Ville Väänänen
63527M
ville.vaananen@aalto.fi

Exercise Report

S-114.4202 Special Course in Computational Engineering II

September 23, 2011

Round 1

Exercise 1.1

A)

The problem can be written in matrix form as follows:

$$\begin{aligned}\mathbf{y} &= \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}^T \\ \mathbf{a} &= \begin{bmatrix} a_1 & a_2 \end{bmatrix}^T \\ \mathbf{X} &= \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \\ \mathbf{y} &= \mathbf{X}\mathbf{a}\end{aligned}$$

B)

$$\mathrm{E}(a_1, a_2) = (\mathbf{y} - \mathbf{X}\mathbf{a})^T(\mathbf{y} - \mathbf{X}\mathbf{a})$$

C)

To compute the LS-estimate $\hat{\mathbf{a}}$, we need to find the global minimum of E, which can be found by setting its gradient to zero (it's a quadratic form):

$$\begin{aligned}\nabla \mathrm{E} &= -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{a}) \\ &= 2\mathbf{X}^T\mathbf{X}\mathbf{a} - 2\mathbf{X}^T\mathbf{y} \\ \Rightarrow \hat{\mathbf{a}} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}\end{aligned}$$

Exercise 1.2

A)

The second order differential equation can be written as a first order vector valued differential equation as follows:

$$\begin{aligned}\frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t} &= \begin{bmatrix} 0 & -c^2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x'(t) \\ x(t) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} w(t) \\ \Leftrightarrow \mathbf{x}'(t) &= \mathbf{F}\mathbf{x}(t) + \mathbf{L}w(t).\end{aligned}$$

Let us proceed to solve this equation:

$$\begin{aligned}\mathbf{x}'(t) - \mathbf{F}\mathbf{x}(t) &= \mathbf{L}w(t) \\ e^{-\mathbf{F}t}\mathbf{x}'(t) - e^{-\mathbf{F}t}\mathbf{F}\mathbf{x}(t) &= e^{-\mathbf{F}t}\mathbf{L}w(t) \\ \frac{\mathrm{d}}{\mathrm{d}t}\left(e^{-\mathbf{F}t}\mathbf{x}(t)\right) &= e^{-\mathbf{F}t}\mathbf{L}w(t)\end{aligned}$$

so that

$$\mathbf{x}(t) = e^{\mathbf{F}(t-t_0)}\mathbf{x}(t_0) + \int_{t_0}^{t} e^{\mathbf{F}(t-s)}\mathbf{L}w(s)\ \mathrm{d}s \tag{1}$$

where

$$\mathbf{x}(t_0) = \begin{bmatrix} v_0 \\ x_0 \end{bmatrix}$$

B)

After discretizing the time into $n+1$ instants as $\{t_0 = 0, t_1 = \Delta t, \ldots, t_n = n\Delta t\}$ and assuming $w(s) = w(t_{k-1})\ \forall s \in [t_{k-1}, t_k)$ we can write (1) as

$$\begin{aligned}
\mathbf{x}(t_k) &= e^{\mathbf{F}(t_k - t_0)}\mathbf{x}(t_0) + \sum_{j=1}^{k} \int_{t_{j-1}}^{t_j} e^{\mathbf{F}(t_j - s)}\mathbf{L}w(s)\ \mathrm{d}s \\
&= e^{k\mathbf{F}\Delta t}\mathbf{x}(t_0) + \sum_{j=1}^{k} \int_{t_{j-1}}^{t_j} e^{\mathbf{F}(t_j - s)}\ \mathrm{d}s\mathbf{L}w(t_{j-1})
\end{aligned}$$

Now, because the of the constant time intervals, the integral inside the summation is also constant:

$$\begin{aligned}
&= e^{k\mathbf{F}\Delta t}\mathbf{x}(t_0) + \int_0^{\Delta t} e^{\mathbf{F}(\Delta t - s)}\ \mathrm{d}s\mathbf{L} \sum_{j=1}^{k} w(t_{j-1}) \\
&= e^{\mathbf{F}\Delta t}\mathbf{x}(t_{k-1}) + \int_0^{\Delta t} e^{\mathbf{F}(\Delta t - s)}\ \mathrm{d}s\mathbf{L}w(t_{k-1}) \\
&= \mathbf{A}\mathbf{x}(t_{k-1}) + \mathbf{B}w(t_{k-1})
\end{aligned}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & e^{-c^2\Delta t} \\ e^{\Delta t} & 1 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \Delta \\ e^{\Delta t} - 1 \end{bmatrix}$$

C)

Assuming $w_{k-1} \sim N(0, \frac{q_c}{\Delta t})$ and $r_k \sim N(0, R_k)$ and that at each timestep t_k we measure the value of $x(t_k)$ and additive zero mean gaussian noise with variance R_k and designating $\mathbf{x}_k = \mathbf{x}(t_k)$ and $w_k = w(t_k)$ we get the following linear Gaussian state space model:

$$\begin{aligned}
\mathbf{x}_k &= \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}w_{k-1} \\
y_k &= \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{x_k} + r_k
\end{aligned}$$

so that

$$\begin{aligned}
\mathbf{x}_k | \mathbf{x}_{k-1} &\sim N\left(\mathbf{A}\mathbf{x}_{k-1},\ \mathbf{B}\frac{q_c}{\Delta t}\mathbf{B}^T\right) \\
y_k | \mathbf{x}_k &\sim N\left(\begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{x}_k,\ R_k\right)
\end{aligned}$$

D)

Assuming $c = 0$ we have

$$\mathbf{M} = e^{(\Delta t - s)\mathbf{F}} \mathbf{L} q_c \mathbf{L}^T \left(e^{(\Delta t - s)\mathbf{F}} \right)^T = \begin{bmatrix} q_c & q_c e^{(\Delta t - s)} \\ q_c e^{(\Delta t - s)} & q_c e^{2(\Delta t - s)} \end{bmatrix}$$

so that

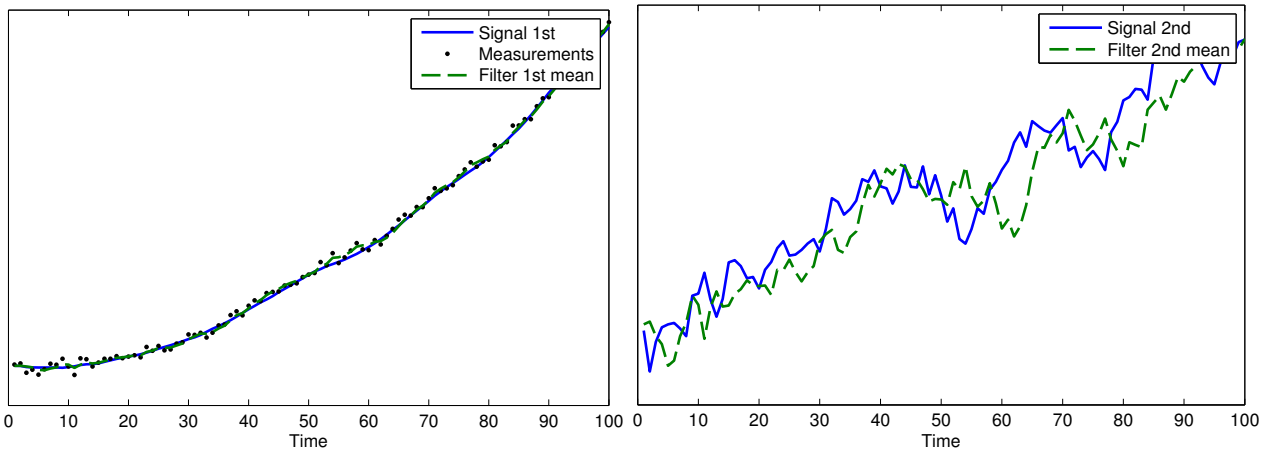
$$\mathbf{Q}_{k-1} = \int_0^{\Delta t} \mathbf{M} \, \mathrm{d}s = \begin{bmatrix} q_c \Delta t & q_c \left(e^{\Delta t} - 1 \right) \\ q_c \left(e^{\Delta t} - 1 \right) & \frac{q_c}{2} \left(e^{2\Delta t} - 1 \right) \end{bmatrix}$$

The corresponding covariance matrix in the ZOH discretation case is

$$\mathbf{B} \frac{q_c}{\Delta t} \mathbf{B}^T = \begin{bmatrix} q_c \Delta t & q_c \left(e^{\Delta t} - 1 \right) \\ q_c \left(e^{\Delta t} - 1 \right) & \frac{q_c}{\Delta t} \left(e^{2\Delta t} - 2e^{\Delta t} + 1 \right) \end{bmatrix}$$

Exercise 1.3

In this exercise Kalman filtering is practised with the help of the EKF/UKF toolbox. The results are illustrated in figure 1, the RMSE values are presented in table 1 and the m-code is listed in listing 1.



(a) First components and measurements

(b) Second components

Figure 1: Illustrations of the signal, the measurements and the Kalman filter means in exercise 1.3

Table 1: RMSE values for the first component of the signal when the Kalman filter means are used as the estimate and when the measuremenst are used as the estimate in exercise 1.3

	Filter	**Measurements**
RMSE	60.89	103.11

Listing 1: m-code in exercise 1.3

```
%% true signal

A = [1 1;0 1];
Q = diag([1/100 1]);
H = [1 0];
R = 100;

n = 100;
x0 = mvnrnd([0;0],eye(2))';

X = [x0 zeros(2,n-1)];
for k=2:n
    X(:,k) = A*X(:,k-1)+mvnrnd([0;0],Q)';
end
Y = X(1,:)+sqrt(R)*randn(1,n);

%% Filter

MM = zeros(2,n);
PP = zeros(2,2,n);
M = zeros(2,1);
P = eye(2);
for k=1:size(Y,2)
    %
    % Track with KF
    %
    [M,P] = kf_predict(M,P,A,Q);
    [M,P] = kf_update(M,P,Y(k),H,R);

    MM(:,k) = M;
    PP(:,:,k) = P;
end

if 1 %plot
    plot(1:n,X(1,:),1:n,Y,'k.',1:n,MM(1,:),'--','MarkerSize',8);
    set(gca,'YTick',[]);
    legend('Signal 1st','Measurements','Filter 1st mean');
    xlabel('Time');
    exportplot('ex_1_3_signal.pdf',figW,figH,gcf,1.5);
    figure;
    plot(1:n,X(2,:),1:n,MM(2,:),'--');
    set(gca,'YTick',[]);
    legend('Signal 2nd','Filter 2nd mean');
    xlabel('Time');
    exportplot('ex_1_3_derivative.pdf',figW,figH,gcf,1.5);

    rLabels = {'RMSE'};
    cLabels = {'Filter' 'Measurements'};
    matrix2latex([rmse(X(1,:),MM(1,:)) rmse(X(1,:),Y)],'ex_1_3_rmse.tex',...
           'alignment','d{?}{2}','format','$%.5f$','columnLabels',cLabels,...
           'rowLabels',rLabels,'rowLabelAlignment','r');
end
```

Round 2

Exercise 2.1

A)

The posterior is of the form

$$p(\mathbf{a}|y_{1:n}) = Ze^{f(\mathbf{a})}$$

where the exponent $f(\mathbf{a}) : \mathbb{R}^2 \to \mathbb{R}$ can be written as

$$\begin{aligned} f(\mathbf{a}) &= -\frac{1}{2}\left((\mathbf{X}\mathbf{a}-\mathbf{y})^T(\mathbf{X}\mathbf{a}-\mathbf{y}) + \frac{1}{\sigma^2}\mathbf{a}^T\mathbf{a}\right) \\ &= -\frac{1}{2}\left((\mathbf{a}-\mathbf{m})^T\mathbf{P}^{-1}(\mathbf{a}-\mathbf{m})\right) \end{aligned}$$

with suitably defined mean $\mathbf{m}$ and covariance matrix $\mathbf{P}$. Here $\mathbf{a}$, $\mathbf{X}$ and $\mathbf{y}$ are defined as in Round 1 exercise 1.

B)

The maximum of the posterior, which also is its mean in this case, is at the maximum of $f(\mathbf{a})$, which can be found at the point where its gradient vanishes:

$$\frac{\partial f(\mathbf{a})}{\partial \mathbf{a}} = -\mathbf{X}^T(\mathbf{X}\mathbf{a}-\mathbf{y}) - \frac{1}{\sigma^2}^T\mathbf{a} = -\left(\mathbf{X}^T\mathbf{X} + \frac{1}{\sigma^2}\mathbf{I}\right)\mathbf{a} + \mathbf{X}^T\mathbf{y}$$

so that at the maximum $\mathbf{m}$

$$\begin{aligned} \mathbf{X}^T\mathbf{X}\mathbf{m} + \frac{1}{\sigma^2}\mathbf{m} &= \mathbf{X}^T\mathbf{y} \\ \mathbf{m} &= \left(\mathbf{X}^T\mathbf{X} + \frac{1}{\sigma^2}\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y} \end{aligned}$$

C)

The Hessian matrix of f is

$$\frac{\partial^2 f(\mathbf{a})}{\partial \mathbf{a}^2} = -\left(\mathbf{X}^T\mathbf{X} + \frac{1}{\sigma^2}\mathbf{I}\right).$$

In order to relate this to $\mathbf{P}$, we can calculate that

$$\frac{\partial^2}{\partial \mathbf{a}^2}\left[-\frac{1}{2}\left((\mathbf{a}-\mathbf{m})^T\mathbf{P}^{-1}(\mathbf{a}-\mathbf{m})\right)\right] = -\mathbf{P}^{-1} \tag{2}$$

so that

$$\mathbf{P} = \left(\mathbf{X}^T\mathbf{X} + \frac{1}{\sigma^2}\mathbf{I}\right)^{-1}$$

D)

The resulting posterior distribution is then

$$\begin{aligned} p(\mathbf{a}|y_{1:n}) &= \mathrm{N}(\mathbf{a}|\mathbf{m},\mathbf{P}) \\ \mathbf{m} &= \mathbf{P}\mathbf{X}^T\mathbf{y} \\ \mathbf{P} &= \left(\mathbf{X}^T\mathbf{X} + \frac{1}{\sigma^2}\mathbf{I}\right)^{-1} \end{aligned}$$

Comparing $\mathbf{m}$ to the LS estimate $\mathbf{m}_{\mathrm{LS}}$ in Round 1 Exercise 1 we can see that if the prior variance σ^2 approaches infinity, then $\mathbf{m} \to \mathbf{m}_{\mathrm{LS}}$.

Exercise 2.2

The linear regression model in the last exercise can be written in the following state space form

$$\begin{aligned}
\mathbf{a}_k &= \mathbf{a}_{k-1} = \mathbf{a} = \begin{bmatrix} a_1 & a_2 \end{bmatrix}^T \sim N(\mathbf{a}|\mathbf{o}, \sigma^2\mathbf{I}) \\
y_k &= \mathbf{H}_k\mathbf{a} + \varepsilon_k,\ k = 1, \ldots, n \\
\mathbf{H}_k &= \begin{bmatrix} x_k & 1 \end{bmatrix} \\
\varepsilon_k &\sim N(\varepsilon_k|0, 1)
\end{aligned}$$

From these definitions we can deduce that the measurement distribution is

$$p(y_k|x_k, \mathbf{a}) = N(y_k|\mathbf{H}_k\mathbf{a}, 1)$$

As before, we are interested in the posterior distribution of $\mathbf{a}$. The "states" x_k are fixed and have no associated uncertainty.

A)

After the first observation we have

$$p(\mathbf{a}|y_1, x_1) = \frac{p(y_1|x_1, \mathbf{a})p(\mathbf{a})}{p(y_1|x_1)}$$

Now noting that $p(\mathbf{a}) = N(a_1|0, \sigma^2)N(a_2|0, \sigma^2)$ and designating $Z = p(y_1|x_1)^{-1}$, we find that this distribution is exactly the same as the posterior in the previous exercise with $n = 1$. Similarly after $k \leq n$ measurements we get (the measurements are considered i.i.d)

$$p(\mathbf{a}|y_{1:k}, x_{1:k}) = \frac{\prod_{j=1}^{k} p(y_j|x_j, \mathbf{a})p(\mathbf{a})}{p(y_{1:k}|x_{1:k})}.$$

that is again the same as the posterior in the previous exercise with $n = k$.

We can then use the results from the last exercise by replacing n with k in the equations. If we designate by $\mathbf{X}_k$ and $\mathbf{y}_k$ the $\mathbf{X}$ and $\mathbf{y}$ of the previous exercise but with k instead of n elements and with $\mathbf{m}_k$ and $\mathbf{P}_k$ the mean and covariance matrix of the posterior after k measurements, we get

$$\begin{aligned}
\mathbf{m}_k &= \mathbf{P}_k\mathbf{X}_k^T\mathbf{y}_k \\
\mathbf{P}_k &= \left(\mathbf{X}_k^T\mathbf{X}_k + \frac{1}{\sigma^2}\mathbf{I}\right)^{-1}
\end{aligned} \tag{3}$$

B)

For the covariance matrix we can write

$$\begin{aligned}
\mathbf{X}_k^T\mathbf{X}_k &= \begin{bmatrix} \sum_i^k x_i^2 & \sum_i^k x_i \\ \sum_i^k x_i & k \end{bmatrix} \\
\mathbf{H}_k^T\mathbf{H}_k &= \begin{bmatrix} x_k^2 & x_k \\ x_k & 1 \end{bmatrix} \\
\Leftrightarrow \mathbf{X}_k^T\mathbf{X}_k &= \mathbf{X}_{k-1}^T\mathbf{X}_{k-1} + \mathbf{H}_k^T\mathbf{H}_k
\end{aligned}$$

giving

$$\mathbf{P}_k = \left(\mathbf{X}_{k-1}^T \mathbf{X}_{k-1} + \mathbf{H}_k^T \mathbf{H}_k + \frac{1}{\sigma^2} \mathbf{I} \right)^{-1}.$$

Similarly for the mean we get

$$\begin{aligned} \mathbf{X}_k^T \mathbf{y}_k &= \begin{bmatrix} \sum_i^k x_i y_i \\ \sum_i^k y_i \end{bmatrix} \\ \mathbf{H}_k^T y_k &= \begin{bmatrix} x_k y_k \\ y_k \end{bmatrix} \\ \Leftrightarrow \mathbf{X}_k^T \mathbf{y}_k &= \mathbf{X}_{k-1}^T \mathbf{y}_{k-1} + \mathbf{H}_k^T y_k \end{aligned}$$

giving

$$\mathbf{m}_k = \mathbf{P}_k \mathbf{X}_{k-1}^T \mathbf{y}_{k-1} + \mathbf{P}_k \mathbf{H}_k^T y_k \tag{4}$$

c)

Let's start by substituting first $\mathbf{K}_k$ and then $\mathbf{S}_k$ into $\mathbf{P}_k$

$$\begin{aligned} \mathbf{P}_k &= \mathbf{P}_{k-1} - \mathbf{P}_{k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1} \mathbf{S}_k \mathbf{S}_k^{-1} \mathbf{H}_k \mathbf{P}_{k-1} \\ &= \mathbf{P}_{k-1} - \mathbf{P}_{k-1} \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_{k-1} \mathbf{H}_k^T + 1 \right)^{-1} \mathbf{H}_k \mathbf{P}_{k-1} \end{aligned} \tag{5}$$

apply the matrix inversion lemma

$$\begin{aligned} &= \left(\mathbf{P}_{k-1}^{-1} + \mathbf{H}_k^T \mathbf{H}_k \right)^{-1} \\ &= \left(\mathbf{X}_{k-1}^T \mathbf{X}_{k-1} + \frac{1}{\sigma^2} \mathbf{I} + \mathbf{H}_k^T \mathbf{H}_k \right)^{-1} \\ &= \left(\mathbf{X}_k^T \mathbf{X}_k + \frac{1}{\sigma^2} \mathbf{I} \right)^{-1} \end{aligned}$$

d)

In part b) it was proved that the result in part a) can be written as in (4). By using the identities $\mathbf{K}_k = \mathbf{P}_k \mathbf{H}_k^T = \mathbf{P}_{k-1} \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_{k-1} \mathbf{H}_k^T + 1 \right)^{-1}$ this can be derived from the Kalman filter equations:

$$\begin{aligned} \mathbf{m}_k &= \mathbf{m}_{k-1} + \mathbf{K}_k (\mathbf{y}_k - \mathbf{H}_k \mathbf{m}_{k-1}) \\ &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \, \mathbf{m}_{k-1} + \mathbf{K}_k \mathbf{y}_k \end{aligned}$$

apply equation (3)

$$\begin{aligned} &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \, \mathbf{P}_{k-1} \mathbf{X}_{k-1}^T \mathbf{y}_{k-1} + \mathbf{K}_k \mathbf{y}_k \\ &= (\mathbf{P}_{k-1} - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_{k-1}) \, \mathbf{X}_{k-1}^T \mathbf{y}_{k-1} + \mathbf{K}_k \mathbf{y}_k \end{aligned}$$

apply the identities

$$= \left(\mathbf{P}_{k-1} - \mathbf{P}_{k-1} \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_{k-1} \mathbf{H}_k^T + 1 \right)^{-1} \mathbf{H}_k \mathbf{P}_{k-1} \right) \mathbf{X}_{k-1}^T \mathbf{y}_{k-1} + \mathbf{P}_k \mathbf{H}_k^T \mathbf{y}_k$$

apply equation (5)

$$= \mathbf{P}_k \mathbf{X}_{k-1}^T \mathbf{y}_{k-1} + \mathbf{P}_k \mathbf{H}_k^T y_k$$

E)

- step 1
  - mean

$$\begin{aligned} \mathbf{m}_0 &= 0 \\ \mathbf{P}_0 &= \sigma^2 \mathbf{I} \\ \Leftrightarrow \mathbf{m}_1 &= \mathbf{m}_0 + \mathbf{P}_1 \mathbf{H}_1^T (\mathbf{y}_1 - \mathbf{H}_1 \mathbf{m}_0) \\ &= \mathbf{P}_1 \mathbf{X}_1^T \mathbf{y}_1 \end{aligned}$$

  - variance

$$\mathbf{P}_1 = \left(\mathbf{X}_1^T \mathbf{X}_1 + \mathbf{P}_0^{-1} \right)^{-1}$$

- step k
  - mean (assume $\mathbf{m}_{k-1} = \mathbf{P}_{k-1} \mathbf{X}_{k-1}^T \mathbf{y}_{k-1}$)

$$\begin{aligned} \mathbf{m}_k &= \mathbf{m}_{k-1} + \mathbf{K}_k (\mathbf{y}_k - \mathbf{H}_k \mathbf{m}_{k-1}) \\ &= \mathbf{P}_k \mathbf{X}_{k-1}^T \mathbf{y}_{k-1} + \mathbf{P}_k \mathbf{H}_k^T y_k \\ &= \mathbf{P}_k \mathbf{X}_k^T \mathbf{y}_k \end{aligned}$$

  - variance (assume $\mathbf{P}_{k-1} = \left(\mathbf{X}_{k-1}^T \mathbf{X}_{k-1} + \mathbf{P}_0^{-1} \right)^{-1}$)

$$\begin{aligned} \mathbf{P}_k &= \left(\mathbf{P}_{k-1}^{-1} + \mathbf{H}_k^T \mathbf{H}_k \right)^{-1} \\ &= \left(\mathbf{X}_{k-1}^T \mathbf{X}_{k-1} + \mathbf{P}_0^{-1} + \mathbf{H}_k^T \mathbf{H}_k \right)^{-1} \\ &= \left(\mathbf{X}_k^T \mathbf{X}_k + \mathbf{P}_0^{-1} \right)^{-1} \end{aligned}$$

Exercise 2.3

A)

$$\begin{aligned} p(\mathbf{x}) &= N(\mathbf{x}|\mathbf{m}, \mathbf{P}), \ \mathbf{x} \in \mathbb{R}^n \\ p(\mathbf{y}|\mathbf{x}) &= N(\mathbf{y}|\mathbf{H}\mathbf{x}, \mathbf{R}), \ \mathbf{y} \in \mathbb{R}^m \\ \Leftrightarrow p(\mathbf{x}, \mathbf{y}) &= p(\mathbf{y}|\mathbf{x}) p(\mathbf{x}) \\ &= C \exp\left(-\frac{1}{2} (\mathbf{x} - \mathbf{m})^T \mathbf{P}^{-1} (\mathbf{x} - \mathbf{m}) - \frac{1}{2} (\mathbf{y} - \mathbf{H}\mathbf{x})^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{H}\mathbf{x}) \right) \\ &= C \exp\left(-\frac{1}{2} \begin{bmatrix} \mathbf{x} - \mathbf{m} \\ \mathbf{y} - \mathbf{H}\mathbf{m} \end{bmatrix}^T \begin{bmatrix} \mathbf{P} & \mathbf{P}\mathbf{H}^T \\ \mathbf{H}\mathbf{P} & \mathbf{H}^T \mathbf{P} \mathbf{H} + \mathbf{R} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{x} - \mathbf{m} \\ \mathbf{y} - \mathbf{H}\mathbf{m} \end{bmatrix} \right) \end{aligned}$$

Now from the last form we can see that the joint distribution is clearly normal, which means that all the marginal distributions must be normal too. To get the mean and variance of a variable from its conditional distribution, we can use the following well known identities (easily provable by writing the expectations as integrals):

$$\begin{aligned} \mathrm{E}\left[\mathbf{y}\right] &= \mathrm{E}\left[\mathrm{E}\left[\mathbf{y}|\mathbf{x}\right]\right] \\ \mathrm{var}\left(\mathbf{y}\right) &= \mathrm{E}\left[\mathrm{var}\left(\mathbf{y}|\mathbf{x}\right)\right] + \mathrm{var}\left(\mathrm{E}\left[\mathbf{y}|\mathbf{x}\right]\right) \end{aligned}$$

Now it's easy to see that

$$\begin{aligned}
\mathrm{E}\left[\mathbf{y}\right] &= \mathrm{E}\left[\mathbf{Hx}\right] = \mathbf{H}\mathrm{E}\left[\mathbf{x}\right] = \mathbf{Hm} \\
\mathrm{var}\left(\mathbf{y}\right) &= \mathrm{E}\left[\mathbf{R}\right] + \mathrm{var}\left(\mathbf{Hx}\right) = \mathbf{R} + \mathbf{H}\mathrm{var}\left(\mathbf{x}\right)\mathbf{H}^T = \mathbf{R} + \mathbf{HPH}^T \\
&\Leftrightarrow \mathbf{y} \sim N(\mathbf{Hm}, \mathbf{HPH}^T + \mathbf{R})
\end{aligned}$$

B)

$$\begin{aligned}
p(\mathbf{x}) &= N(\mathbf{x}|\mathbf{m}, \mathbf{P}) \\
p(\mathbf{y}|\mathbf{x}) &= N(\mathbf{y}|\mathbf{Hx}, \mathbf{R}) \\
\Leftrightarrow p(\mathbf{x}, \mathbf{y}) &= p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \\
&= \frac{1}{(2\pi)^{\frac{n+m}{2}}\sqrt{|\mathbf{P}|\,|\mathbf{R}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T\mathbf{P}^{-1}(\mathbf{x}-\mathbf{m}) - \frac{1}{2}(\mathbf{y}-\mathbf{Hx})^T\mathbf{R}^{-1}(\mathbf{y}-\mathbf{Hx})\right)
\end{aligned}$$

$$\begin{aligned}
\Rightarrow p(\mathbf{y}) &= \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \\
&= \frac{1}{(2\pi)^{\frac{m}{2}}\sqrt{\left|\mathbf{HPH}^T + \mathbf{R}\right|}} \exp\left(-\frac{1}{2}(\mathbf{y}-\mathbf{Hm})^T(\mathbf{HPH}^T + \mathbf{R})^{-1}(\mathbf{y}-\mathbf{Hm})\right)
\end{aligned}$$

C)

Here we prove the following result: let

$$\begin{bmatrix}\mathbf{x} \\ \mathbf{y}\end{bmatrix}^T \sim N\left(\begin{bmatrix}\mathbf{a} \\ \mathbf{b}\end{bmatrix}^{-1}, \begin{bmatrix}\mathbf{A} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{B}\end{bmatrix}\right)$$

then

$$\mathbf{x}|\mathbf{y} \sim N\left(\mathbf{a} + \mathbf{C}\mathbf{B}^{-1}(\mathbf{y}-\mathbf{b}), \mathbf{A} - \mathbf{C}\mathbf{B}^{-1}\mathbf{C}^T\right)$$

Let's denote the inverse of the joint covariance matrix as

$$\begin{bmatrix}\mathbf{A} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{B}\end{bmatrix}^{-1} = \begin{bmatrix}\mathbf{D}_{11} & \mathbf{D}_{12} \\ \mathbf{D}_{12}^T & \mathbf{D}_{22}\end{bmatrix}$$

and then expand the quadratic form in the exponent:

$$\begin{aligned}
f(\mathbf{x}, \mathbf{y}) &= -\frac{1}{2}\begin{bmatrix}\mathbf{x}-\mathbf{a} \\ \mathbf{y}-\mathbf{b}\end{bmatrix}^T \begin{bmatrix}\mathbf{D}_{11} & \mathbf{D}_{12} \\ \mathbf{D}_{12}^T & \mathbf{D}_{22}\end{bmatrix} \begin{bmatrix}\mathbf{x}-\mathbf{a} \\ \mathbf{y}-\mathbf{b}\end{bmatrix} \\
&= -\frac{1}{2}\begin{bmatrix}(\mathbf{x}-\mathbf{a})^T & (\mathbf{y}-\mathbf{b})^T\end{bmatrix} \begin{bmatrix}\mathbf{D}_{11}(\mathbf{x}-\mathbf{a}) & \mathbf{D}_{12}(\mathbf{y}-\mathbf{b}) \\ \mathbf{D}_{12}^T(\mathbf{x}-\mathbf{a}) & \mathbf{D}_{22}(\mathbf{y}-\mathbf{b})\end{bmatrix} \\
&= -\frac{1}{2}\left((\mathbf{x}-\mathbf{a})^T\mathbf{D}_{11}(\mathbf{x}-\mathbf{a}) + 2(\mathbf{x}-\mathbf{a})^T\mathbf{D}_{12}(\mathbf{y}-\mathbf{b}) + (\mathbf{y}-\mathbf{b})^T\mathbf{D}_{22}^T(\mathbf{y}-\mathbf{b})\right)
\end{aligned}$$

In this Gaussian case the mean is also the maximum and the maximum of the exponential function can be found at the maximum of the exponent. Thus if we take the partial derivative of the exponent with respect to $\mathbf{x}$ (meaning that $\mathbf{y}$ is held fixed) and see where it vanishes, we have found the mean $\mathbf{m}$ of $\mathbf{x}|\mathbf{y}$:

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{y}) = 0 \\
\Leftrightarrow\ & -\mathbf{D}_{11}(\mathbf{m} - \mathbf{a}) - \mathbf{D}_{12}(\mathbf{y} - \mathbf{b}) = 0 \\
\Leftrightarrow\ & \mathbf{m} = \mathbf{D}_{11}^{-1}\left(\mathbf{D}_{11}\mathbf{a} - \mathbf{D}_{12}(\mathbf{y} - \mathbf{b})\right)
\end{aligned}$$

apply the identity $\mathbf{D}_{12} = -\mathbf{D}_{11}\mathbf{C}\mathbf{B}^{-1}$

$$\begin{aligned}
&= \mathbf{a} + \mathbf{D}_{11}^{-1}\mathbf{D}_{11}\mathbf{C}\mathbf{B}^{-1}(\mathbf{y} - \mathbf{b}) \\
&= \mathbf{a} + \mathbf{C}\mathbf{B}^{-1}(\mathbf{y} - \mathbf{b})
\end{aligned}$$

The variance can be found similarly by taking the second partial derivative of the exponent (the Hessian matrix) with respect to $\mathbf{x}$ and applying the identity $\mathbf{D}_{11}^{-1} = \mathbf{A} - \mathbf{C}\mathbf{B}^{-1}\mathbf{C}^T$:

$$\begin{aligned}
\frac{\partial^2}{\partial \mathbf{x}^2} f(\mathbf{x}, \mathbf{y}) &= \frac{\partial}{\partial \mathbf{x}}\left(-\mathbf{D}_{11}(\mathbf{x} - \mathbf{a}) - \mathbf{D}_{12}(\mathbf{y} - \mathbf{b})\right) \\
&= -\left(\mathbf{A} - \mathbf{C}\mathbf{B}^{-1}\mathbf{C}^T\right)^{-1}
\end{aligned}$$

After applying equation (2), we note that the result has been proven.

Round 3

Exercise 3.1

We now have the non-zero mean noise state space model

$$\begin{aligned}
\mathbf{x}_k &= \mathbf{A}\mathbf{x}_{k-1} + \mathbf{q}_{k-1} \\
\mathbf{y}_k &= \mathbf{H}\mathbf{x}_\mathbf{k} + \mathbf{r}_k \\
\mathbf{q}_{k-1} &\sim N\left(\mathbf{m}_q, \mathbf{Q}\right) \\
\mathbf{r}_k &\sim N\left(\mathbf{m}_r, \mathbf{R}\right)
\end{aligned}$$

meaning

$$\begin{aligned}
\mathbf{x}_k|\mathbf{x}_{k-1} &\sim N\left(\mathbf{A}\mathbf{x}_{k-1} + \mathbf{m}_q, \mathbf{Q}\right) \\
\mathbf{y}_k|\mathbf{x}_k &\sim N\left(\mathbf{H}\mathbf{x}_\mathbf{k} + \mathbf{m}_r, \mathbf{R}\right)
\end{aligned}$$

By following closely the derivation of the Kalman filter equations in the course material, we get

prediction:

$$\begin{aligned}
\mathbf{m}_k^- &= \mathbf{A}\mathbf{m}_{k-1} + \mathbf{m}_q \\
\mathbf{P}_k^- &= \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}
\end{aligned}$$

update:

$$\begin{aligned}
\mathbf{v}_k &= \mathbf{y}_k - \mathbf{H}\mathbf{m}_k^- - \mathbf{m}_r \\
\mathbf{S}_k &= \mathbf{H}\mathbf{P}_k^-\mathbf{H} + \mathbf{R} \\
\mathbf{K}_k &= \mathbf{P}_k^-\mathbf{H}^T\mathbf{S}_k^{-1} \\
\mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\mathbf{v}_k \\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T
\end{aligned}$$

Exercise 3.2

Here we filter a Gaussian random walk model with the Kalman filter. The model is

$$\begin{aligned}
x_k &= x_{k-1} + q_{k-1} \\
y_k &= x_k + r_k \\
q_{k-1} &\sim N(0, Q) \\
r_k &\sim N(0, R)
\end{aligned}$$

meaning

$$\begin{aligned}
x_k|x_{k-1} &\sim N(x_{k-1}, Q) \\
y_k|x_k &\sim N(x_k, R)
\end{aligned}$$

In this one-dimensional case approximating the required integrals using a grid approximation is feasible. In figure 2 the random walk signal with $n = 100$ timesteps is plotted with the means given by the Kalman filter and the grid approximation, where the y-axis has been discretized into 500 equidistant points. In figure 3 the variances of the approximated Gaussian distributions are plotted for the Kalman filter and the grid approximation. As can be seen, thThe m-code is presented in listing 2.

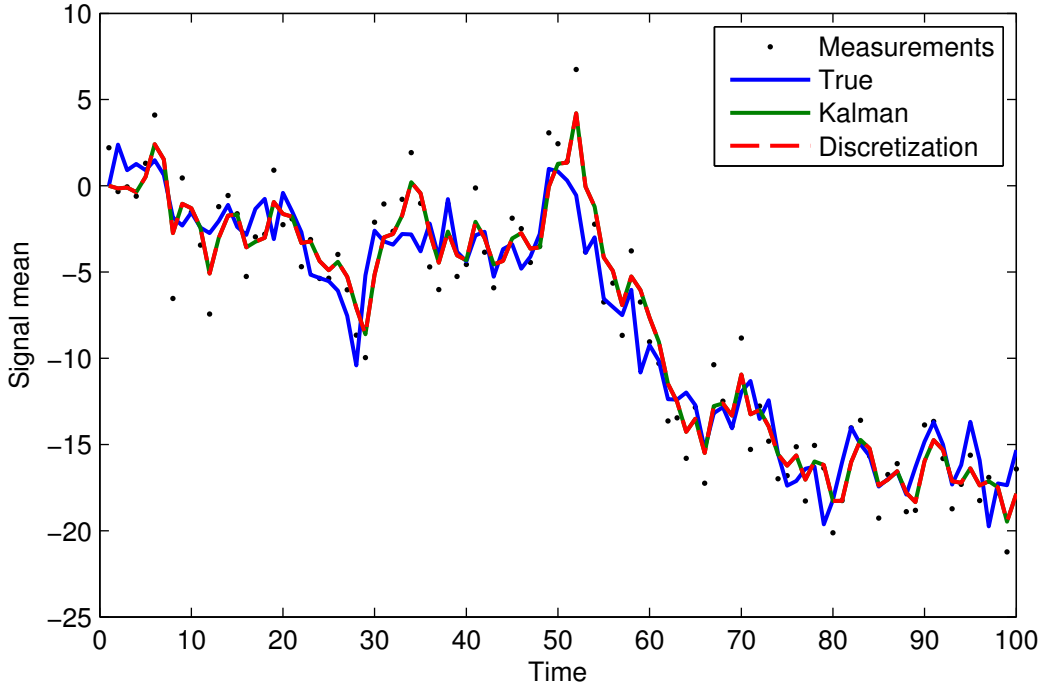


Figure 2: The true random walk signal (blue), the mean of the Kalman filter (red) and the mean of the grid approximation (green) in exercise 3.2

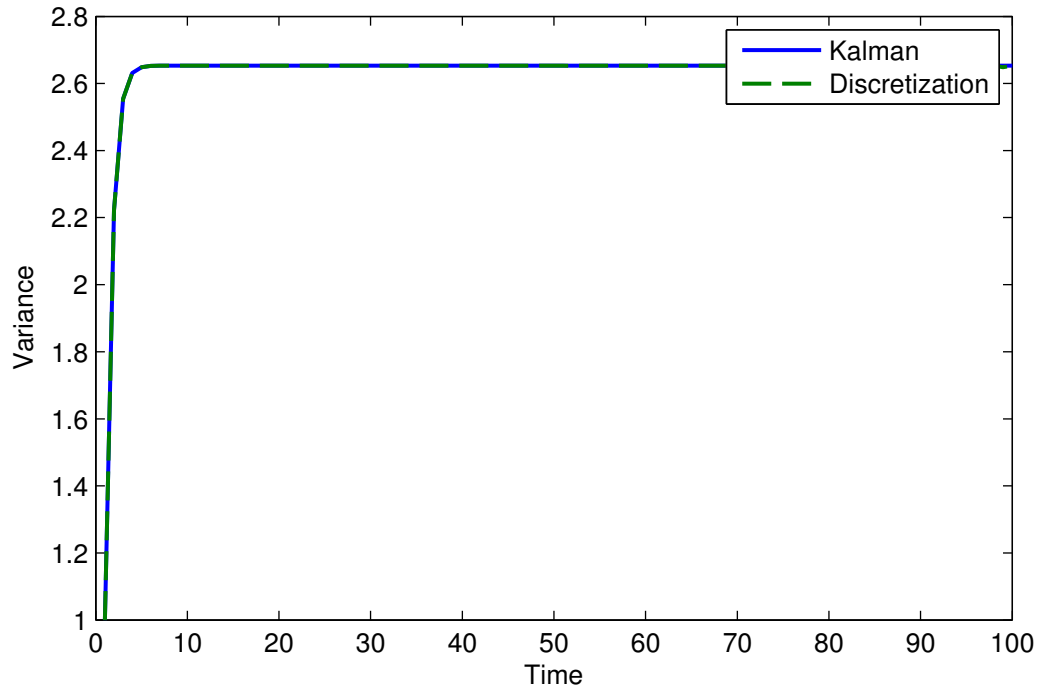


Figure 3: The true random walk signal (blue), the mean of the Kalman filter (red) and the mean of the grid approximation (green) in exercise 3.2

Listing 2: m-code in exercise 3.2

```
%% Exercise round 3, exercise 2

%true signal
N=100;
q=3;

% signal
s=cumsum([0 normrnd(0,sqrt(q),1,N-1)]);

% measurements
A = 1;
H = 1;
r = 5;
mn = 0;
n = N;
x = floor(1:N/n:N);
y = s(1,x)+normrnd(mn,sqrt(r),1,n);

%% Kalman filter
m = 0;
P = 1;
ms = zeros(1,n);ms(1)=m;
Ps = zeros(1,n);Ps(1)=P;
Ks = zeros(1,n-1);
for k=2:n
    % update
    K = (P+q)/(P+q+r);
    m = (1-K)*m+K*y(k);
    P = P+q-K^2*(P+q+r);
    ms(k) = m;
    Ps(k) = P;
    Ks(k-1) = K;
end

mso = ms;
Pso = Ps;
R(1) = rmse(x,ms);

% discretization
a=min(y)-0.15*(max(y)-min(y));
b=max(y)+0.15*(max(y)-min(y));
N = 1000;
t = linspace(a,b,N);
[TX,TY] = meshgrid(t);
p_dyn = normpdf(TX,TY,sqrt(q));
m = 0;
P = 1;
p_ = normpdf(t,m,P);
ms = zeros(1,n);ms(1)=m;
Ps = zeros(1,n);Ps(1)=P;
distr = zeros(N,n);
for k=2:n
    p = sum(p_dyn.*repmat(p_',1,N));
    p = normpdf(y(k),t,sqrt(r)).*p;
    p = p/sum(p);
    p_ = p;
    distr(:,k) = p';
    m = t*p';
    ms(k) = m; % mean
    Ps(k) = (t-m).^2*p';
end;
dms = ms;
dPs = Ps;
```

Exercise 3.3

Here we consider the Kalman filter for a noisy resonator model. The model is presented as a state-space model in the exercise paper and is not reproduced here. The state $\mathbf{x}_k \in \mathbb{R}^2$ at time k consists of the location $x_k^{(1)}$ and its derivative $x_k^{(2)}$.

A)

We compare the Kalman filter solution to a base line solution, where the measurement y_k is directly used as $x_k^{(1)}$ and $x_k^{(2)}$ is calculated as a weighted average of the measurement differences. The base line solution is presented in figure 4 and the Kalman filter solution in figure 5

Unsurprisingly, the Kalman filter solution is clearly better than the baseline solution, which is directly affected by the noise. The root-mean-square errors for the solutions are presented in table 2.

B)

Here the stationary Kalman filter, where the Kalman gain $\mathbf{K}_k$ is constant, is compared to the solutions in 3.3A. The constant for the Kalman gain was computed numerically by running the filter for a long time. The graphical solution is presented in figure 6 and the RMSE in table 2. The RMSE is a little smaller for the stationary Kalman filter solution. Comparing the graphical solutions of the ordinary Kalman filter and the stationary Kalman filter carefully, it can be seen that during the first few steps, the stationary Kalman filter makes a better estimate. But since the Kalman gain seems to converge to its constant value very quickly, the solutions are close to identical after the first ten steps.

Table 2: The RMSE values in exercise 3.3

	Baseline	**Kalman**	**Stat. Kalman**
RMSE	0.54	0.24	0.23

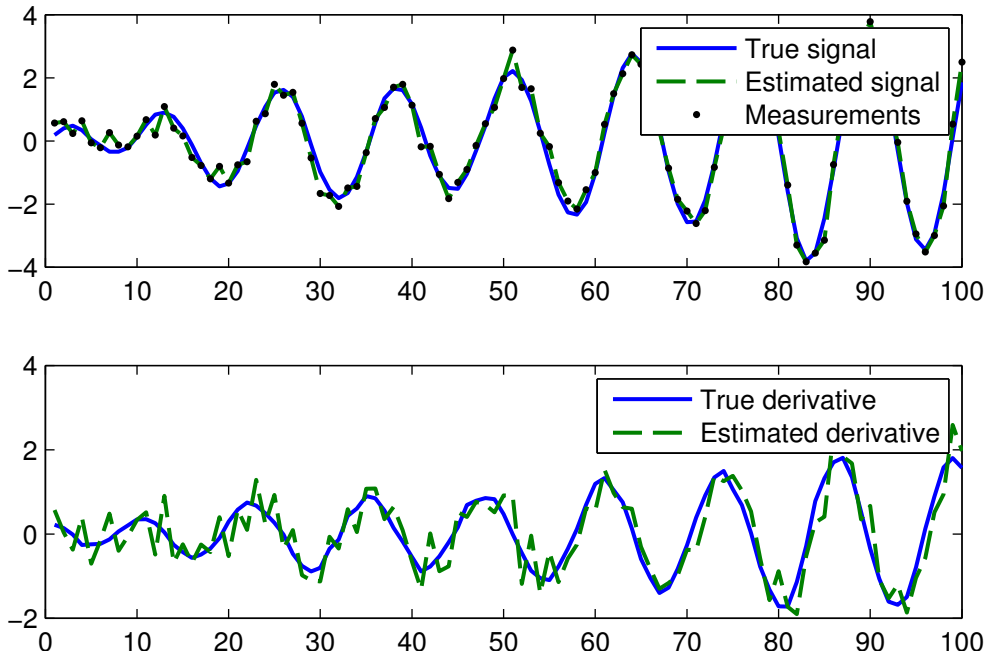


Figure 4: The base line solution in exercise 3.3

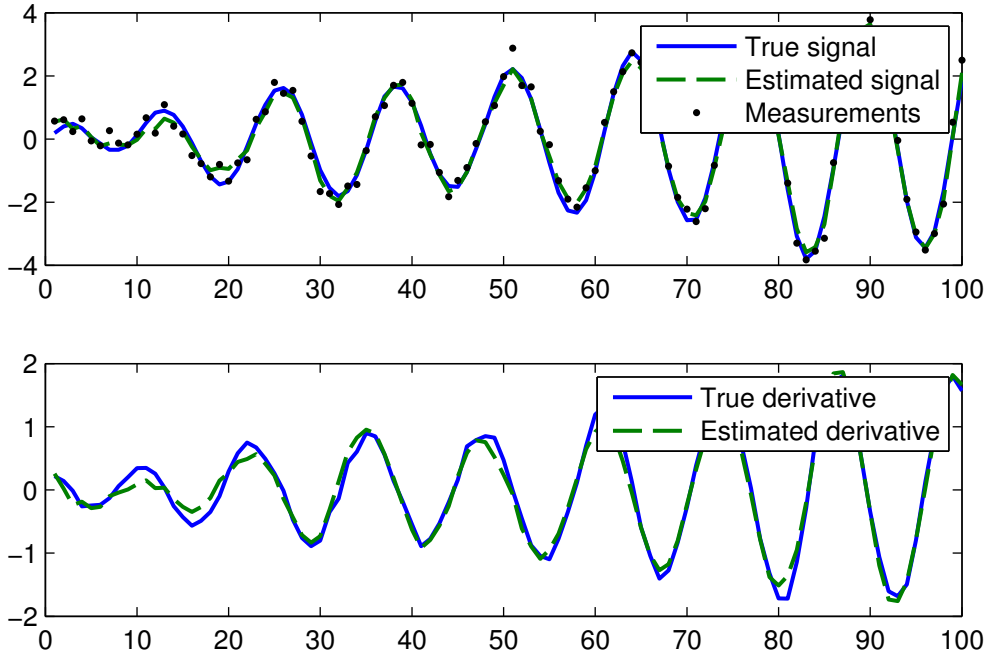


Figure 5: The Kalman filter solution in exercise 3.3

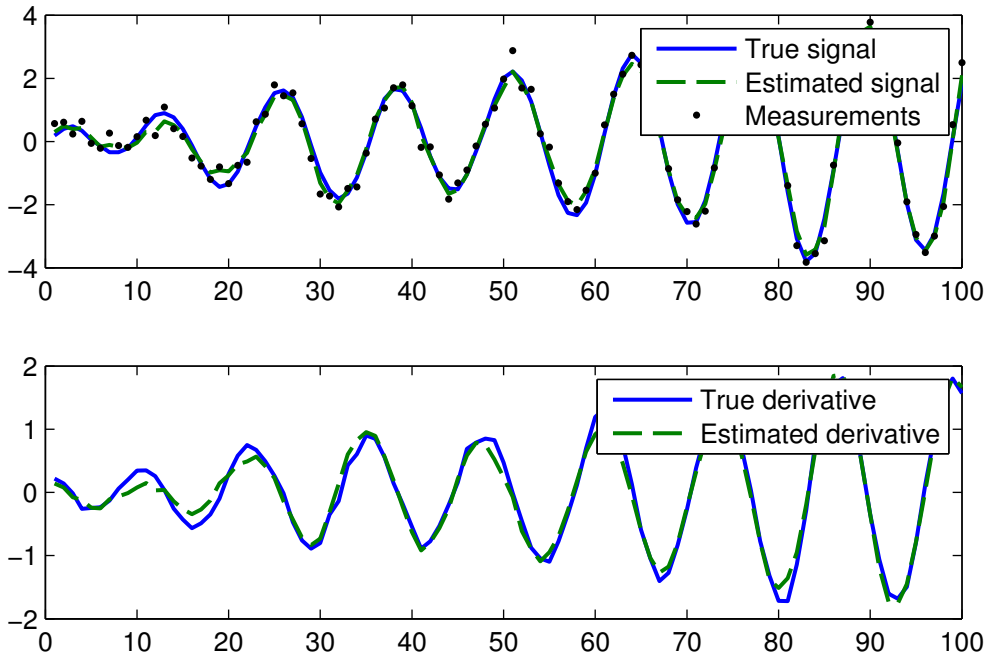


Figure 6: The stationary Kalman filter solution in exercise 3.3

Listing 3: m-code in exercise 3.3

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Kalman filter solution
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

m = x0;     % Initialize to true value
P = eye(2); % Some uncertanty in covariance
EST2 = zeros(2,steps);
EST2P = zeros(2,2,steps);

Ks = EST2;
H = [1 0];
kdiff = zeros(1,steps);
for k=1:steps
  % prediction
  m_ = A*m;
  P_ = A*P*A'+Q;

  % update
  K = P_(:,1)/(P_(1,1)+r);
  Ks(:,k) = K;
  m = m_ + K*(Y(k)-m_(1));
  P = P_-(P_(1,1)+r)*(K*K');

  % Store the results
  EST2(:,k) = m;
  EST2P(:,:,k) = P;
end

% Plot the signal and its estimate
figure;
subplot(2,1,1);
plot(T,X(1,:),'-',T,EST2(1,:),'--',T,Y,'k.','MarkerSize',8);
legend('True signal','Estimated signal','Measurements');

% Plot the derivative and its estimate
subplot(2,1,2);
plot(T,X(2,:),'-',T,EST2(2,:),'--');
legend('True derivative','Estimated derivative');
exportplot('ex_3_3_kalmansol.pdf',figW,figH,gcf,1.5);

% Compute error
err2 = rmse(X,EST2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Stationary Kalman filter solution
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

m3 = x0;     % Initialize to true value
K  =  [0.424975200169185;
       0.111137301539005];  % Store the stationary gain here

EST3 = zeros(2,steps);

for k=1:steps
  % Replace these with the stationary Kalman filter equations
  m3 = A*m3+K*(Y(k)-A(1,:)*m3);

  % Store the results
  EST3(:,k) = m3;
end
```

Round 4

Exercise 4.1

In this exercise we consider the following non-linear state space model:

$$\begin{aligned} x_k &= x_{k-1} - 0.01\sin(x_{k-1}) + q_{k-1} \\ &= f(x_{k-1}) + q_{k-1} \\ y_k &= 0.5\sin(2x_k) + r_k \\ &= h(x_k) + r_k \\ q_{k-1} &\sim N\left(0, 0.01^2\right) \\ r_k &\sim N\left(0, 0.02\right) \end{aligned}$$

A)

To implement the extended Kalman filter for the model, we need the following derivatives:

$$\begin{aligned} f'(x) &= -0.01\cos(x) + 1 \\ h'(x) &= \cos(2x) \end{aligned}$$

In order to use the Kalman filter for this problem, $p\left(x_k|x_{k-1}\right)$ and $p\left(y_k|x_k\right)$ need to be approximated by a Gaussian distribution. In the EKF this is done by using linear approximations to the nonlinearities. The result is a Gaussian approximation to the filtering distribution. The result of applying EKF to this problem is presented in figure 7, where the signal and measurements were simulated for 200 timesteps, starting from $x_0 = \frac{2}{5}\pi$.

B)

In statistically linearized filter (SLF) the linear approximation of the EKF is replaced by statistical linearization. In order to use the SLF, we need to derive the expectations for $f(x)$ and $h(x)$.

Let's start with $f(x)$:

$$\mathrm{E}\left[f(x)\right] = m - 0.01\int_{-\infty}^{\infty}\left(\sin(x)\right)N(x|m,P)\ \mathrm{d}x$$

To calculate this integral, let's look at the inverse Fourier transform $g(x)$ of the Gaussian pdf $N(x|m,P)$:

$$\begin{aligned} g(x) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} N(\omega|m,P)e^{i\omega x}\ \mathrm{d}\omega \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} N(\omega|m,P)\left(\cos(\omega x) + i\sin(\omega x)\right)\ \mathrm{d}\omega \end{aligned}$$

from where we can deduce, that the integral we're looking for is equal to $2\pi\mathrm{Im}[g(1)]$, where "Im" denotes the imaginary part. Happily we also know that

$$
\begin{aligned}
g(x) &= \frac{1}{2\pi} e^{-\frac{1}{2}\left(Px^2 - 2mxi\right)} \\
\Rightarrow 2\pi \text{Im}[g(1)] &= \sin(m) e^{-\frac{1}{2}P} \\
\Rightarrow \text{E}\left[f(x)\right] &= m - 0.01 \sin(m) e^{-\frac{1}{2}P}.
\end{aligned}
$$

From this we can readily infer that $\text{E}\left[h(x)\right] = 2\pi \text{Im}[g(2)]$, giving

$$
\text{E}\left[h(x)\right] = \frac{1}{2} \sin(2m) e^{-2P}
$$

We also need the expectations of $f(x)(x-m)$ and $h(x)(x-m)$:

$$
\begin{aligned}
\text{E}\left[f(x)(x-m)\right] &= \text{E}\left[xf(x)\right] - m\text{E}\left[f(x)\right] \\
&= \text{E}\left[x^2\right] - 0.01\text{E}\left[x \sin(x)\right] - m\text{E}\left[f(x)\right]
\end{aligned}
$$

To calculate $\text{E}\left[x \sin(x)\right]$, we again use the fact that the answer is $2\pi \text{Im}[g_2(1)]$, where g_2 is the inverse Fourier transform of $xN(x|m, P)$ that we happen to know to be

$$
g_2(x) = \frac{1}{2\pi} e^{-\frac{1}{2}\left(Px^2\right)} \left(m + xPi\right) e^{imx}
$$

giving

$$
\begin{aligned}
2\pi \text{Im}[g_2(1)] &= e^{-\frac{1}{2}(P)} \left(P \cos(m) + m \sin(m)\right) \\
\Rightarrow \text{E}\left[f(x)(x-m)\right] &= P - 0.01P \cos(m) e^{-\frac{1}{2}(P)}
\end{aligned}
$$

and finally along the same lines we get

$$
\text{E}\left[h(x)(x-m)\right] = P \cos(2m) e^{-2(P)}
$$

Putting these together, we have

$$
\begin{aligned}
\text{E}\left[f(x_{k-1})\right] &= m_{k-1} - 0.01 \sin(m_{k-1}) e^{-\frac{1}{2}P_{k-1}} \\
\text{E}\left[f(x_{k-1})\delta x_{k-1}\right] &= P_{k-1} - 0.01 \cos(m_{k-1}) P_{k-1} e^{-\frac{1}{2}P_{k-1}}
\end{aligned}
$$

where expectations are with respect to $N(x_{k-1}|m_{k-1}, P_{k-1})$ and

$$
\begin{aligned}
\text{E}\left[h(x_k)\right] &= 0.5 \sin(2m_k^-) e^{-2P_k^-} \\
\text{E}\left[h(x_k)\delta x_k\right] &= \cos(2m_k^-) P_k^- e^{-2P_k^-}
\end{aligned}
$$

where expectations are with respect to $N(x_k|m_k^-, P_k^-)$

The results of applying the SLF in the same situation as in 4.1A are presented also in figure 7. The RMSE's for both of the methods is presented in table 3 and the mcode is shown in listing 4.

Table 3: The RMSE values in exercise 4.1

	EKF	**SLF**
RMSE	0.67	0.69

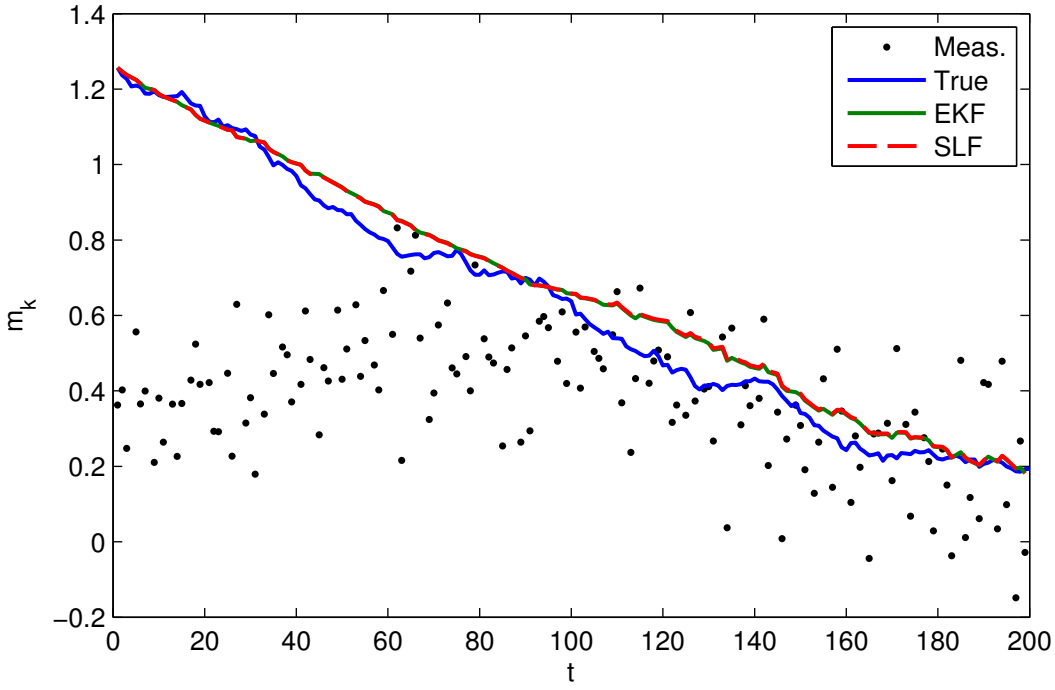


Figure 7: The true signal and the EKF and SLF approximations in exercise 4.1

Listing 4: m-code in exercise 4.1

```
%% true signal

N=200;
q=0.01^2;

s=zeros(1,N);
f=@(x)x-0.01*sin(x);
F=@(x)1-0.01*cos(x);
h=@(x)0.5*sin(2*x);
H=@(x)cos(2*x);
Y=s;
x0 = 0.4*pi;
R = [];
%x0 = 0;
x = x0;
s(1) = x0;
for k=2:N
    x = f(x)+sqrt(q)*randn;
    s(k) = x;
end

% measurements
r = 0.02;
mn = 0;
n = 150;
x = floor(1:N/n:N);
y = h(s(1,x))+normrnd(0,sqrt(r),1,n);

% EKF

m = x0;
P = q;
ms = [m zeros(1,n-1)];
Ps = [P zeros(1,n-1)];
```

```
for k=2:n
    % prediction
    m_= f(m);
    P_ = F(m)^2*P+q;
    % update
    v = y(k)-h(m_);
    S = H(m_)^2*P_+r;
    K = P_*H(m_)/S;
    m = m_+K*v;
    P = P_-K^2*S;
    ms(k) = m;
    Ps(k) = P;
end
ms_ekf = ms;
Ps_ekf = Ps;
fprintf('EKF %3.4f\n',rmse(s(x),ms));
R(1) = rmse(s(x),ms);

% SLF

Ef=@(m,p)m-0.01*sin(m)*exp(-1*p/2);
Efdx=@(m,p)p-0.01*cos(m)*p*exp(-p/2);
Eh=@(m,p)0.5*sin(2*m)*exp(-2*p);
Ehdx=@(m,p)cos(2*m)*p*exp(-2*p);

m = x0;
P = q;
ms = [m zeros(1,n-1)];
Ps = [P zeros(1,n-1)];
for k=2:n
    % prediction
    m_= Ef(m,P);
    P_ = Efdx(m,P)^2/P+q;
    % update
    v = y(k)-Eh(m_,P_);
    S = Ehdx(m_,P_)^2/P_+r;
    K = Ehdx(m_,P_)/S;
    m = m_+K*v;
    P = P_-K^2*S;
    ms(k) = m;
    Ps(k) = P;
end
ms_slkf = ms;
Ps_slkf = Ps;
```

Exercise 4.2

Here we derive an alternative form of SLF.

A)

Let $\mathbf{x} \in \mathbb{R}^d$, $\mathbf{G}_\mathbf{x}(\mathbf{x}) = \dfrac{\partial \mathbf{g}(\mathbf{x})}{\partial \mathbf{x}}$ and $N(\mathbf{x}|\mathbf{m}, \mathbf{P})$ is the gaussian pdf of $\mathbf{x}$ with mean $\mathbf{m}$ and covariance matrix $\mathbf{P}$, then:

$$\begin{aligned}
\mathrm{E}\left[\mathbf{g}(\mathbf{x})(\mathbf{x}-\mathbf{m})^T\right] &= \int_{\mathbb{R}^d} \mathbf{g}(\mathbf{x})(\mathbf{x}-\mathbf{m})^T N(\mathbf{x}|\mathbf{m},\mathbf{P}) \ \mathrm{d}\mathbf{x} \\
&= \int_{\mathbb{R}^d} \mathbf{g}(\mathbf{x})\mathbf{P}\mathbf{P}^{-1}(\mathbf{x}-\mathbf{m})^T N(\mathbf{x}|\mathbf{m},\mathbf{P}) \ \mathrm{d}\mathbf{x} \\
&= \int_{\mathbb{R}^d} -\mathbf{g}(\mathbf{x})\mathbf{P} \frac{\partial}{\partial \mathbf{x}} N(\mathbf{x}|\mathbf{m},\mathbf{P}) \ \mathrm{d}\mathbf{x} \\
&= \lim_{\mathbf{x}\to-\infty}\left(\mathbf{g}(\mathbf{x})\mathbf{P}N(\mathbf{x}|\mathbf{m},\mathbf{P})\right) - \lim_{\mathbf{x}\to\infty}\left(\mathbf{g}(\mathbf{x})\mathbf{P}N(\mathbf{x}|\mathbf{m},\mathbf{P})\right) + \int_{\mathbb{R}^d} \mathbf{G}_\mathbf{x}(\mathbf{x}) N(\mathbf{x}|\mathbf{m},\mathbf{P}) \ \mathrm{d}\mathbf{x}\mathbf{P} \\
&= \mathrm{E}\left[\mathbf{G}_\mathbf{x}(\mathbf{x})\right]\mathbf{P}
\end{aligned}$$

B)

Let $\boldsymbol{\mu}(\mathbf{m}) = \mathrm{E}\left[\mathbf{g}(\mathbf{x})\right] = \int_{\mathbb{R}^d} \mathbf{g}(\mathbf{x}) N(\mathbf{x}|\mathbf{m},\mathbf{P}) \ \mathrm{d}\mathbf{x}$, then

$$\begin{aligned}
\frac{\partial \boldsymbol{\mu}}{\partial \mathbf{m}} &= \int_{\mathbb{R}^d} \mathbf{g}(\mathbf{x}) \frac{\partial}{\partial \mathbf{m}} N(\mathbf{x}|\mathbf{m},\mathbf{P}) \ \mathrm{d}\mathbf{x} \\
&= \int_{\mathbb{R}^d} -\mathbf{g}(\mathbf{x})\mathbf{P}^{-1}(\mathbf{x}-\mathbf{m})^T N(\mathbf{x}|\mathbf{m},\mathbf{P}) \ \mathrm{d}\mathbf{x} \\
&= \mathrm{E}\left[\mathbf{G}_\mathbf{x}(\mathbf{x})\right]
\end{aligned}$$

C)

Using the result of 4.2A and denoting the Jacobians of $\mathbf{f}$ and $\mathbf{h}$ by $\mathbf{F}_\mathbf{x}$ and $\mathbf{H}_\mathbf{x}$ respectively, we can write the SLF equations as

prediction:

$$\begin{aligned}
\mathbf{m}_k^- &= \mathrm{E}\left[\mathbf{f}(\mathbf{x}_{k-1})\right] \\
\mathbf{P}_k^- &= \mathrm{E}\left[\mathbf{F}_\mathbf{x}(\mathbf{x}_{k-1})\right]\mathbf{P}_{k-1}^T \mathrm{E}\left[\mathbf{F}_\mathbf{x}(\mathbf{x}_{k-1})\right]^T + \mathbf{Q}_{k-1}
\end{aligned}$$

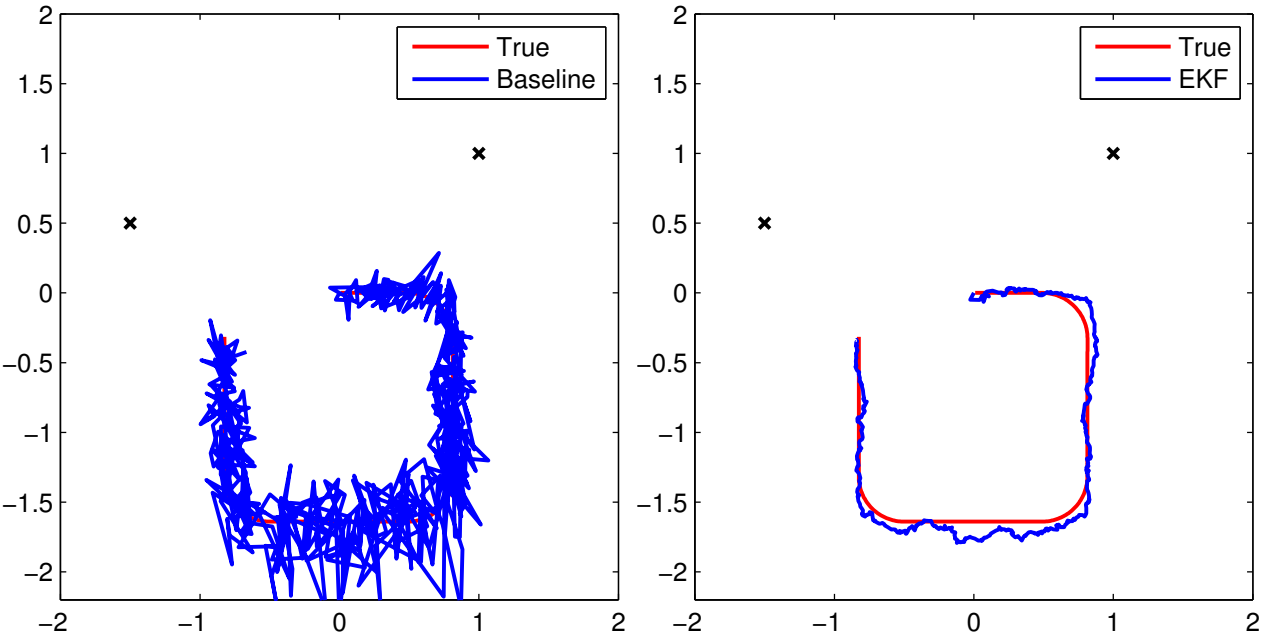
update:

$$\begin{aligned}
\mathbf{v}_k &= \mathbf{y}_k - \mathrm{E}\left[\mathbf{h}(\mathbf{x}_k)\right] \\
\mathbf{S}_k &= \mathrm{E}\left[\mathbf{H}_\mathbf{x}(\mathbf{x}_k)\right](\mathbf{P}_k^-)^T \mathrm{E}\left[\mathbf{H}_\mathbf{x}(\mathbf{x}_k)\right]^T + \mathbf{R}_k \\
\mathbf{K}_k &= \mathrm{E}\left[\mathbf{H}_\mathbf{x}(\mathbf{x}_k)\right]^T \mathbf{P}_k^- \mathbf{S}_k^{-1} \\
\mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\mathbf{v}_k \\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T
\end{aligned}$$

Exercise 4.3

A)

In this exercise the classical problem of bearings only taget tracking is considered. The problem setting is not reproduced here, but an EKF was implemented for an approximate solution. The graphical results are presented in figures 8a and 8b, the RMSE value in table 4 and the mcode in listing 5.



(a) Baseline approximation

(b) EKF approximation

Figure 8: The true trajectory, the baseline approximation and the EKF approximation in exercise 4.3A

Table 4: The RMSE values in exercise 4.3A

	Baseline	**EKF**
RMSE	1.02	0.44

Listing 5: m-code in exercise 4.3A

```
%%%% EKF %%%%%%%%%%

function [o,hh]=ekf()
    m = x0;            % Initialize to true value
    P = eye(4);        % Some uncertainty
    ms = [m zeros(4,steps-1)];
    for k=2:steps
        %% Compute estimate here
        m_ = A*m;
        P_ = A*P*A'+ Q;
        y = Theta(:,k);
        v =  y - h(m_);
        S_ = H(m_)*P_*H(m_)'+R;
        K = P_*H(m_)'/S_;
        m = m_+K*v;
        P = P_-K*S_*K';
        ms(:,k) = m;
        Ps(:,:,k) = P;
    end

    %% Compute error
    fprintf('EKF %3.4f\n',rmse(X,ms));
    o = ms;
    hh = showtrace('b');
    axis([-2 2 -2.5 1.5]);
end

function o=h(x)
    o = atan2(x(2)-S(2,:)',x(1)-S(1,:)');
end

function o=H(x)
    H1 = @(xx,s)-(xx(2)-s(2))/((xx-s)'*(xx-s));
    H2 = @(xx,s)(xx(1)-s(1))/((xx-s)'*(xx-s));
    o = zeros(size(S,2),4);
    for ii = 1:size(S,2)
        o(ii,1:2) = [H1(x(1:2),S(:,ii)) H2(x(1:2),S(:,ii))];
    end
end
```

Round 5

Exercise 5.1

Here we implement the unscented Kalman filter for the problem in exercise 4.1. The parameters of the UKF transformation (which affect the sigma point spread) were chosen as follows:

$$\alpha = 12$$
$$\beta = 0$$
$$\kappa = 1$$

The graphical results are illustrated in figure 9, where the UKF solution is compared with the SLF solution and the true signal. The RMSE values are tabulated in table 5. The mcode in shown listing 6.

Table 5: The RMSE values in exercise 5.1

	EKF	SLF	UKF
RMSE	0.67	0.69	0.78

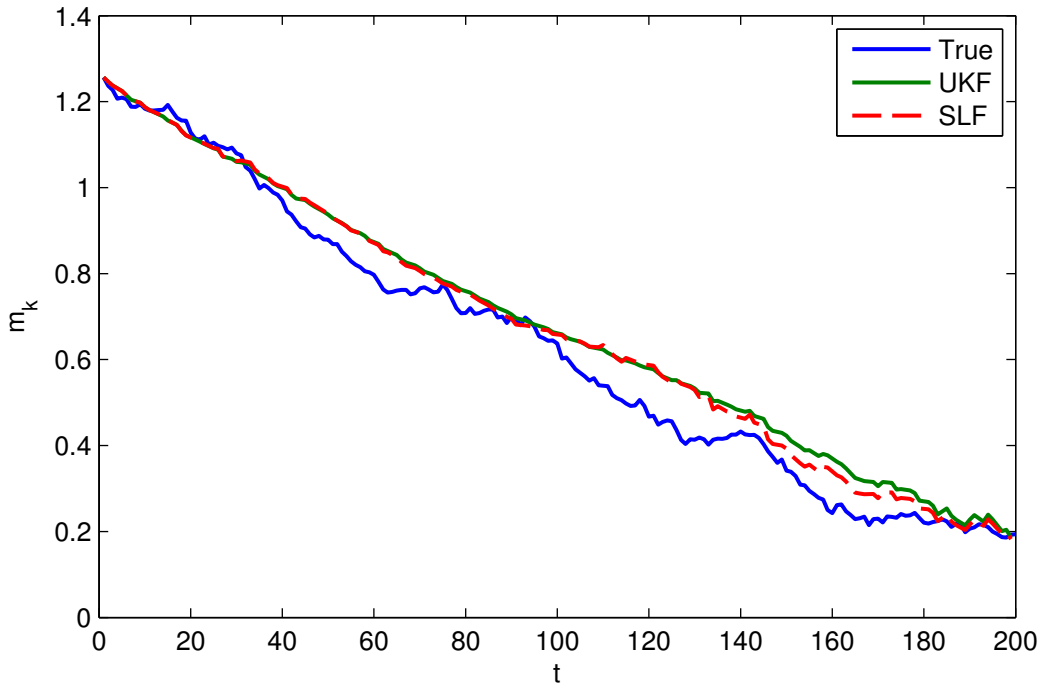


Figure 9: The true signal and the UKF and SLF approximations in exercise 5.1

Listing 6: m-code in exercise 5.1

```
%% UKF

m = x0;
P = q;
ms = [m zeros(1,n-1)];
Ps = [P zeros(1,n-1)];

alpha = 12; beta = 0; kappa = 1; % parameters
lambda = alpha^2*(1+kappa) - 1;

Wi = 1/(2*(1+lambda));
Wm = [lambda/(1+lambda) Wi Wi]; % mean weights
Wc = [lambda/(1+lambda)+(1-alpha^2) Wi Wi]; % covariance weights
e = sqrt(lambda+1)*[0 1 -1]';
for k=2:n
    % prediction
    sig = m*ones(3,1)+sqrt(P)*e;
    % propagate through dynamic model
    sig = f(sig);
    m_ = Wm*sig;
    P_ = Wc*((sig-m_).^2)+q;

    % update
    sig_ = m_*ones(3,1)+sqrt(P_)*e;
    % propagate through measurement model
    sigY = h(sig_);

    u = Wm*sigY;
    S = Wc*((sigY-u).^2)+r;
    C = Wc*((sig_-m_).*(sigY-u));
    K = C/S;
    m = m_+K*(y(k)-u);
    P = P_-K^2*S;
    ms(k) = m;
    Ps(k) = P;
end
```

Exercise 5.2

Here also the Gauss-Hermite Kalman filter (GHKF) and the cubature Kalman filter (CKF) were implemented for the previous problem. The solutions of these methods were indistinguishable no matter what the order of the polynomial chosen for GHKF. It is possible to make the UKF, GHKF and CKF methods identical by choosing the UKF parameters as $\alpha = 1$ and $\kappa = 0$ and the GHKF polynomial order as 2. The RMSE values of all the considered methods are presented in table 6.The m-code is presented in listing 7.

Table 6: The RMSE values in exercise 5.2

	EKF	**SLF**	**UKF**	**GHKF**	**CKF**
RMSE	0.67	0.69	0.78	0.69	0.69

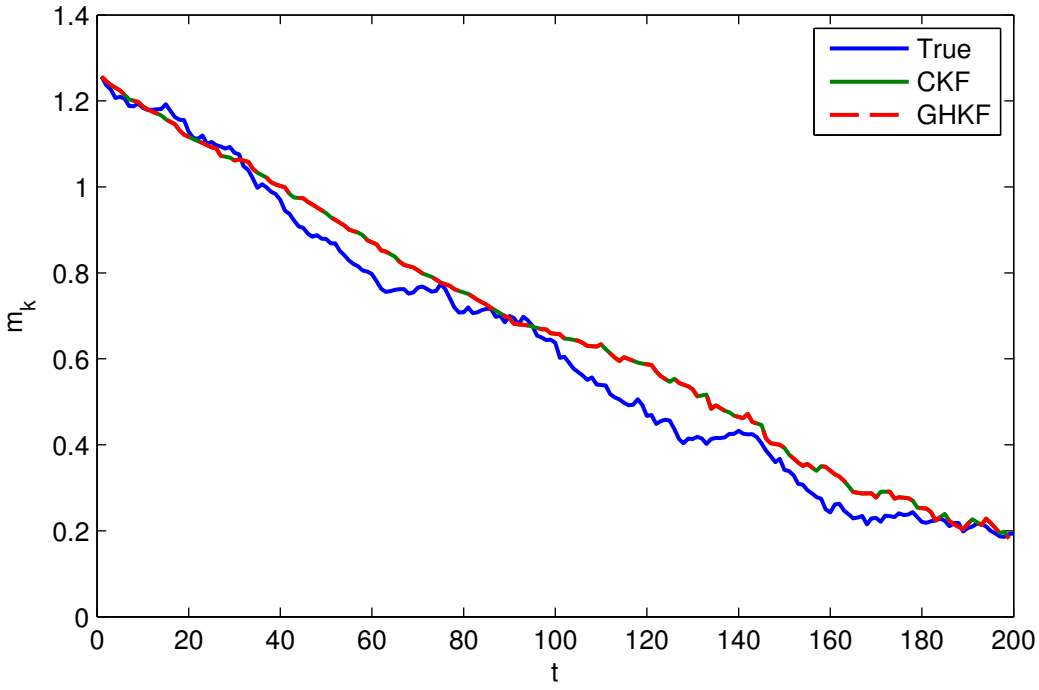


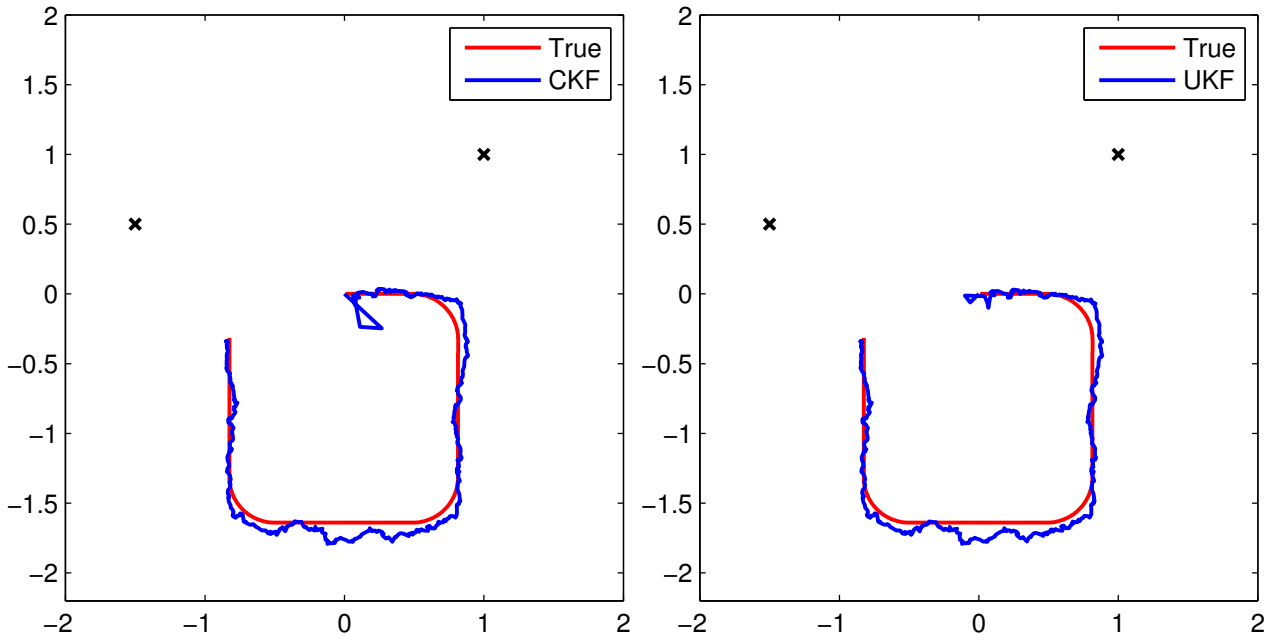
Figure 10: The true signal and the GHKF and CKF approximations in exercise 5.2

Listing 7: m-code in exercise 5.2

```
%% GHKF

m = x0;
P = q;
ms = [m zeros(1,n-1)];
Ps = [P zeros(1,n-1)];

p = 2; % order of the polynomial
e = roots(hermite(p)); % unit sigma points
W = factorial(p)./(p*hermite(p-1,e)').^2; % weights

for k=2:n
    % prediction
    sig = m*ones(p,1)+sqrt(P)*e;
    sig = f(sig);
    m_ = W*sig;
    P_ = W*((sig-m_).^2)+q;
    % update
    sig_ = m_*ones(p,1)+sqrt(P_)*e;
    sigY = h(sig_);

    u = W*sigY;
    S = W*((sigY-u).^2)+r;
    C = W*((sig_-m_).*(sigY-u));
    K = C/S;
    m = m_+K*(y(k)-u);
    P = P_-K^2*S;
    ms(k) = m;
    Ps(k) = P;
end
R(4) = rmse(s(x),ms);
ms_ghkf = ms;

% CKF

m = x0;
P = q;
ms = [m zeros(1,n-1)];
Ps = [P zeros(1,n-1)];

e = [1;-1]; % unit sigma points
p = length(e); % num of sigma points
W = ones(1,p);
W = W/sum(W);

for k=2:n
    % prediction
    sig = m*ones(p,1)+sqrt(P)*e;
    sig = f(sig);
    m_ = W*sig;
    P_ = W*((sig-m_).^2)+q;

    sig_ = m_*ones(p,1)+sqrt(P_)*e;
    sigY = h(sig_);

    u = W*sigY;
    S = W*((sigY-u).^2)+r;
    C = W*((sig_-m_).*(sigY-u));
    K = C/S;
    m = m_+K*(y(k)-u);
    P = P_-K^2*S;
    ms(k) = m;
    Ps(k) = P;
end
R(5) = rmse(s(x),ms);
ms_ckf = ms;
Ps_ckf = Ps;
```

Exercise 5.3

Here we implement the UKF and the CKF for the target tracking problem in exercise 4.3. The parameters for UKF were chosen so that there would some differences in the solution. It seems that increasing the parameter values makes the result worse in the beginning The graphical results are presented in figure 11, where the UKF and CKF solutions are compared and the mcode in shown listing 8. The updated RMSE values are presented in table 7. As can be seen, the CKF/UKF approximation is better but only by a very slight margin.



(a) CKF approximation

(b) UKF approximation

Figure 11: The true trajectory, the EKF approximation and the CKF/UKF approximation in exercise 5.3

Table 7: The RMSE values in exercise 5.3

	Baseline	**EKF**	**CKF**	**UKF**
RMSE	1.02	0.44	0.44	0.44

Listing 8: m-code in exercise 5.3

```
function o = ckf()

    m = x0;            % Initialize to true value
    P = eye(4);        % Some uncertainty
    ms = [m zeros(4,steps-1)];

    for k=2:steps
        [m,P] = ckf_(Theta(:,k));
        ms(:,k) = m;
        Ps(:,:,k) = P;
    end

      %% Compute error
    fprintf('CKF %3.4f\n',rmse(X,ms));
    o = ms;
    showtrace('b');
end

function [mo,Po] = ckf_(yy)
    n = 4;
    nn = size(S,2);
    e = [sqrt(n)*eye(n) -sqrt(n)*eye(n)]; % unit sigma points
    W = ones(1,2*n);
    W = W/sum(W);

    sig = repmat(m,1,2*n)+chol(P,'lower')*e;
    sig = A*sig;
    m_ = sig*W';
    m__ = repmat(m_,1,2*n);
    P_ = (sig-m__)*(sig-m__)'/(2*n)+Q;
    sig_ = m__+chol(P_,'lower')*e;
    sigY = zeros(nn,2*n);
    for j=1:2*n
        sigY(:,j)=h(sig(:,j));
    end
    u = sigY*W';
    u__ = repmat(u,1,2*n);
    S_ = (sigY-u__)*(sigY-u__)'/(2*n)+R;
    C = (sig_-m__)*(sigY-u__)'/(2*n);
    K = C/S_;
    mo = m_+K*(yy-u);
    Po = P_-K*S_*K';
end

function o=ukf()
    m = x0;            % Initialize to true value
    P = eye(4);        % Some uncertainty
    ms = [m zeros(4,steps-1)];
    n = 4;
    nn = size(S,2);
    alpha = 7; beta = 1; kappa = 1;
    lambda = alpha^2*(1+kappa) - 1;
    e = [zeros(n,1) sqrt(n+lambda)*eye(n) -sqrt(n+lambda)*eye(n)]; % unit sigma
         points
    Wm = [lambda/(n+lambda) 1/(2*(n+lambda))*ones(1,2*n)];
    Wc = [lambda/(n+lambda)+(1-alpha^2+beta) 1/(2*(n+lambda))*ones(1,2*n)];
    for k=2:steps

        % form sigma points
        sig = repmat(m,1,2*n+1)+chol(P,'lower')*e;
        % propagate through dynamic model
        sig = A*sig;
        % prediction
        m_ = sig*Wm';
        m__ = repmat(m_,1,2*n+1);
        P_ = (sig-m__)*diag(Wc)*(sig-m__)'+Q;
```

```
            % form sigma points
            sig_ = m__+chol(P_,'lower')*e;
            % propagate through measurement model
            sigY = zeros(nn,2*n);
            for j=1:(2*n+1)
                sigY(:,j)=h(sig_(:,j));
            end

            u = sigY*Wm';
            u__ = repmat(u,1,2*n+1);
            S_ = (sigY-u__)*diag(Wc)*(sigY-u__)'+R;
            C = (sig_-m__)*diag(Wc)*(sigY-u__)';
            K = C/S_;
            m = m_+K*(Theta(:,k)-u);
            P = P_-K*S_*K';
            ms(:,k) = m;
            Ps(:,:,k) = P;
        end
        o = ms;
        showtrace('b');
    end
```

Round 6

Exercise 6.1

A)

prediction:

$$\begin{aligned}\mathbf{m}_k^- &= \mathbf{A}\mathbf{m}_{k-1} \\ \mathbf{P}_k^- &= \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}\end{aligned}$$

update:

$$\begin{aligned}v_k &= y_k - \mathbf{H}\mathbf{m}_k^- \\ \mathbf{S}_k &= \mathbf{H}\mathbf{P}_k^-\mathbf{H} + R \\ \mathbf{K}_k &= \mathbf{P}_k^-\mathbf{H}^T\mathbf{S}_k^{-1} \\ \mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k v_k \\ \mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T\end{aligned}$$

where:

$$\begin{aligned}\mathbf{A} &= \begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix} \\ \mathbf{H} &= \begin{bmatrix}1 & 0\end{bmatrix} \\ \mathbf{Q} &= \begin{bmatrix}0.01 & 0\\ 0 & 1\end{bmatrix} \\ R &= 100\end{aligned}$$

B)

$$
\begin{aligned}
p(\mathbf{x}_k, y_k|\mathbf{x}_{k-1}) &= p(y_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{x}_{k-1}) \\
&= N\left(\begin{bmatrix}\mathbf{x}_k \\ y_k\end{bmatrix} \middle| \begin{bmatrix}\mathbf{A}\mathbf{x}_{k-1} \\ \mathbf{H}\mathbf{A}\mathbf{x}_{k-1}\end{bmatrix}, \begin{bmatrix}\mathbf{Q} & \mathbf{Q}\mathbf{H}^T \\ \mathbf{H}\mathbf{Q} & \mathbf{H}\mathbf{Q}\mathbf{H}^T + R\end{bmatrix}\right) \\
\Rightarrow \pi(\mathbf{x}_k) &= p(\mathbf{x}_k|\mathbf{x}_{k-1}, y_{1:k}) \\
&= p(\mathbf{x}_k|\mathbf{x}_{k-1}, y_k) = N\left(\mathbf{x}_k|\mathbf{m}_k^I, \mathbf{P}_k^I\right) \\
\mathbf{m}_k^I &= \mathbf{A}\mathbf{x}_{k-1} + \mathbf{Q}\mathbf{H}^T\left(\mathbf{H}\mathbf{Q}\mathbf{H}^T + R\right)^{-1}(y_k - \mathbf{H}\mathbf{A}\mathbf{x}_{k-1}) \\
\mathbf{P}_k^I &= \mathbf{Q} - \mathbf{Q}\mathbf{H}^T\left(\mathbf{H}\mathbf{Q}\mathbf{H}^T + R\right)^{-1}\mathbf{H}\mathbf{Q}
\end{aligned}
$$

C)

A pseudocode implementation of the SIR filter with N particles and $n+1$ time-steps using the above optimal importance distribution:

1. **Initialization**: Draw N samples from the prior (for example $N(\mathbf{o}, \mathbf{Q})$)

   $$\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$$

   and set the weights as

   $$w_0^{(i)} = \frac{1}{N}$$

   Set

   $$\mathbf{m}_0 = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_0^{(i)}$$

   $$\mathbf{P}_0 = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{x}_0^{(i)} - \mathbf{m}_0\right)^2$$

2. **`for k=1:n`** :

   - draw N samples from the importance distribution:

     $$\mathbf{x}_k^{(i)} \sim p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)}, y_k) = N\left(\mathbf{x}_k|\mathbf{m}_k^I, \mathbf{P}_k^I\right)$$

   - update the weights as

     $$w_k^{(i)} = w_{k-1}^{(i)}\frac{p(\mathbf{x}_k^{(i)}, y_k|\mathbf{x}_{k-1}^{(i)})}{p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}, y_k)}$$

     $$w_k^{(i)} = \frac{w_k^{(i)}}{\sum_{j=1}^{N} w_k^{(j)}}$$

   - set

     $$\mathbf{m}_k = \sum_{i=1}^{N} w_k^{(i)}\mathbf{x}_k^{(i)}$$

     $$\mathbf{P}_k = \sum_{i=1}^{N} w_k^{(i)}\left(\mathbf{x}_k^{(i)} - \mathbf{m}_k\right)^2$$

- possibly resample: `if` $\frac{1}{\sum_{i=1}^{N}\left(w_k^{(i)}\right)^2} < \frac{N}{10}$

```
% use inverse CDF sampling
po = p; % copy of the samples
cdf = cumsum(W); % the weights in W form the pdf
for l=1:N
    p(l) = po(find(cdf > rand,1));
end
```

D)

A Kalman filter can also be considered as a particle filter with a single particle. As this is a linear Gaussian case, the Kalman filter result is exact, so it could be considered the optimal particle filter in this case. Thus I would definitely choose the Kalman filter for a real implementation.

Exercise 6.2

Here we implement the Bootstrap and sequential importance resampling (SIR) particle filters for the problem in exercise 4.1. Both of the filters were run with $n = 700$ particles and the UKF filter was used in SIR to provide importance distribution. The graphical results are presented in figure 12 and the mcode is shown listing 9. The RMSE values are presented in table 8. As can be seen from the figure and the RMSE table, the SIR filter is somewhat better as was expected.

Table 8: The RMSE values in exercise 6.2

	EKF	**SLF**	**UKF**	**GHKF**	**CKF**	**BS**	**SIR**
RMSE	0.67	0.69	0.78	0.69	0.69	0.77	0.69

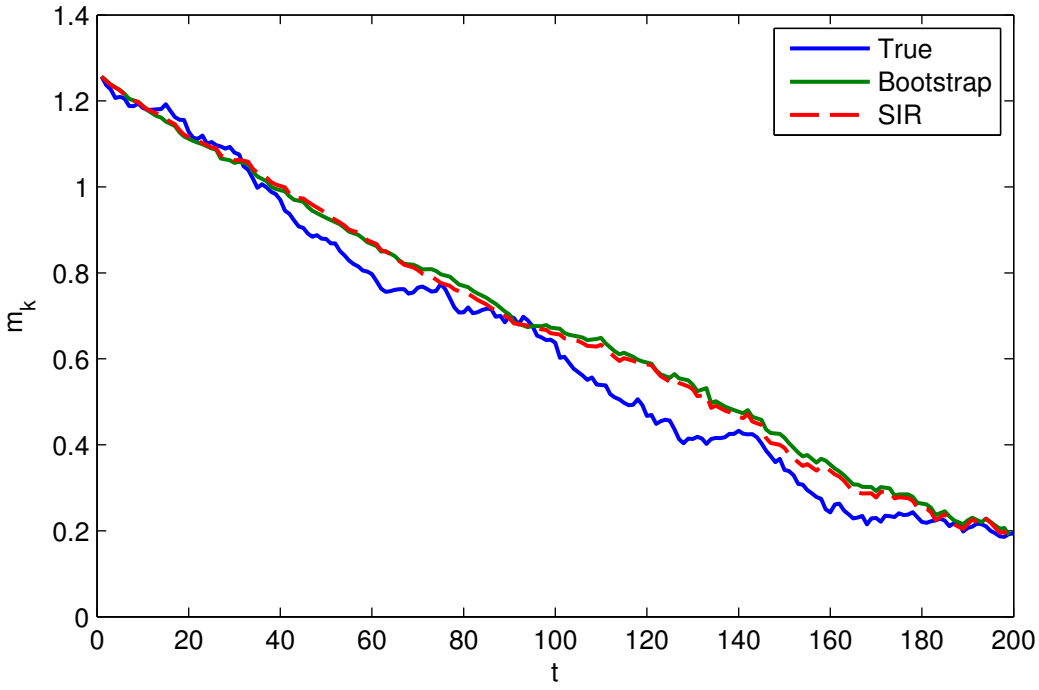


Figure 12: The true signal and the Bootstrap and SIR approximations in exercise 6.2

Listing 9: m-code in exercise 6.2

```
%% bootstrap

m = x0;
P = q;
ms = [m zeros(1,n-1)];
Ps = [P zeros(1,n-1)];
N = 500; % number of particles
p = m*ones(1,N); % particles
W = ones(1,N); % weights
W = W/sum(W);

for k=2:n
    % sample from the dynamic distribution, calculate weights
    %disp(W');
    for l=1:N
        p(l) = normrnd(f(p(l)),sqrt(q));
        W(l) = normpdf(y(k),h(p(l)),sqrt(r));
    end
    % normalize weights
    W = W/sum(W);
    % resample
    cdf = cumsum(W);
    po = p;
    for l=1:N
        p(l) = po(find(cdf > rand,1));
    end
    ms(k) = W*p';
end

R(6) = rmse(s(x),ms);
ms_bs = ms;

% SIR

m = x0;
P = q;
ms = [m zeros(1,n-1)];
Ps = [P zeros(1,n-1)];
N = 500; % number of particles
p = m*ones(1,N); % particles
W = ones(1,N); % weights
W = W/sum(W);

 % use UKF to get the importance distribution
alpha = 2; beta = 0; kappa = 0;
lambda = alpha^2*(1+kappa) - 1;
Wi = 1/(2*(1+lambda));
Wm = [lambda/(1+lambda) Wi Wi];
Wc = [lambda/(1+lambda)+(1-alpha^2) Wi Wi];
res = 0;
e = sqrt(lambda+1)*[0 1 -1]';
for k=2:n
    % prediction
    sig = m*ones(3,1)+sqrt(P)*e;
    % propagate through dynamic model
    sig = f(sig);
    m_ = Wm*sig;
    P_ = Wc*((sig-m_).^2)+q;

    sig_ = m_*ones(3,1)+sqrt(P_)*e;
    sigY = h(sig_);

    u = Wm*sigY;
    S = Wc*((sigY-u).^2)+r;
    C = Wc*((sig_-m_).*(sigY-u));
    K = C/S;
    m = m_+K*(y(k)-u);
```

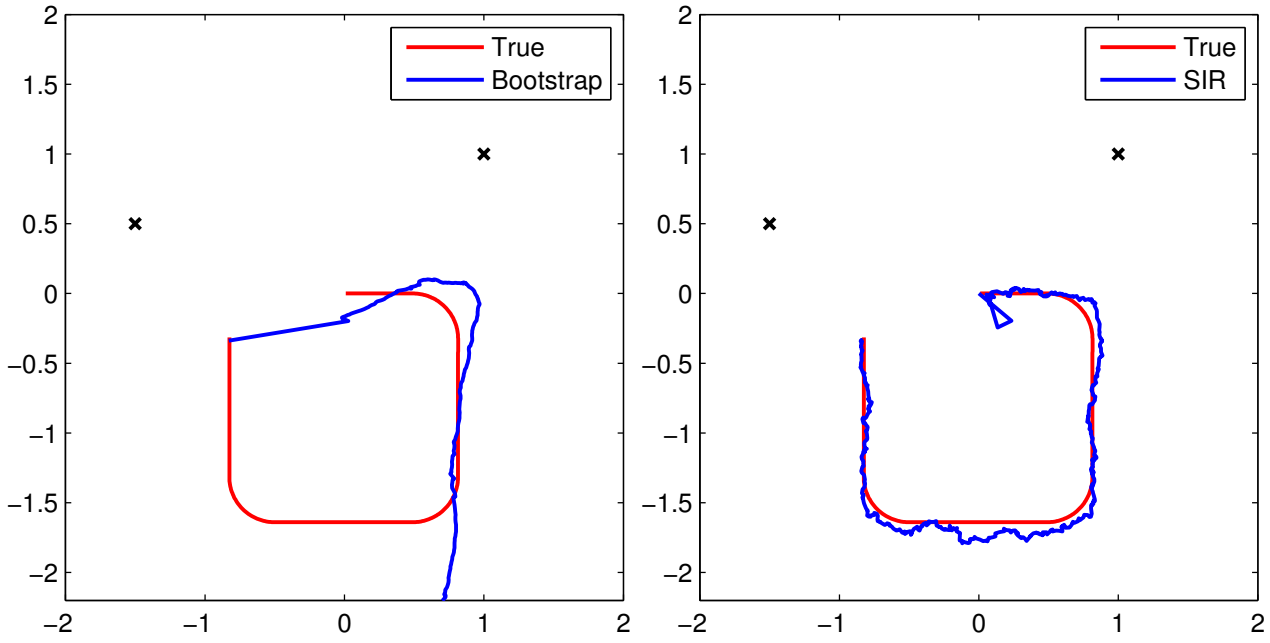
```
    P = P_-K^2*S;

    % draw samples from the importance distribution
    for l=1:N
        pp = normrnd(m,P); % x_k^l
        W(l) = W(l)*... % previous weight
               normpdf(y(k),h(pp),sqrt(r))*... % p(y_k|x_k^l)
               normpdf(pp,f(p(l)),sqrt(q))/... % p(x_k^l|x_k-1^l)
               normpdf(pp,m,sqrt(P)); % pi(x_k^l|x_k-1^l,y_1:k)
        p(l) = pp;
    end
    % normalize weights
    W = W/sum(W);
    % resample if needed
    Neff = 1/(W*W');
    if Neff < N / 10
        %fprintf('resampling, neff = %3.1f\n',Neff);
        po = p;
        cdf = cumsum(W);
        for l=1:N
            p(l) = po(find(cdf > rand,1));
        end
        W = ones(1,N); % weights
        W = W/sum(W);
    else
        %fprintf('not resampling, neff = %3.1f\n',Neff);
        res = res+1;
    end
    ms(k) = W*p';
    Ps(k) = (p-ms(k))*(W.*(p-ms(k)))';

end
```

Exercise 6.3

Here we implement the bootstrap filter and SIR with CKF importance distribution for the target tracking problem in exercise 4.3. The graphical results are presented in figure 13, where the UKF/CKF and EKF solutions are compared and the mcode in shown listing 10. The updated RMSE values are presented in table 9. As can be seen, the CKF/UKF approximation is better but only by a very slight margin.

Table 9: The RMSE values in exercise 6.3

	Baseline	**EKF**	**CKF**	**UKF**	**Bootstrap**	**SIR**
RMSE	1.02	0.44	0.44	0.44	1.42	0.44



(a) Bootstrap approximation

(b) SIR approximation

Figure 13: The true trajectory, the bootstrap approximation and the SIR approximation in exercise 6.3

Listing 10: m-code in exercise 6.3

```
function o=bootstrap()
    P = eye(4);        % Some uncertainty
    ms = [m zeros(4,steps-1)];
    ii = zeros(1,steps);
    N = 150;
    p = mvnrnd(repmat(x0',N,1),P)'; % particles
    W = ones(1,N); % weights
    W = W/sum(W);

    %p(:,1)
    %mvnrnd(A*p(:,1),Q)
    for k=2:steps
        % prediction and update
        for l=1:N
            p(:,l) = mvnrnd(A*p(:,l),Q)';
            W(l) = mvnpdf(Theta(:,k),h(p(:,l)),R);
        end
        W = W/sum(W);

        % resample
        cdf = cumsum(W);
        po = p;
        for l=1:N
            p(:,l) = po(:,find(cdf > rand,1));
        end
        ms(:,k) = p*W';
    end
    size(p)
    o = ms;
    %% Compute error
    fprintf('BOOTSTRAP %3.4f\n',rmse(X,ms));
    showtrace();
end

function o=sir()
    m = x0;
    P = eye(4);        % Some uncertainty
    ms = [m zeros(4,steps-1)];
    ii = zeros(1,steps);
    N = 150;
    p = repmat(x0,1,N); % particles
    W = ones(1,N); % weights
    W = W/sum(W);
    c = 0;
    for k=2:steps

        [m,P] = ckf_(Theta(:,k));
        % prediction
        for l=1:N
            pn = mvnrnd(m,P)';
            po = p(:,l);
            W(l) = W(l)*mvnpdf(Theta(:,k),h(pn),R)*mvnpdf(pn,A*po,Q)/mvnpdf(pn,m,P);
            if(~W(l))
                W(l) = 1/N;
                c=c+1;
            end
            p(:,l) = pn;
        end
        W_ = W/sum(W);
        if sum(isnan(W_)) > 0
            Theta(:,k)
            h(pn)
            mvnpdf(Theta(:,k),h(pn),R)
            pn
            A*po
            mvnpdf(pn,A*po,Q)
            mvnpdf(pn,m,P)
```

```
            W
            k
            break;
        end
        W = W_;
        % resample if needed
        Neff = 1/(W*W');
        if Neff < N / 10
            %fprintf('resampling, neff = %3.1f\n',Neff);
            po = p;
            cdf = cumsum(W);
            for l=1:N
                ran = rand;
                p(:,l) = po(:,find(cdf > ran,1));
            end
            W = ones(1,N); % weights
            W = W/sum(W);
        else
            %fprintf('not resampling, neff = %3.1f\n',Neff);
        end
        ms(:,k) = p*W';
        %anim();
    end
    c
    %% Compute error
    o = ms;
    fprintf('SIR %3.4f\n',rmse(X,ms));
    showtrace();
```

Round 7

Exercise 7.1

A)

Here we have implemented the RTS smoother for the random walk model in exercise 3.2. In this simple one dimensional case the RTS smoother equations are given by:

$$
\begin{aligned}
G_k &= \frac{P_k}{P_k + Q} \\
m_k^s &= m_k + G_k \left[m_{k+1}^s - m_k \right] \\
P_k^s &= P_k + G_k^2 \left[P_{k+1}^s - P_k - Q \right]
\end{aligned}
$$

The graphical results of the means are presented in figure 14 and a comparison of the evolution of the variances of the Kalman filter and the RTS smoother is presented in figure 15. As expected, the RTS smoother variance is smaller in general and equal to the Kalman filter variance when $k = n$. The m-code is shown in listing 11.

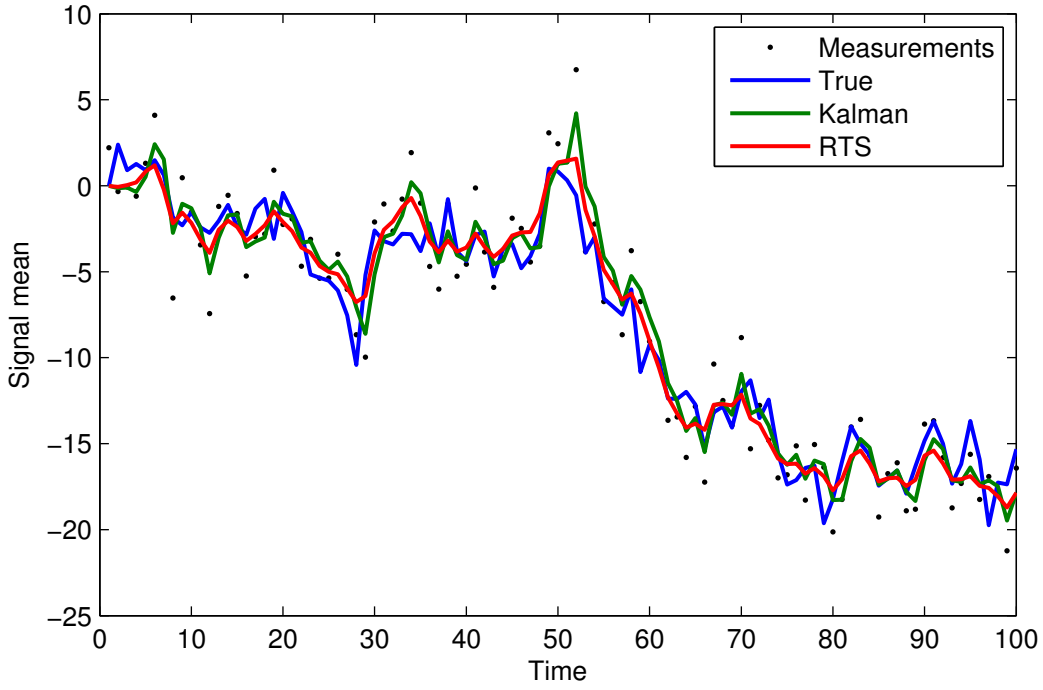


Figure 14: The measurements, the true signal, the Kalman filter solution mean and the RTS smoother solution mean in exercise 7.1A

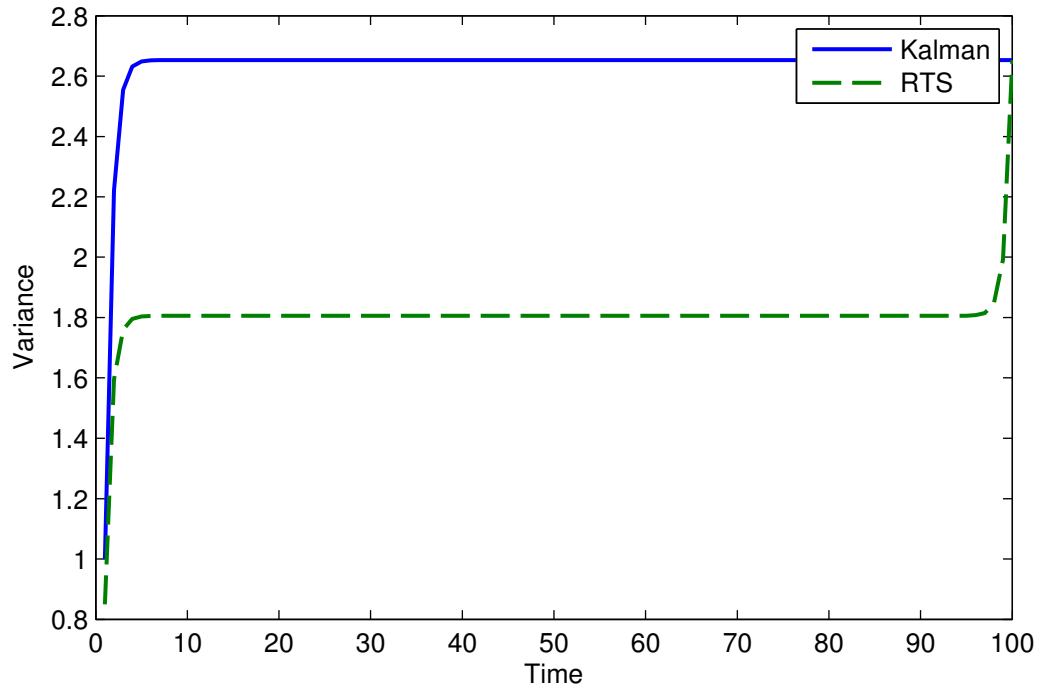


Figure 15: The Kalman filter solution's and the RTS smoother solution's variances in exercise 7.1A

Listing 11: m-code in exercise 7.1A

```
%% RTS smoother

mss = zeros(1,n);
Pss = zeros(1,n);
Gs = mss(1:n-1);
m = mso(end); % mso = Kalman filter means
P = Pso(end); % Pso = Kalman filter variances
mss(end) = m;
Pss(end) = P;

for k=n-1:-1:1
    g = Pso(k)/(Pso(k)+q);
    m = mso(k)+g*(m-mso(k));
    P = Pso(k)+g^2*(P-Pso(k)-q);
    Gs(k) = g;
    mss(k) = m;
    Pss(k) = P;
end
```

B)

Here the exact smoother of the previous exercise is computed by a grid approximation, in a similar fashion as for the Kalman filter in exercise 3.2. The means and variances of the smoother and its grid approximation are presented in figures 16 and 17. The m-code is shown in listing 12.

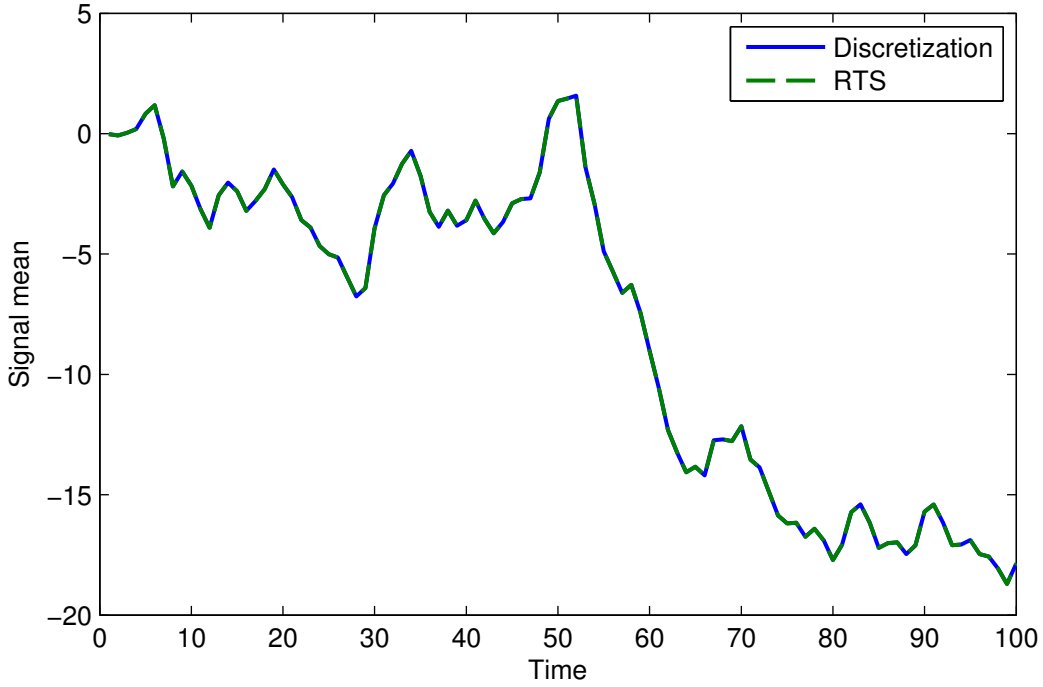


Figure 16: The RTS smoother's and its grid approximation's means in exercise 7.1B

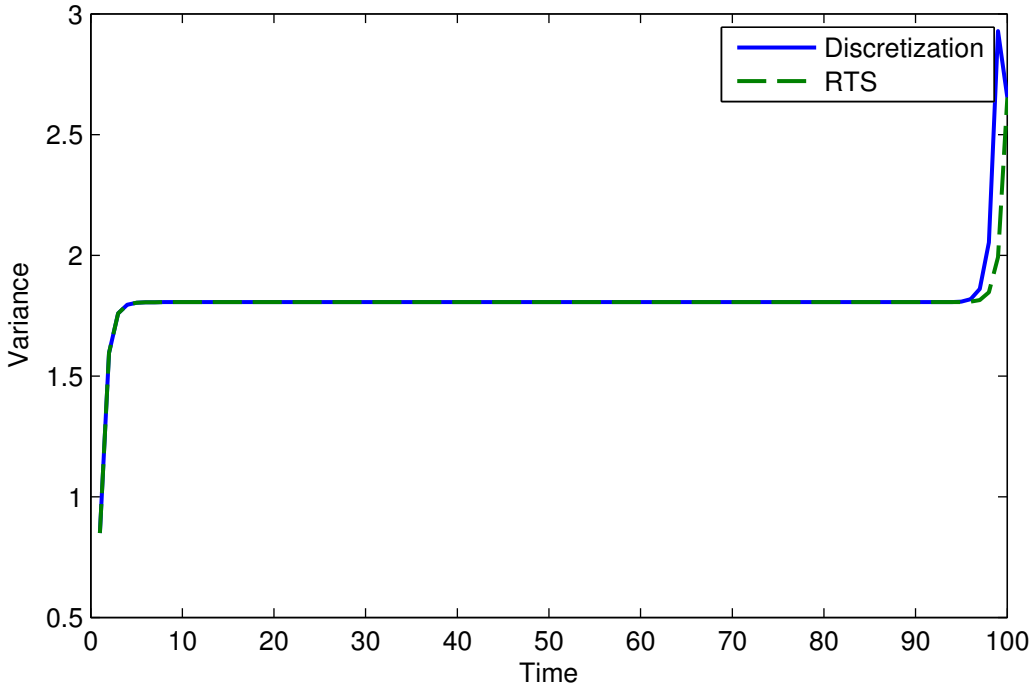


Figure 17: The RTS smoother's and its grid approximation's variances in exercise 7.1B

Listing 12: m-code in exercise 7.1B

```
% RTS smoother -- discrete

mss = zeros(1,n);
Pss = zeros(1,n);
m = mso(end);
P = Pso(end);
mss(end) = m;
Pss(end) = P;
p_ = normpdf(t,m,P);
size(p_)
distr = zeros(N,n);
for k=n-1:-1:1
    % dynamical distribution times the ”previous” smoothing distribution per
    % the predictive distribution (derived analytically in this linear
    % gaussian case)
    p = sum(p_dyn.*repmat(p_',1,N)./repmat(normpdf(t,mso(k),sqrt(Pso(k)+q))',1,N));
    % times the filtering distribution
    p = p.*normpdf(t,mso(k),sqrt(Pso(k)));
    p = p/sum(p);
    p_ = p;
    distr(:,k) = p';
    m = t*p';
    mss(k) = m;
    Pss(k) = (t-m).^2*p';
end
```

c)

If the stationary Kalman filter is used to provide the filtering distributions for the RTS smoother, the smoother is supposed to become a stationary backward filter. I have to admit I didn't quite get this. If an RTS smoother is based on the stationary Kalman filter, the resulting RTS smoother is different from a smoother where $G = \lim_{k\to\infty} G_k$. Here I choose to use the latter where $G_k = G$, whose value is taken as the last value from the smoother in 7.1A (convergence was checked graphically).

In figures 18 and 19 the means and variances of the stationary and non-stationary (the smoother in 7.1A) smoothers are compared and the m-code is shown in listing 13. The RMSE performance of the normal, discrete and stationary RTS smoothers is shown in table 10.

Table 10: The RMSE values in exercise 7.1C

	Kalman	**RTS**	**Stat. RTS**
RMSE	679.15	13.26	13.26

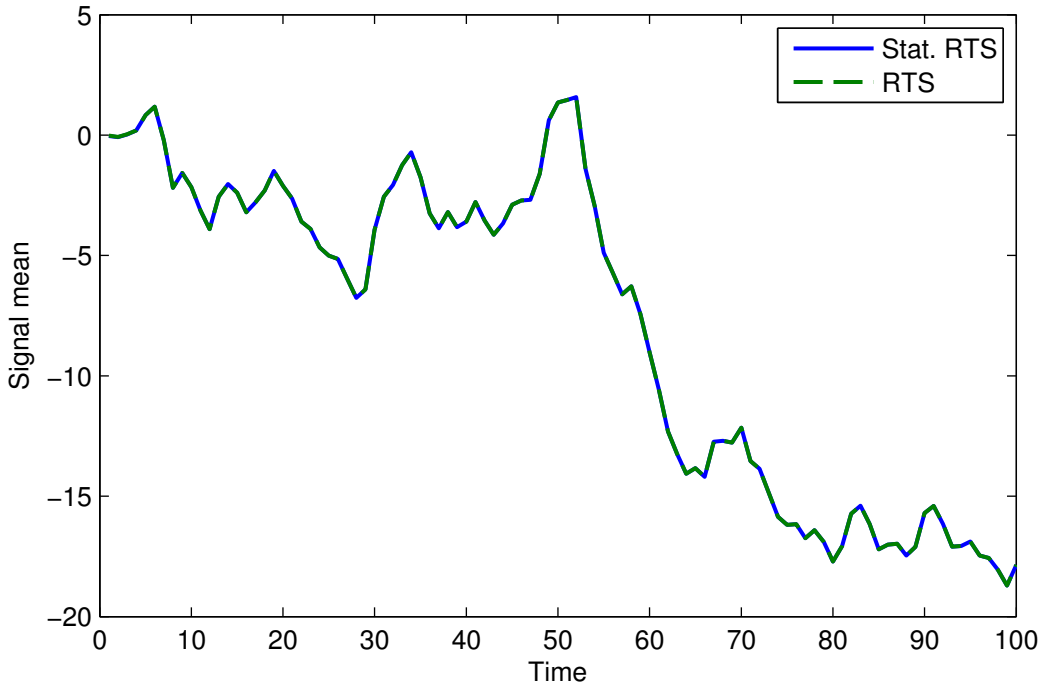


Figure 18: The RTS and the stationary RTS smoother's means in exercise 7.1C

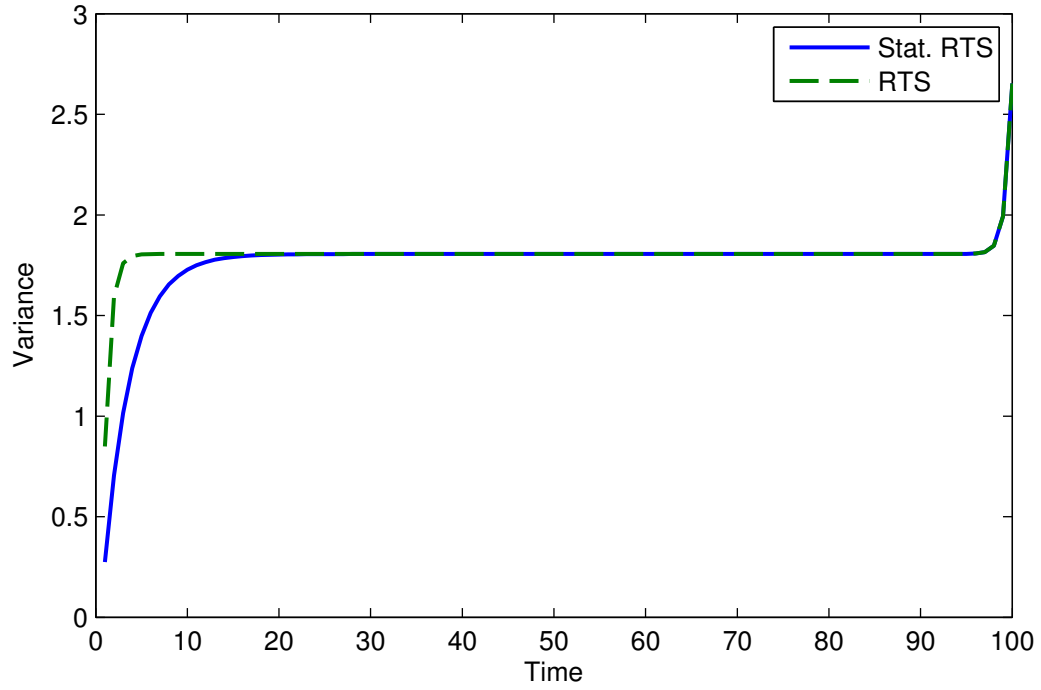


Figure 19: The RTS and the stationary RTS smoother's variances in exercise 7.1C

Listing 13: m-code in exercise 7.1c

```
% RTS smoother -- using the stationary Kalman filter

mss = zeros(1,n);
Pss = zeros(1,n);
G = Gs(end);
m = ms_st(end); % ms_st = stationary Kalman filter means
P = Ps_st(end); % Ps_st = stationary Kalman filter variances
mss(end) = m;
Pss(end) = P;
for k=n-1:-1:1
    m = ms_st(k)+G*(m-ms_st(k));
    P = Ps_st(k)+G^2*(P-Ps_st(k)-q);
    mss(k) = m;
    Pss(k) = P;
end
```

Exercise 7.2

Here we have implemented the RTS smoother to the resonator model in exercise 3.3. The performances of the base-line, the Kalman filter and the RTS smoother solutions are presented in table 11 and the graphical solutions are compared in figure 20. The m-code for the smoother is shown in listing 14.

Table 11: The RMSE values in exercise 7.2

	Baseline	**Kalman**	**Stat. Kalman**	**RTS**
RMSE	0.54	0.24	0.23	0.23

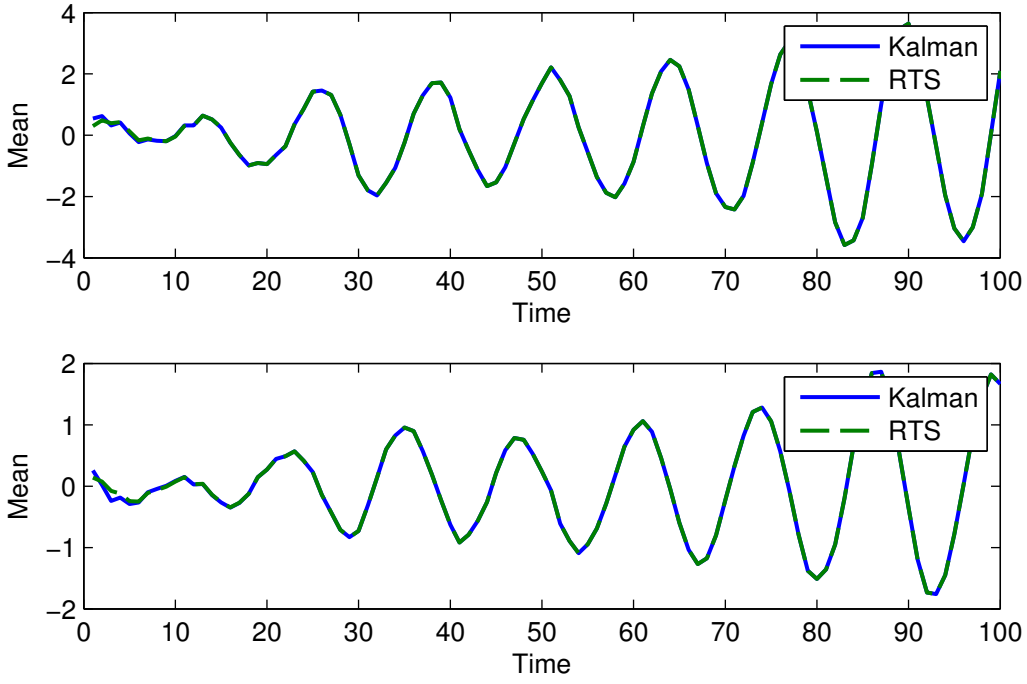


Figure 20: The Kalman filter's and RTS smoother's means in exercise 7.2

Listing 14: m-code in exercise 7.2

```
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %
  % RTS Smoother
  % The estimates of x_k are stored as columns of
  % the matrix EST3.
  %
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mss = zeros(2,steps);
Pss = EST2P;
Gs = mss;
m_s = EST2(:,end);
P_s = EST2P(:,:,end);
mss(:,end) = m_s;
Pss(:,:,end) = P_s;

for k=steps-1:-1:1
    m = EST2(:,k);
    P = EST2P(:,:,k);
    % prediction
    m_ = A*m;
    P_ = A*P*A'+Q;

    % update
    G = P*A/P_;
    m_s = m + G*(m_s-m_);
    P_s = P + G*(P_s-P_)*G';
    mss(:,k) = m_s;
    Pss(:,:,k) = P_s;

end
```

Exercise 7.3

A)

We proceed in the same fashion as in the derivation of the Kalman filter and the statistically linearized Kalman filter. The goal is to calculate the smoothing distribution $p(\mathbf{x}_k|\mathbf{y}_{1:T})$, where T is the number of all the observations, with a recursive formula. Let's assume that we know the Gaussian filtering distributions $p(\mathbf{x}_k|\mathbf{y}_{1:k}) = N(\mathbf{x}_k|\mathbf{m}_k, \mathbf{P}_k)$ and that we have a non-linear dynamical model with additive Gaussian noise: $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k) + \mathbf{q}_k$. Now taking into account the assumed Markov properties, we get

$$p(\mathbf{x}_k, \mathbf{x}_{k+1}|\mathbf{y}_{1:k}) = p(\mathbf{x}_{k+1}|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{y}_{1:k}).$$

To make this joint distribution Gaussian, we can apply the statistical linearization based Gaussian approximation, giving

$$p(\mathbf{x}_k, \mathbf{x}_{k+1}|\mathbf{y}_{1:k}) = N\left(\begin{bmatrix}\mathbf{x}_k \\ \mathbf{x}_{k+1}\end{bmatrix} \middle| \begin{bmatrix}\mathbf{m}_k \\ \mathbf{m}^-_{k+1}\end{bmatrix}, \text{diag}\left(\begin{bmatrix}\mathbf{m}_k \\ \mathbf{P}^-_{k+1}\end{bmatrix}\right)\right)$$

Taking the marginal distribution of $\mathbf{x}_{k+1}$ gives the prediction equations that I will present in the end. Taking again into account the conditional independence properties of the Markov chain we see that $p(\mathbf{x}_k|\mathbf{x}_{k+1}, \mathbf{y}_{1:T}) = p(\mathbf{x}_k|\mathbf{x}_{k+1}, \mathbf{y}_{1:k})$ which is given by the conditional distribution lemma of a joint normal distribution. Then

$$p(\mathbf{x}_k, \mathbf{x}_{k+1}|\mathbf{y}_{1:T}) = p(\mathbf{x}_k|\mathbf{x}_{k+1}, \mathbf{y}_{1:k})p(\mathbf{x}_{k+1}|\mathbf{y}_{1:T}),$$

where $p(\mathbf{x}_{k+1}|\mathbf{y}_{1:T})$ is the smoothing distribution of the previous step, is again joint normal, so that $p(\mathbf{x}_k|\mathbf{y}_{1:T})$ is given by the marginal distribution formula for the joint normal distribution. Putting all this together and using the matrix inversion lemma we get:

prediction:

$$\begin{aligned}\mathbf{m}_{k+1}^- &= \mathrm{E}\left[\mathbf{f}(\mathbf{x}_k)\right] \\ \mathbf{P}_{k+1}^- &= \mathrm{E}\left[\mathbf{f}(\mathbf{x}_k)\delta\mathbf{x}_k^T\right]\mathbf{P}_k^{-1}\mathrm{E}\left[\mathbf{f}(\mathbf{x}_k)\delta\mathbf{x}_k^T\right]^T + \mathbf{Q}_k\end{aligned}$$

update:

$$\begin{aligned}\mathbf{G}_k &= \mathrm{E}\left[\mathbf{f}(\mathbf{x}_k)\delta\mathbf{x}_k^T\right]^T\left[\mathbf{P}_{k+1}^-\right]^{-1} \\ \mathbf{m}_k^s &= \mathbf{m}_k + \mathbf{G}_k\left[\mathbf{m}_{k+1}^s - \mathbf{m}_{k+1}^-\right] \\ \mathbf{P}_k^s &= \mathbf{P}_k + \mathbf{G}_k\left[\mathbf{P}_{k+1}^s - \mathbf{P}_{k+1}^-\right]\end{aligned}$$

B)

Here we have implemented the statistically linearized RTS smoother (SLRTS) and the extended RTS smoother (ERTS) for the problem in exercise 4.1. The performance of the smoothers is illustrated in figure 21 and the RMSE's of all the implemented methods for this problem are compared in table 12. The m-code for the smoothers is shown in listing 15.

Table 12: The RMSE values in exercise 7.3B

	EKF	**SLF**	**UKF**	**GHKF**	**CKF**	**BS**	**SIR**	**ERTS**	**SLRTS**
RMSE	0.67	0.69	0.78	0.69	0.69	0.77	0.69	0.32	0.32

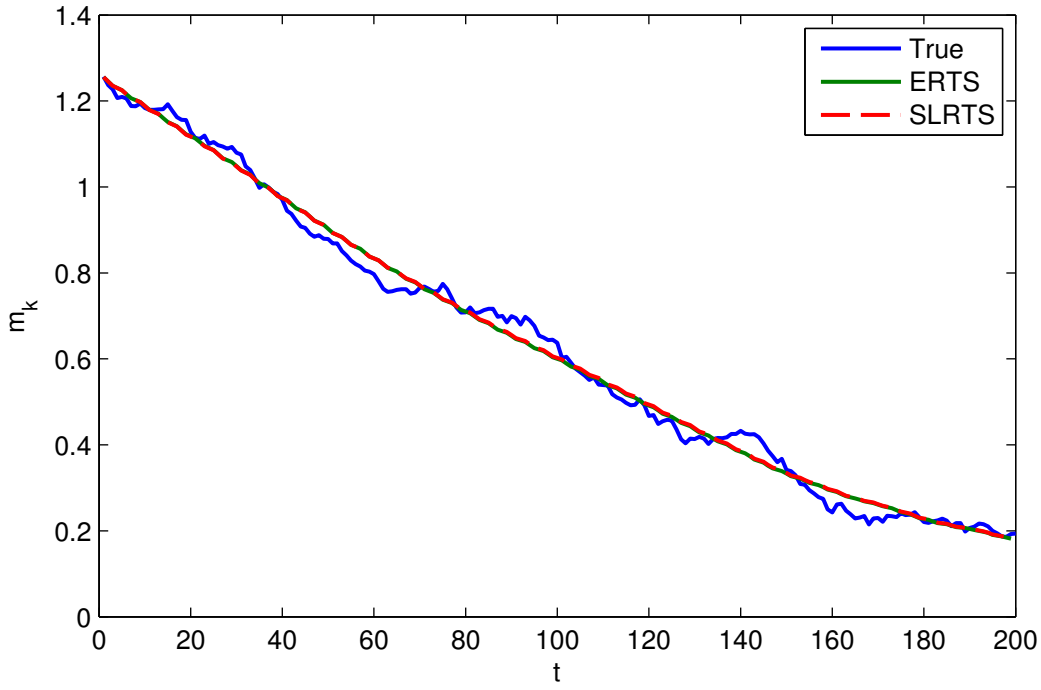


Figure 21: The RTS smoothers for the EKF (ERTS) and SLF (SLRTS) filters in exercise 7.3B

Listing 15: m-code in exercise 7.3B

```
%% ERTS

mss = zeros(1,n);
Pss = mss;
m_s = ms_ekf(:,end);
P_s = Ps_ekf(:,end);
mss(:,end) = m_s;
Pss(:,end) = P_s;

for k=n-1:-1:1
    m = ms_ekf(k);
    P = Ps_ekf(k);
    % prediction
    m_ = f(m);
    P_ = F(m)^2*P+q;

    % update
    G = P*F(m)/P_;
    m_s = m + G*(m_s-m_);
    P_s = P + G*(P_s-P_)*G';
    mss(k) = m_s;
    Pss(k) = P_s;

end

R(8) = rmse(s(x),mss);
ms_erts = mss;

% SL RTS

mss = zeros(1,n);
Pss = mss;
m_s = ms_slkf(:,end);
P_s = Ps_slkf(:,end);
mss(:,end) = m_s;
Pss(:,end) = P_s;

for k=n-1:-1:1
    m = ms_slkf(k);
    P = Ps_slkf(k);
    % prediction
    m_= Ef(m,P);
    P_ = Efdx(m,P)^2/P+q;

    % update
    G = Efdx(m,P)/P_;
    m_s = m + G*(m_s-m_);
    P_s = P + G*(P_s-P_)*G';
    mss(k) = m_s;
    Pss(k) = P_s;

end
```

c)

This is analogical to exercise 4.2C:

prediction:

$$\mathbf{m}_{k+1}^{-} = \mathrm{E}\left[\mathbf{f}(\mathbf{x}_k)\right]$$
$$\mathbf{P}_{k+1}^{-} = \mathrm{E}\left[\mathbf{F}_{\mathbf{x}}(\mathbf{x}_k)\right]\mathbf{P}_k^T\mathrm{E}\left[\mathbf{F}_{\mathbf{x}}(\mathbf{x}_k)\right]^T + \mathbf{Q}_k$$

update:

$$\mathbf{G}_k = \mathrm{E}\left[\mathbf{F}_{\mathbf{x}}(\mathbf{x}_k)\right]^T \left[\mathbf{P}_{k+1}^-\right]^{-1}$$
$$\mathbf{m}_k^s = \mathbf{m}_k + \mathbf{G}_k \left[\mathbf{m}_{k+1}^s - \mathbf{m}_{k+1}^-\right]$$
$$\mathbf{P}_k^s = \mathbf{P}_k + \mathbf{G}_k \left[\mathbf{P}_{k+1}^s - \mathbf{P}_{k+1}^-\right]$$