q & 0 & \cdots & 0 \\ 0 & Q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q_f \end{bmatrix}, \quad \tilde{R} = \begin{bmatrix} R & 0 & \cdots & 0 \\ 0 & R & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R \end{bmatrix}.$$

We notice that minimizing

$$z^T Q z + (Fz + GU)^T \tilde{Q}(Fz + GU) + U^T \tilde{R} U$$

is equivalent to solving the deterministic LQR problem

$$\text{minimize} \quad \sum_{t=0}^{N-1} \left(x_t^T Q x_t + u_t^T R u_t \right) + x_N^T Q_f x_N,$$

subject to

$$x_{t+1} = Ax_t + Bu_t, \quad t = 0, \ldots, N-1,$$

with $x_0 = z$. Thus we get $\psi_0^\star(z) = u_0^\star = K_0 z$ and

$$\psi_t^\star(z) = u_t^\star = K_t x_t = K_t(A + BK_{t-1}) \cdots (A + BK_0)z, \quad t = 0, \ldots, N-1,$$

where

$$K_t = -(R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A, \quad t = 0, \ldots, N-1,$$

and

$$P_t = Q + A^T P_{t+1} A - A^T P_{t+1} B (R + B^T P_{t+1} B)^{-1} B^T P_{t+1} A, \quad t = N-1, \ldots, 0,$$

with $P_N = Q_f$.

4. *Simulation of a Gauss-Markov system from statistical steady-state.* We consider a Gauss-Markov system,

$$x_{t+1} = Ax_t + w_t,$$

where $A \in \mathbf{R}^{n \times n}$ is stable (*i.e.*, its eigenvalues all have magnitude less than one), w_t are IID with $w_t \sim \mathcal{N}(0, W)$, and $x_0 \sim \mathcal{N}(0, \Sigma_0)$, independent of all w_t. Let Σ_x denote the asymptotic value of the state covariance. If $x_0 \sim \mathcal{N}(0, \Sigma_x)$ (*i.e.*, $\Sigma_0 = \Sigma_x$), then we have $\mathbf{E}\, x_t = 0$ and $\mathbf{E}\, x_t x_t^T = \Sigma_t$ for all t. We refer to this as statistical equilibrium, or statistical steady-state.

Generate a random $A \in \mathbf{R}^{10 \times 10}$ in Matlab using `A = randn(n)`, then scaling it so its spectral radius (maximum magnitude of all eigenvalues) is 0.99. Choose W to be a random positive semidefinite matrix, for example using `W = randn(n); W = W'*W;`. Create two sets of 50 trajectories for 100 time steps; in one set, initialize with $x_0 = 0$, in the other, with $x_0 \sim \mathcal{N}(0, \Sigma_x)$.

Create two plots, overlaying the trajectories of $(x_t)_1$ within each set. Comment briefly on what you see.

Solution.

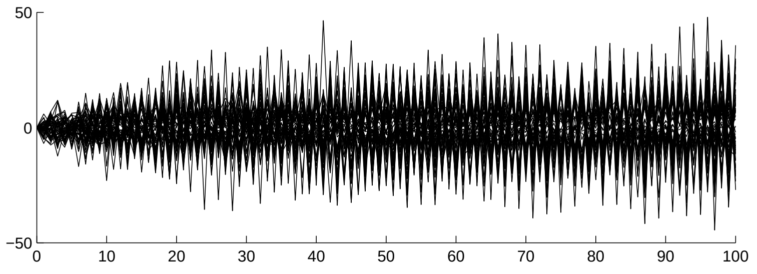
To calculate Σ_x, we solve (using, for example, `dlyap`) the Lyapunov equation

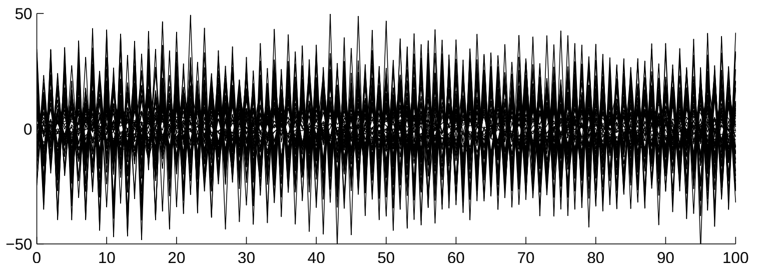
$$\Sigma_x = A\Sigma_x A^T + W.$$

We use this to generate and plot trajectories in Matlab.

Matlab code and the resulting graphs appear below.

```
randn('state', 2920); n = 10; N = 100;
A = randn(n); A = (0.99/max(abs(eig(A))))*A;
W = randn(n); W = W'*W; Whalf = sqrtm(W); Ex = dlyap(A, W);
subplot(211); cla reset; hold on; subplot(212); cla reset; hold on;
for j = 1:50
    x_zero = zeros(n, N+1);
    for i = 1:N
        x_zero(:,i+1) = A*x_zero(:,i) + Whalf*randn(n,1);
    end

    x_ss = zeros(n, N); x_ss(:,1) = sqrtm(Ex)*randn(n,1);
    for i = 1:N
        x_ss(:,i+1) = A*x_ss(:,i) + Whalf*randn(n,1);
    end

    subplot(211); plot(0:N, x_zero(1,:)); subplot(212); plot(0:N, x_ss(1,:))
end
subplot(211); axis([0 N -50 50]); subplot(212); axis([0 N -50 50])
print -deps2 stat_steady_state.eps
```





5. *Implementing a Kalman filter.* In this problem you will implement a simple Kalman filter for a linear Gauss-Markov system

$$x_{t+1} = Ax_t + w_t, \quad y_t = Cx_t + v_t$$

with $x_0 \sim \mathcal{N}(0, I)$, $w_t \sim \mathcal{N}(0, W)$ and $v_t \sim \mathcal{N}(0, V)$.

Generate a system in Matlab by randomly generating a matrix $A \in \mathbf{R}^{10\times 10}$ and scaling it so its spectral radius is 0.95, a matrix $C \in \mathbf{R}^{3\times 10}$, and positive definite matrices W and V. Find the Kalman filter for this system.

Plot $\sqrt{\mathbf{E}\,\|x_t\|^2}$ and $\sqrt{\mathbf{E}\,\|x_t - \hat{x}_t\|^2}$, for $t = 1, \ldots, 50$. Then, simulate the system for 50 time steps, plotting $\|x_t\|_2$ and $\|x_t - \hat{x}_t\|_2$.

Solution.

This requires a straight-forward implementation of a Kalman filter. Matlab code and the resulting graphs appear below.

```
randn('state', 2918); m = 3; n = 10; N = 50;
A = randn(n); A = (0.95/max(abs(eig(A))))*A;
W = 0.1*randn(n); W = W'*W; Whalf = sqrtm(W);
V = 0.1*randn(m); V = V'*V; Vhalf = sqrtm(V);
C = randn(m, n);

E = eye(n); Epred = E;
normx = [sqrt(trace(E))]; normdx = [sqrt(trace(E))];
x = sqrtm(E)*randn(n,1); xhat = zeros(n,1);
normxt = [norm(x)]; normdxt = [norm(x - xhat)];
Ex = eye(n);
for t = 1:N
    % Variance update.
    Ex = A*Ex*A' + W;

    % Propagate the system forward.
    x = A*x + Whalf*randn(n,1);
    y = C*x + Vhalf*randn(m,1);

    % Measurement update.
    xhat = A*xhat + Epred*C'*inv(C*Epred*C' + V)*(y - C*A*xhat);
    E = Epred - Epred*C'*inv(C*Epred*C' + V)*C*Epred;

    % Time update.
    Epred = A*E*A' + W;

    % For plots.
    normx = [normx sqrt(trace(Ex))]; normdx = [normdx sqrt(trace(E))];
```

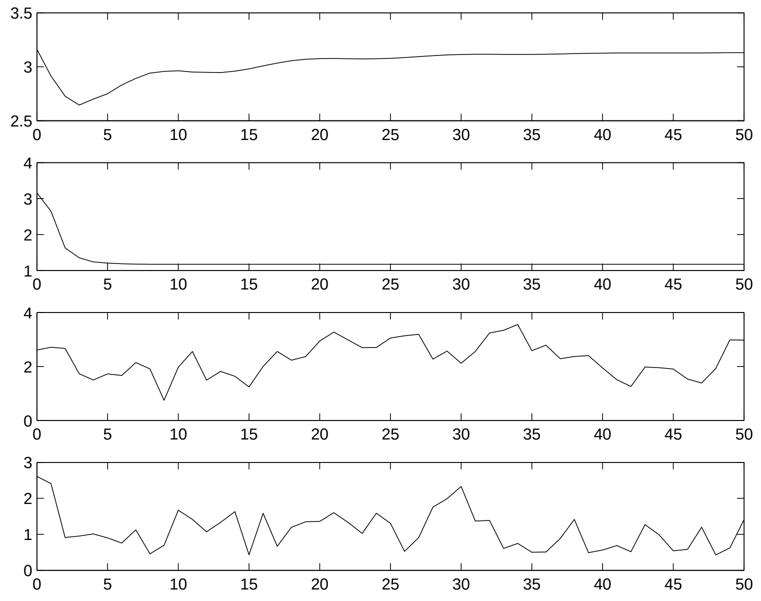


Figure 1: From top to bottom: $\mathbf{E}\,\|x_t\|_2$, $\mathbf{E}\,\|x_t - \hat{x}_t\|_2$; then $\|x_t\|_2$ and $\|x_t - \hat{x}_t\|_2$ for one particular realization.

```
    normxt = [normxt norm(x)]; normdxt = [normdxt norm(xhat - x)];
end
subplot(411); plot(0:N, normx); subplot(412); plot(0:N, normdx)
subplot(413); plot(0:N, normxt); subplot(414); plot(0:N, normdxt)
print -deps2 kalman.eps
```

6. *Simultaneous sensor selection and state estimation.* We consider a standard state estimation setup:

$$x_{t+1} = Ax_t + w_t, \qquad y_t = C_t x_t + v_t,$$

where $A \in \mathbf{R}^{n\times n}$ is constant, but C_t can vary with time. The process and measurement noise are independent of each other and the initial state $x(0)$, with

$$x(0) \sim \mathcal{N}(0, \Sigma_0), \qquad w_t \sim \mathcal{N}(0, W), \qquad v_t \sim \mathcal{N}(0, V).$$

The standard formulas for the Kalman filter allow you to compute the next state prediction $\hat{x}_{t|t-1}$, current state prediction $\hat{x}_{t|t}$, and the associated prediction error covariances $\Sigma_{t|t-1}$ and $\Sigma_{t|t}$.

Now we are going to introduce a twist. The measurement matrix C_t is one of K possible values, *i.e.*, $C_t \in \{C_1, \dots, C_K\}$. In other words, at each time t, we have $C_t = C_{i_t}$. The sequence i_t specifies which of the K possible measurements is taken at time t. For example, the sequence $2, 2, \dots$ means that $C_t = C_2$ for all t; the sequence

$$1, 2, \dots, K,\ 1, 2 \dots, K,\ \dots$$

is called *round-robin*: we cycle through the possible measurements, in order, over and over again.

Here's the interesting part: *you* get to choose the measurement sequence $i_0, i_1, \ldots,$. You will use the following greedy algorithm. You will choose the sequence in order; having chosen $i_0, \ldots, i_{t-1}$, you will choose i_t so as to minimize the mean-square prediction error associated with $\hat{x}_{t|t}$. This is the same as choosing i_t so that $\mathbf{Tr}\,\Sigma_{t|t}$ is minimized. Roughly speaking, at each step, you choose the sensor that results in the smallest mean-square state prediction error, given the sensor choices you've made so far, plus the one you're choosing.

Let's be very clear about this method for choosing i_t. The choice of $i_0, \ldots, i_{t-1}$ determines $\Sigma_{t|t-1}$; then, $\Sigma_{t|t}$ depends on i_t, *i.e.*, which of $C_1, \ldots, C_K$ is chosen as C_t. Among these K choices, you pick the one that minimizes $\mathbf{Tr}\,\Sigma_{t|t}$.

This method does not require knowledge of the actual measurements $y_0, y_1, \ldots,$ so we can determine the sequence of measurements we are going to make *before any data have been received.* In particular, the sequence can be determined ahead of time (at least up to some large value of t), and stored in a file.

Now we get to the question. You will work with the specific system with

$$A = \begin{bmatrix} -0.6 & 0.8 & 0.5 \\ -0.1 & 1.5 & -1.1 \\ 1.1 & 0.4 & -0.2 \end{bmatrix}, \qquad W = I, \qquad V = 0.1^2, \qquad \Sigma_0 = I,$$

and $K = 3$ with

$$C_1 = \begin{bmatrix} 0.74 & -0.21 & -0.64 \end{bmatrix}, \qquad C_2 = \begin{bmatrix} 0.37 & 0.86 & 0.37 \end{bmatrix}, \qquad C_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}.$$

(a) *Using one sensor.* Plot the mean-square current state prediction error $\mathbf{Tr}\,\Sigma(t|t)$ versus t, for the three special cases when $C_t = C_1$ for all t, $C_t = C_2$ for all t, and $C_t = C_3$ for all t.

(b) *Round-robbin.* Plot the mean-square current state prediction error $\mathbf{Tr}\,\Sigma(t|t)$ versus t, using sensor sequence $1, 2, 3, 1, 2, 3, \ldots$

(c) *Greedy sensor selection.* Find the specific sensor sequence generated by the algorithm described above. Show us the sequence, by plotting i_t versus t. Plot the resulting mean-square estimation error, $\mathbf{Tr}\,\Sigma_{t|t}$, versus t. Briefly compare the results to what you found in parts (a) and (b).

In all three parts, you can show the plots over the interval $t = 0, \ldots, 50$.

To save you some time, we have created the file `sens_data.m`, which contains the problem data. The file also contains two lines, currently commented out, that implement a generic Kalman filter measurement and time update. You're welcome to use these, or to use or write your own.

Solution.

(a) Let $\Sigma_i(t|t)$ be the estimation error covariance when $C_t = C_i$, for all t. To plot the evolution of the MSE with time, we just have to iteratively apply the time and measurement update formulas from the lecture notes.

In order to find the asymptotic value of $\mathbf{Tr}\,\Sigma_i(t|t)$ (which we will denote $\mathbf{Tr}\,\Sigma_{i,\mathrm{ss}}(t|t)$), we first have to solve the DARE

$$\hat{\Sigma}_i = A\hat{\Sigma}_i A^T + W - A\hat{\Sigma}_i C_i^T (C_i\hat{\Sigma}_i C^T + V)^{-1} C_i\hat{\Sigma}_i A^T,$$

and then apply the measurement update formula

$$\Sigma_{i,\mathrm{ss}}(t|t) = \hat{\Sigma}_i - \hat{\Sigma}_i C_i^T (C_i\hat{\Sigma}_i C^T + V)^{-1} C_i\hat{\Sigma}_i.$$

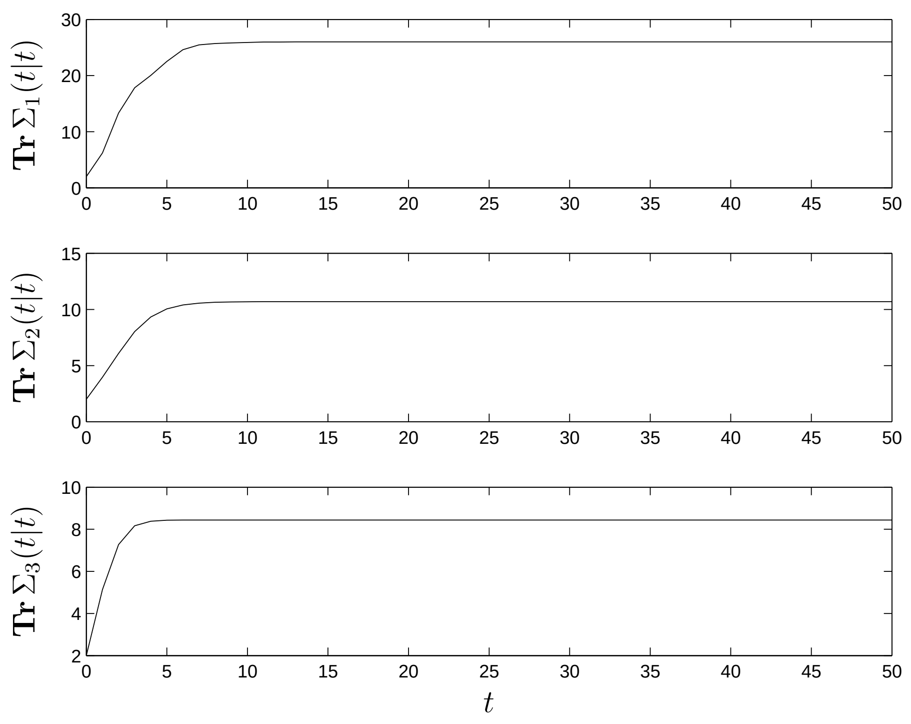
The following matlab code was used for this part of the problem:

```
sens_data
N = 50;

% Fixed Sensor Policy
Sigmahat1 = Sigma0; Sigmahat2 = Sigma0; Sigmahat3 = Sigma0;
mses1=[]; mses2 = []; mses3 = [];
for n = 1:N+1
    % First sensor
    C = C1;
    % Measurement Update
    Sigma1 = Sigmahat1-Sigmahat1*C'*inv(C*Sigmahat1*C'+V)*...
        C*Sigmahat1;
    % Time Update
    Sigmahat1 = A*Sigma1*A'+W;
    mses1 = [mses1 trace(Sigma1)];

    % Second sensor
    C = C2;
    % Measurement Update
    Sigma2 = Sigmahat2-Sigmahat2*C'*inv(C*Sigmahat2*C'+V)*...
        C*Sigmahat2;
    % Time Update
    Sigmahat2 = A*Sigma2*A'+W;
    mses2 = [mses2 trace(Sigma2)];

    % Third sensor
    C = C3;
    % Measurement Update
    Sigma3 = Sigmahat3-Sigmahat3*C'*inv(C*Sigmahat3*C'+V)*...
        C*Sigmahat3;
    % Time Update
    Sigmahat3 = A*Sigma3*A'+W;
    mses3 = [mses3 trace(Sigma3)];
end

figure
subplot(3,1,1)
plot(0:N,mses1)
ylabel('mse1')
subplot(3,1,2)
plot(0:N,mses2)
ylabel('mse2')
subplot(3,1,3)
plot(0:N,mses3)
ylabel('mse3')
xlabel('time')

print -deps msefixed.eps

% Find steady-state values
% First sensor
C = C1;
Shat = dare(A',C',W,V);
mse1 = trace(Shat-Shat*C'*inv(C*Shat*C'+V)*C*Shat)

% Second sensor
C = C2;
Shat = dare(A',C',W,V);
mse2 = trace(Shat-Shat*C'*inv(C*Shat*C'+V)*C*Shat)

% Third sensor
C = C3;
Shat = dare(A',C',W,V);
mse3 = trace(Shat-Shat*C'*inv(C*Shat*C'+V)*C*Shat)
```

The steady-state values of the MSE for each i are

$$\mathbf{Tr}\,\Sigma_{1,\mathrm{ss}}(t|t) = 26.01, \qquad \mathbf{Tr}\,\Sigma_{2,\mathrm{ss}}(t|t) = 10.69, \qquad \mathbf{Tr}\,\Sigma_{3,\mathrm{ss}}(t|t) = 8.44.$$

The following plots show the evolution of $\mathbf{Tr}\,\Sigma_i(t|t)$ with time, for each i.

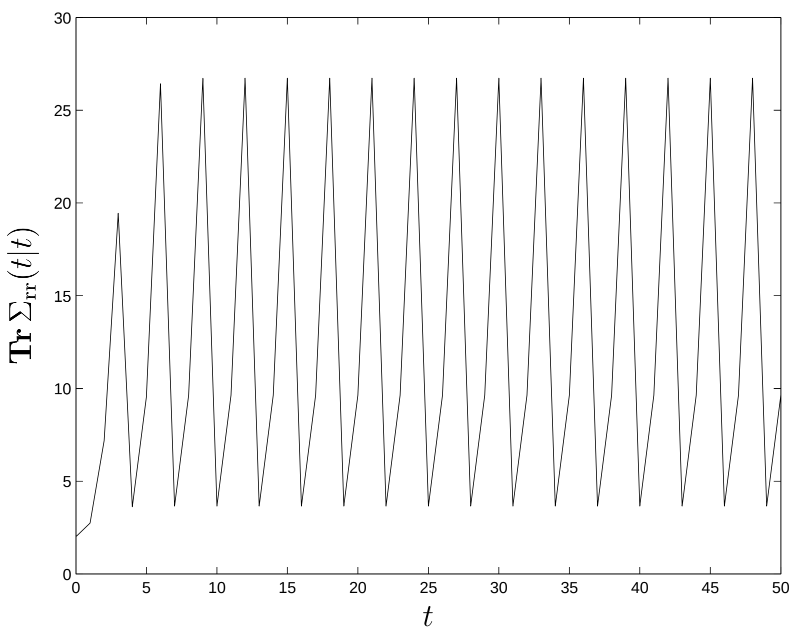


It is evident that the best fixed sensor choice is $C_t = C_3$, for all t.

(b) Let $\Sigma_{\rm rr}(t|t)$ be the estimation error covariance when using a round-robbin sensor sequence. The following matlab code calculates and plots $\mathbf{Tr}\,\Sigma_{\rm rr}(t|t)$, for $t = 0, \dots, 50$:

```
% Round robbin
Sigmahat = Sigma0;
mse_rr=[];
time = 1;
while(1)
    % Sensor 1
    C = C1;

    % Measurement Update
    Sigma = Sigmahat-Sigmahat*C'*inv(C*Sigmahat*C'+V)*...
        C*Sigmahat;

    % Time Update
    Sigmahat = A*Sigma*A'+W;

    mse_rr = [mse_rr trace(Sigma)];
    time = time+1;
    if(time>N+1), break; end
    % Sensor 2
    C = C2;

    % Measurement Update
    Sigma = Sigmahat-Sigmahat*C'*inv(C*Sigmahat*C'+V)*...
        C*Sigmahat;

    % Time Update
    Sigmahat = A*Sigma*A'+W;

    mse_rr = [mse_rr trace(Sigma)];
    time = time+1;
    if(time>N+1), break; end

    % Sensor 3
    C = C3;

    % Measurement Update
    Sigma = Sigmahat-Sigmahat*C'*inv(C*Sigmahat*C'+V)*...
        C*Sigmahat;

    % Time Update
    Sigmahat = A*Sigma*A'+W;

    mse_rr = [mse_rr trace(Sigma)];
    time = time+1;
    if(time>N+1), break; end
end

figure
plot(0:N,mse_rr);
ylabel('mserr')
xlabel('time')
print -deps mserr.eps
```

The following plot shows the evolution of $\mathbf{Tr}\,\Sigma_{\mathrm{rr}}(t|t)$ with time.

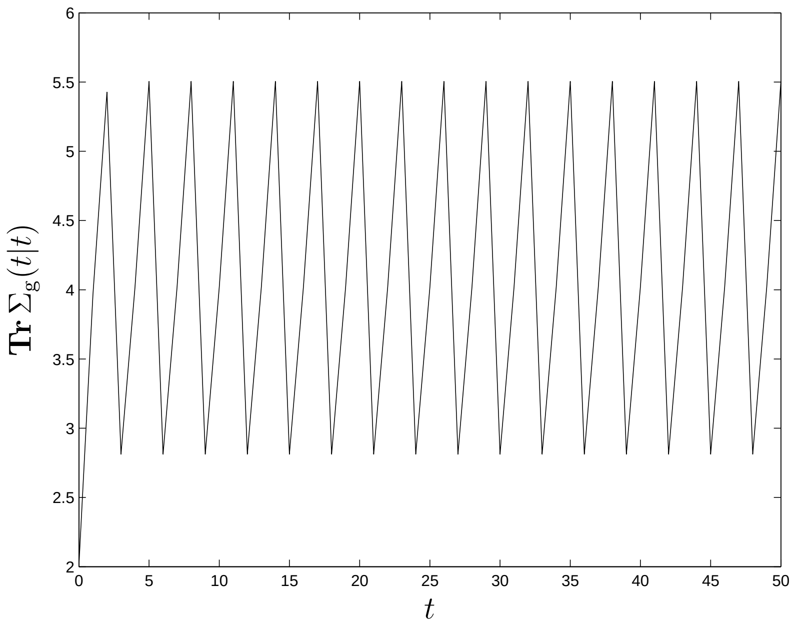


The round-robbin sensor sequence is performing much worse than selecting the best fixed sensor.

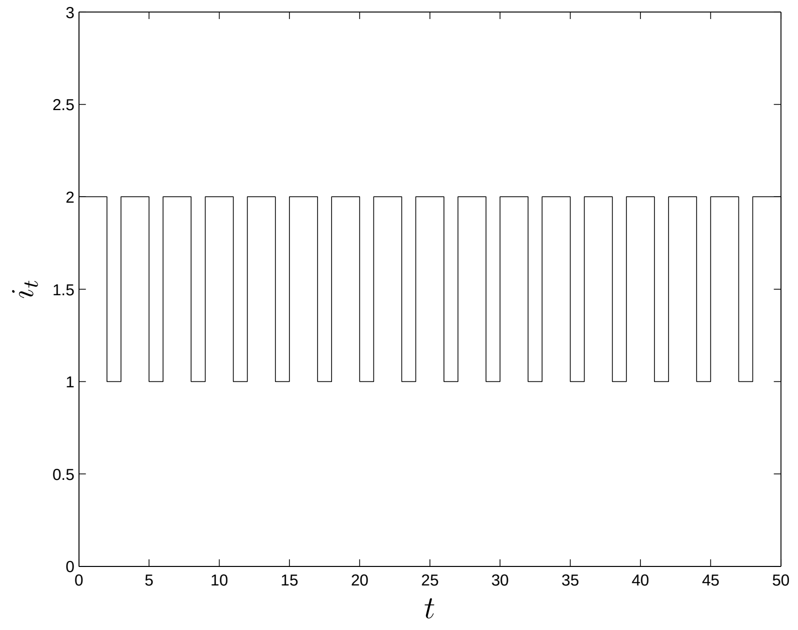
(c) Let $\Sigma_{\mathrm{g}}(t|t)$ be the estimation error covariance when using the proposed greedy sensor selection heuristic. The following matlab code calculates and plots $\mathbf{Tr}\,\Sigma_{\mathrm{g}}(t|t)$, for $t = 0, \ldots, 50$:

```
% Greedy algorithm
Sigmahat = Sigma0;
mse_g=[];
policy = [];
for n = 1:N+1
    % Measurement Updates
    % First sensor
    C = C1;
    Sigma1 = Sigmahat-Sigmahat*C'*inv(C*Sigmahat*C'+V)*...
        C*Sigmahat;

    % Second sensor
    C = C2;
    Sigma2 = Sigmahat-Sigmahat*C'*inv(C*Sigmahat*C'+V)*...
        C*Sigmahat;

    % Third sensor
    C = C3;
    Sigma3 = Sigmahat-Sigmahat*C'*inv(C*Sigmahat*C'+V)*...
        C*Sigmahat;
    % Greedy sensor selection
    mses = [trace(Sigma1) trace(Sigma2) trace(Sigma3)];
    [min_mse,ind] = min(mses);
    ind = ind(1);
    policy = [policy ind];
    mse_g = [mse_g min_mse];
    switch ind
        case 1
            Sigma = Sigma1;
        case 2
            Sigma = Sigma2;
        case 3
            Sigma = Sigma3;
    end

    % Time update
    Sigmahat = A*Sigma*A'+W;
end

figure
plot(0:N,mse_g);
ylabel('mseg')
xlabel('time')
print -deps mseg.eps

figure
stairs(0:N,policy)
ylabel('policy')
xlabel('time')
axis([0 N 0 3])
print -deps polg.eps
```

The following plot shows the evolution of $\mathbf{Tr}\,\Sigma_{\rm g}(t|t)$ with time.



The following plot shows the sensor sequence i_t used by the greedy heuristic.



For this particular system, the greedy heuristic is better than using a single fixed sensor with respect to the MSE, since

$$\mathbf{Tr}\,\Sigma_{\rm g}(t|t) < \mathbf{Tr}\,\Sigma_3(t|t),$$

for all t. It is interesting to note that the sensor sequence used by the heuristic does not contain C_3, which is the optimal fixed sensor.