Ville Väänänen
63527M
ville.vaananen@aalto.fi

Exercise Report

S-114.4202 Special Course in Computational Engineering II

September 2, 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

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

C)

Assuming $w_{k-1} \sim N(0, \frac{q_c}{\Delta t})$ 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 $\mathbf{x}_k = \mathbf{x}(t_k)$ we get the following linear Gaussian state space model:

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

so that

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

D)

Solving the co

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 $\mathbf{P}_k$

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

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

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

$$\Leftrightarrow \mathbf{y} \sim N(\mathbf{Hm}, \mathbf{HPH}^T + \mathbf{R})$$

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

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

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 1 the random walk signal with $n = 100$ timesteps is plotted with the means given by the Kalman filter and the grid approximation. In figure 2 the variances of the approximated Gaussian distributions are plotted for the Kalman filter and the grid approximation. The m-code is presented in listing 1.

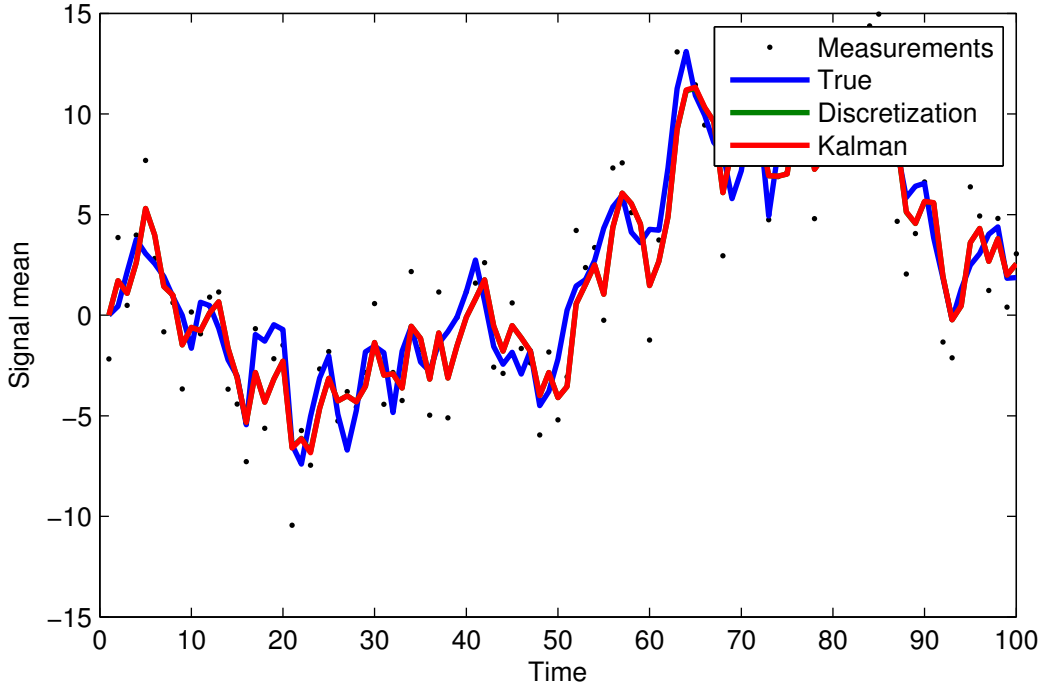


Figure 1: The true random walk signal (blue), the mean of the Kalman filter (red) and the mean of the grid approximation (green) in exercise 3.2

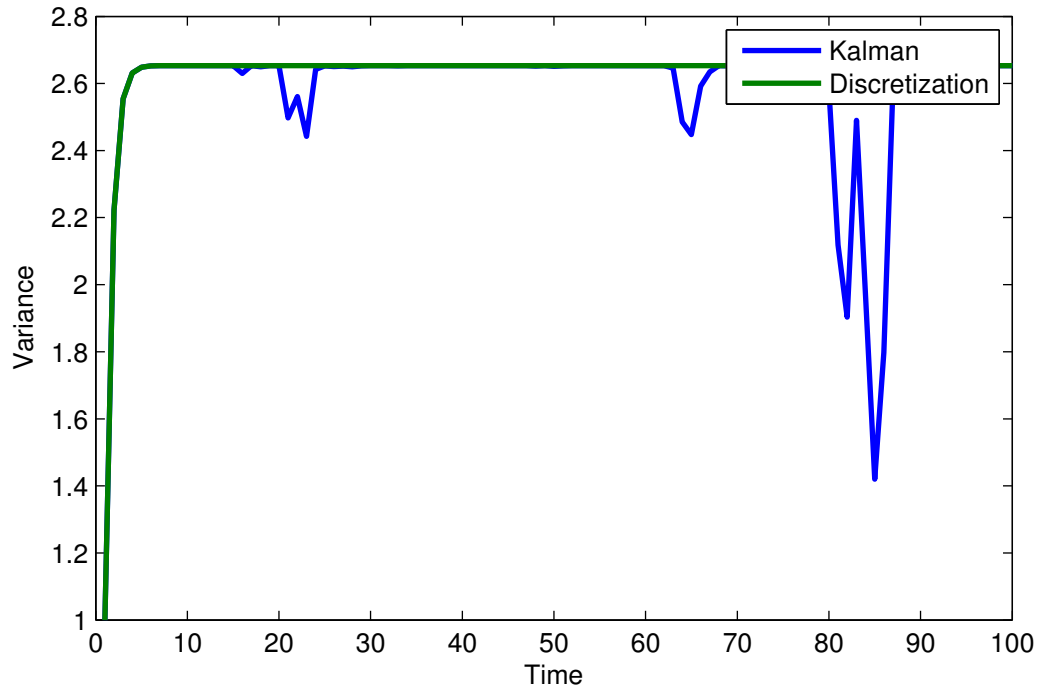


Figure 2: The true random walk signal (blue), the mean of the Kalman filter (red) and the mean of the grid approximation (green) in exercise 3.2

Listing 1: 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;
m2 = 0;
P2 = 1;
ms = zeros(1,n);ms(1)=m;
ms2 = zeros(1,n);
Ps2 = zeros(1,n);
Ps = zeros(1,n);Ps(1)=P;
Ks = Ps;
for k=2:n
    % prediction
    m_= m;
    P_ = P+q;
    % update
    K = P_/(P_+r);
    m = m_+K*(y(k)-m_);
    P = P_-(P_^2/(P_+r));
    ms(k) = m;
    Ps(k) = P;
    Ks(k) = K;
end

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

% discretization
a=min(y);
b=max(y);
N = 500;
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;

if 1 % plot
    % means, measurements
    plot(x,y,'.k',...
        1:n,s,...
        x,dms,...
        x,mso);
    legend('Measurements','True','Discretization','Kalman');
    xlabel('Time');
    ylabel('Signal mean');
    exportplot('ex_3_2_means.pdf',figW,figH);
```

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 3 and the Kalman filter solution in figure 4

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 1, from where it can be seen that the Kalman filter RMSE is 44% smaller. The relevant parts of the mcode for this exercise is shown in listing 2.

Listing 2: m-code in exercise 3.3A

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Kalman filter solution. The estimates
% of x_k are stored as columns of
% the matrix EST2.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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');
```

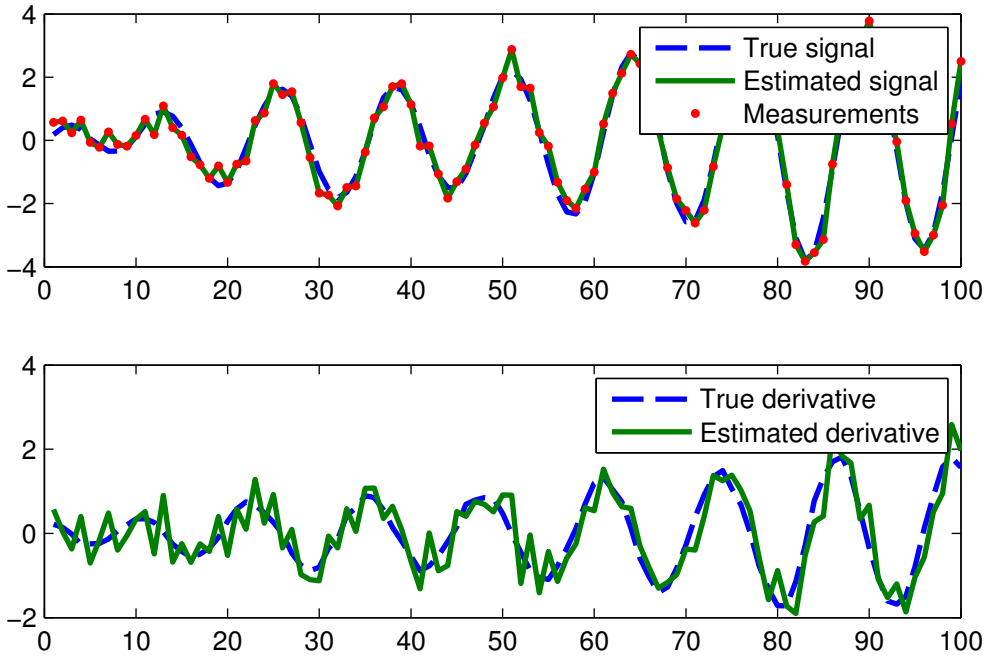


Figure 3: The base line solution in exercise 3.3

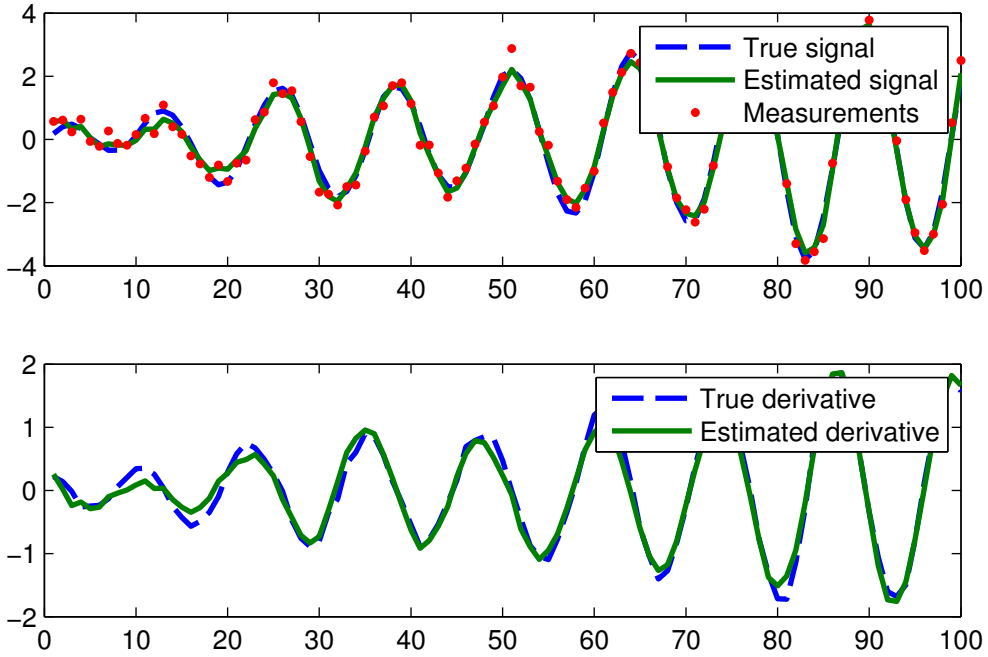


Figure 4: The Kalman filter solution in exercise 3.3

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

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 5 and the RMSE in table 1. 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.The relevant parts of the mcode for this exercise is shown in listing 3.

Listing 3: m-code in exercise 3.3B

```
  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

  % Plot the signal and its estimate
  figure;
  subplot(2,1,1);
  plot(T,X(1,:),'--',T,EST3(1,:),'-',T,Y,'.','MarkerSize',10);
  legend('True signal','Estimated signal','Measurements');

  % Plot the derivative and its estimate
  subplot(2,1,2);
  plot(T,X(2,:),'--',T,EST3(2,:),'-');
  legend('True derivative','Estimated derivative');
exportplot('ex_3_3_statkalmansol.pdf',figW,figH);
```

Table 1: The RMSE values in exercise 3.3

Baseline	Kalman	Stationary Kalman
0.536	0.237	0.234

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 6, 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 the following expectations:

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

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

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

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

The result of applying the SLF in the same situation as in 4.1A is presented also in figure 6. The RMSE's for both of the methods is presented in table 2 and the mcode is shown in listing 4.

Table 2: The RMSE values in exercise 4.1

EKF	SLF
1.83	0.84

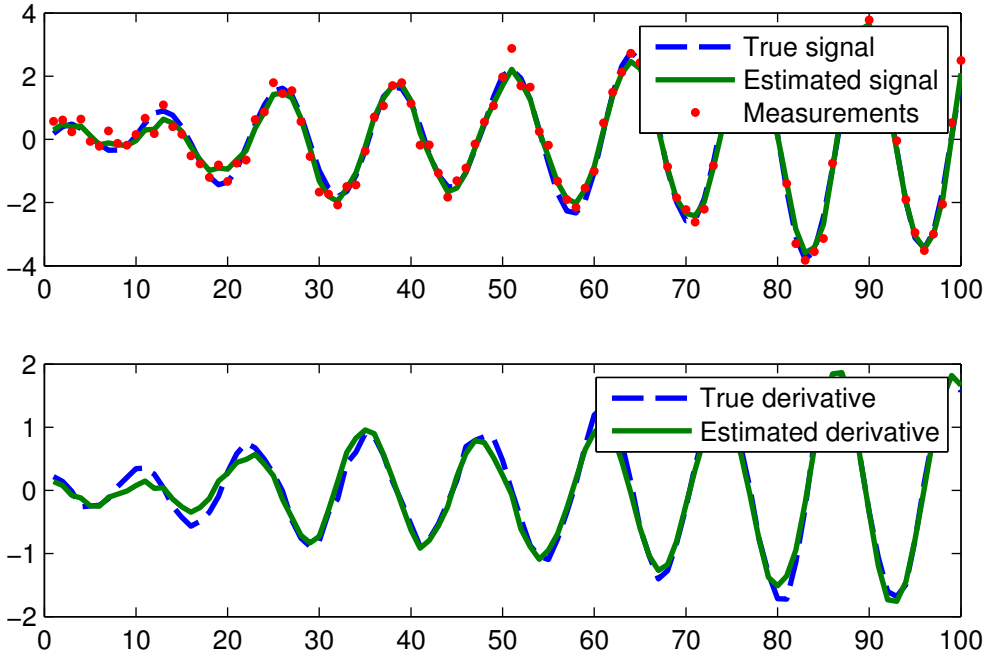


Figure 5: The stationary Kalman filter solution in exercise 3.3

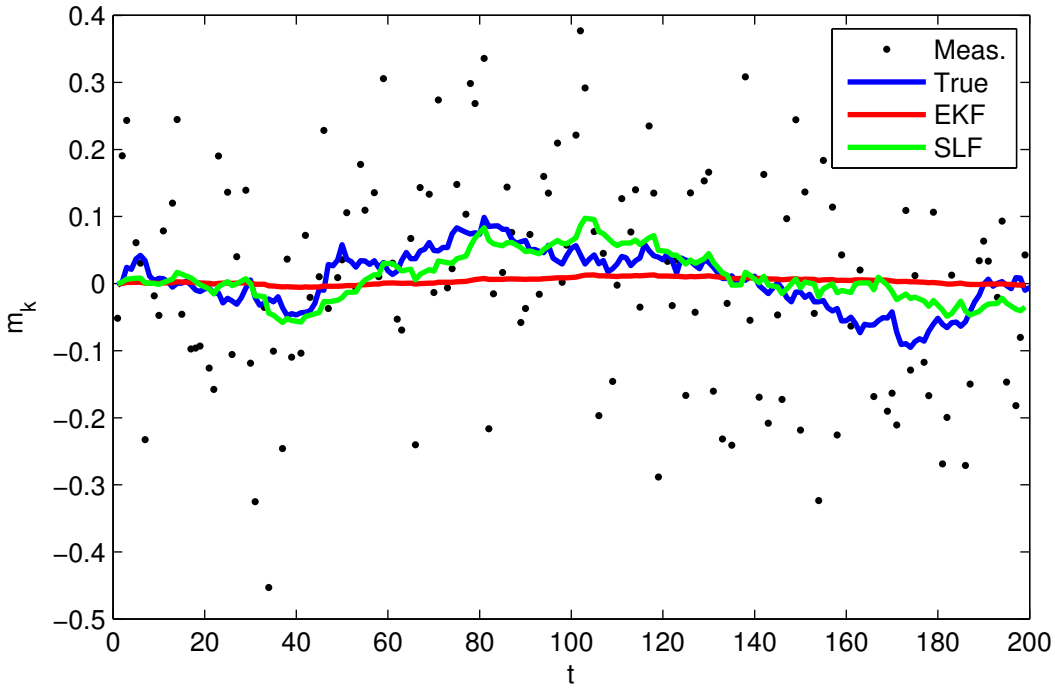


Figure 6: The true signal and the EKF and SLF approximations in exercise 4.1

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)-0.01*cos(x);
h=@(x)0.5*sin(2*x);
H=@(x)cos(2*x);
Y=s;
%x0 = 0.8*pi;
x0 = 0;
x = x0;
for k=1: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);
plot(x,y,'.k',1:N,s,'MarkerSize',8);
hold on;

%% EKF

m = x0;
P = 1;
ms = zeros(1,n);
Ps = zeros(1,n);
for k=1: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;
plot(x,ms,'-r');
fprintf('EKF %3.4f\n',rmse(s(x),ms));
%% SLF
es=@(m,p)m-...
        (m^3+3*m*p^2)/6+...
        (m^5+10*m^3*p^2+15*m*p^4)/120-...
        (m^7+21*m^5*p^2+105*m^3*p^4+105*m*p^6)/(7*6*5*4*3*2);

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 = zeros(1,n);
Ps = zeros(1,n);
for k=1:n
```

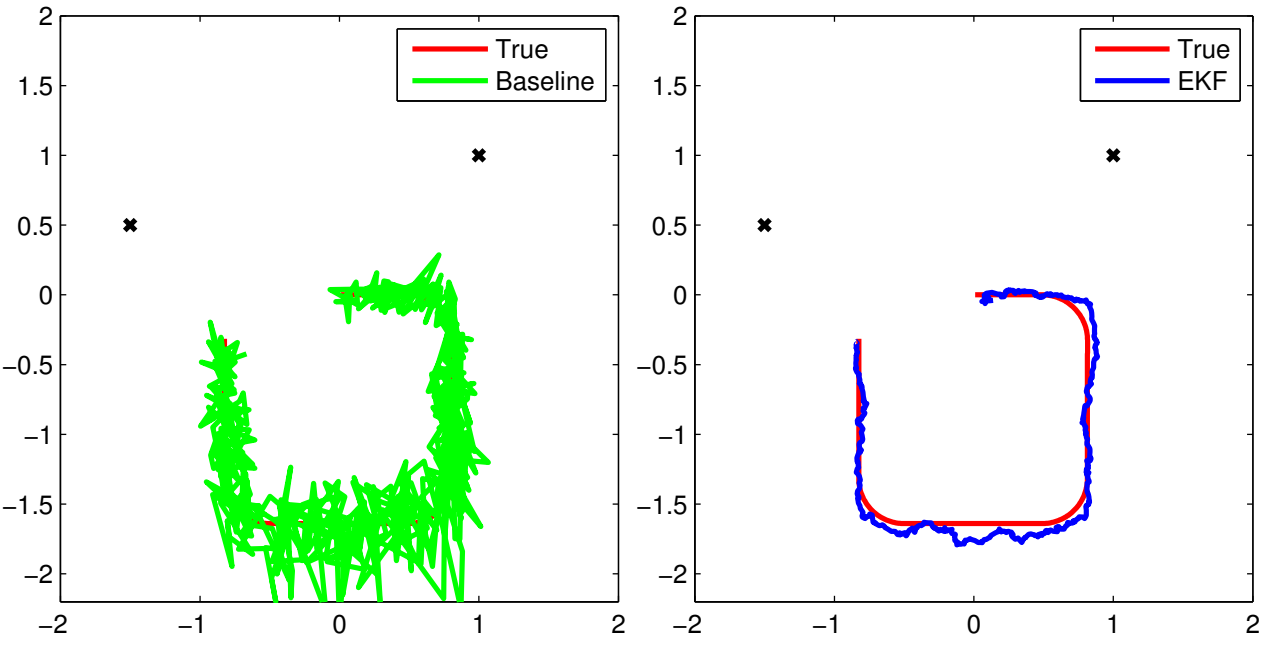
```
% prediction
m_= Ef(m,P);
P_ = Efdx(m,P)^2/P+q;
% update
v = y(k)-Eh(m_,P_);
S = Ehdx(m_,P_)^2/P_+r;
K = Ehdx(m_,P_)/S;
m = m_+K*v;
P = P_-K^2*S;
ms(k) = m;
Ps(k) = P;
```

Exercise 4.2

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 7a and 7b, the RMSE value in table 3 and the mcode in listing 5.



(a) Baseline approximation (b) EKF approximation

Figure 7: The true trajectory, the baseline approximation and the EKF approximation in exercise 4.3A

Table 3: The RMSE values in exercise 4.3A

Baseline	EKF
1.019	0.441

Listing 5: m-code in exercise 4.3A

```
            dy1 = sin(Theta(1,k));
            dx2 = cos(Theta(2,k));
            dy2 = sin(Theta(2,k));
            d = [dx1 dx2; dy1 dy2]\[S(1,2)-S(1,1);S(2,2)-S(2,1)];

            cross_xy = S(:,1) + [dx1;dy1]*d(1);

            %% Compute estimate
            m(3:4) = [0;0];
            m(1:2) = cross_xy;
            ms(:,k) = m;

            %anim();

        end

        fprintf('BL %3.4f\n',rmse(X,ms));
        h= showtrace('g');
        %disp(h);
        o = ms;
    end

    %%%%%%%%%%%%%%%%%%%
    %%%% EKF %%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%
    function [o,hh]=ekf()
        m = x0;            % Initialize to true value
        P = eye(4);        % Some uncertainty
        ms = zeros(4,steps);
        for k=1: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';
```

Round 5

Exercise 5.1

Here we implement the unscented Kalman filter for the problem in exercise 4.1 The graphical results are presented in figure 8, where the UKF solution is compared with the EKF solution, and the mcode in shown listing 6. With the chosen parameter values the UKF and SLF solutions become identical and so the RMSE's are also identical.

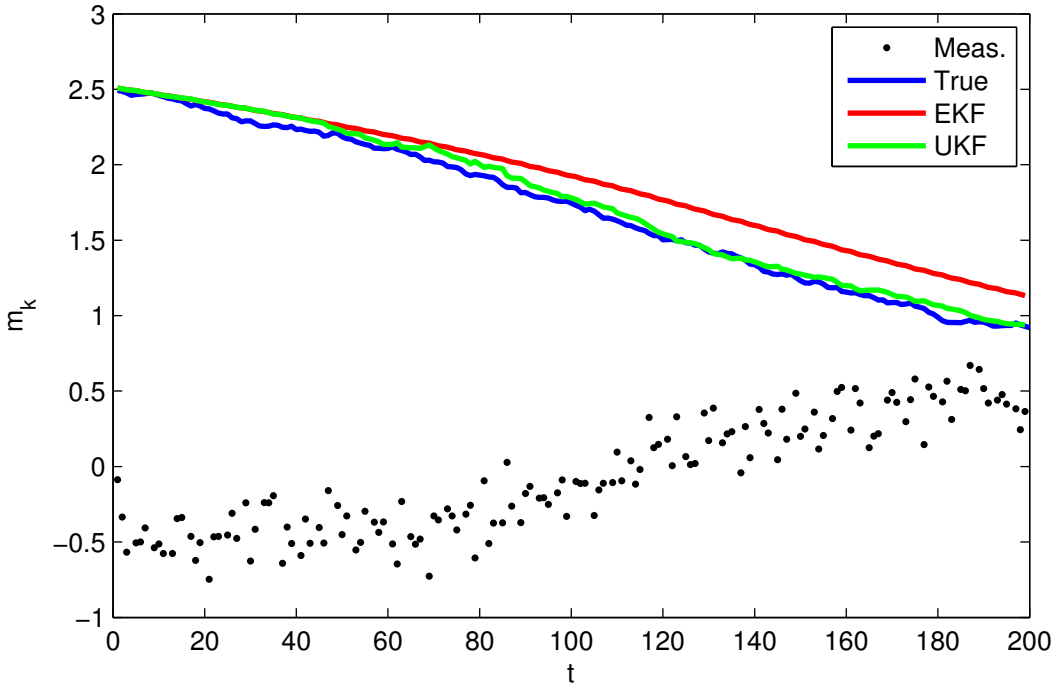


Figure 8: The true signal and the EKF and UKF approximations in exercise 5.1

Listing 6: m-code in exercise 5.1

```
%% UKF

m = x0;
P = q;
ms = zeros(1,n);
Ps = zeros(1,n);
sig = zeros(0,3);
alpha = 1; 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];
for k=1:n
    % prediction
    % form the three sigma points in the original space
    sig = m*ones(3,1)+sqrt(lambda+1)*sqrt(P)*[0 1 -1]';
    sig = f(sig);
    m_ = Wm*sig;
    P_ = Wc*((sig-m_).^2)+q;

    sig_ = m_*ones(3,1)+sqrt(lambda+1)*sqrt(P_)*[0 1 -1]';
    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 identical no matter what the order of the polynomial chosen for GHKF. The solutions are also identical to the UKF (with the chosen parameters) and SLF solutions. The mcode is presented in listing 7.

Listing 7: m-code in exercise 5.2

```
%% GHKF

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

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

for k=1: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_);
```

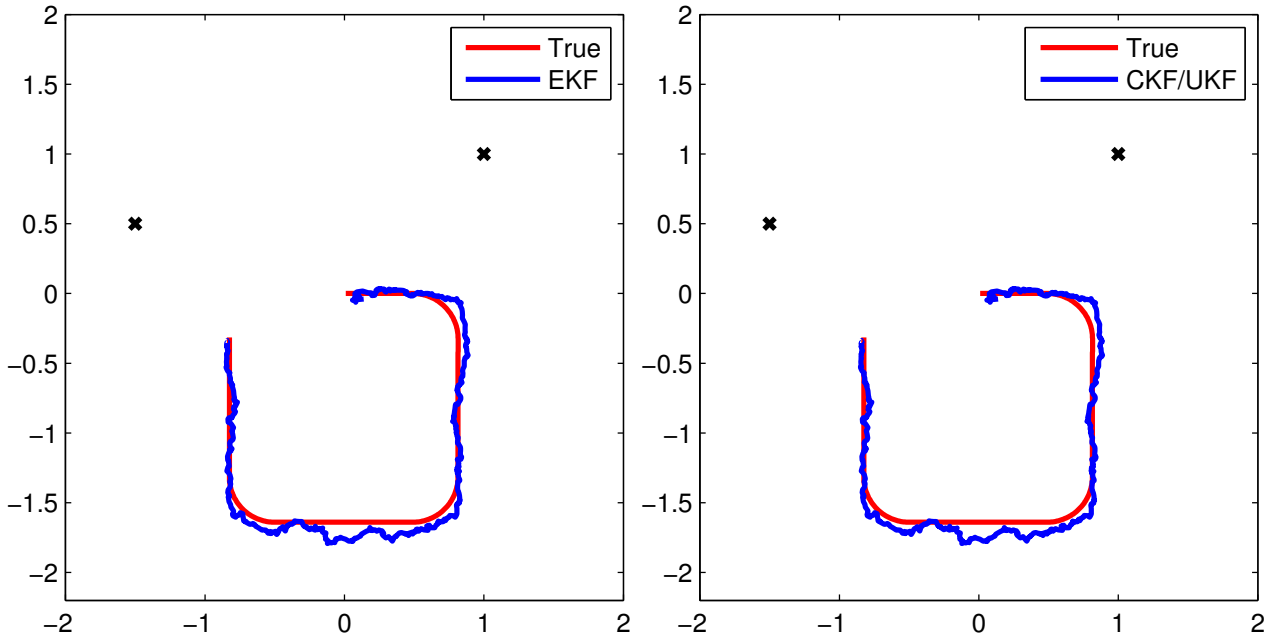
```
    u = W*sigY;
    S = W*((sigY-u).^2)+r;
    C = W*((sig_-m_).*(sigY-u));
    K = C/S;
    m = m_+K*(y(k)-u);
    P = P_-K^2*S;
    ms(k) = m;
    Ps(k) = P;
end

plot(x,ms,'-m');
fprintf('GHKF %3.4f\n',rmse(s(x),ms));

%% CKF

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

p = 2; % order of the polynomial
e = [1;-1]; % unit sigma points
W = ones(1,2);
W = W/sum(W);

for k=1: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
```

Exercise 5.3

Here we implement the UKF and the CKF for the target tracking problem in exercise 4.3. Again the parameters for UKF were chosen so that these two methods became identical The graphical results are presented in figure 9, where the UKF/CKF and EKF solutions are compared and the mcode in shown listing 8. The updated RMSE values are presented in table 4. As can be seen, the CKF/UKF approximation is better but only by a very slight margin.

Table 4: The RMSE values in exercise 5.3

Baseline	EKF	CKF/UKF
1.019	0.441	0.438



(a) EKF approximation

(b) CKF/UKF approximation

Figure 9: The true trajectory, the EKF approximation and the CKF/UKF approximation in exercise 5.3

Listing 8: m-code in exercise 5.3

```
function ms = ckf()

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

    for k=1:steps
        [m,P] = ckf_(Theta(:,k));
        ms(:,k) = m;
        Ps(:,:,k) = P;
    end

      %% Compute error
    fprintf('CKF %3.4f\n',rmse(X,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 ms=ukf()
    m = x0;            % Initialize to true value
    P = eye(4);        % Some uncertainty
    ms = zeros(4,steps);
    n = 4;
    nn = size(S,2);
    alpha = 1; kappa = 0;
    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) 1/(2*(n+lambda))*ones(1,2*n)];
    for k=1:steps
        % prediction
        sig = repmat(m,1,2*n+1)+chol(P,'lower')*e;
        sig = A*sig;
        m_ = sig*Wm';
        m__ = repmat(m_,1,2*n+1);
        P_ = (sig-m__)*diag(Wc)*(sig-m__)'+Q;
        sig_ = m__+chol(P_,'lower')*e;
        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

        fprintf('UKF %3.4f\n',rmse(X,ms));
        showtrace();
    end
```

Round 6

Exercise 6.1

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 10 and the mcode in shown listing 9. The RMSE values are presented in table 5. As can be seen from the figure and the RMSE table, the SIR filter is somewhat better as was expected.

Table 5: The RMSE values in exercise 6.2

Bootstrap	SIR
0.388	0.355

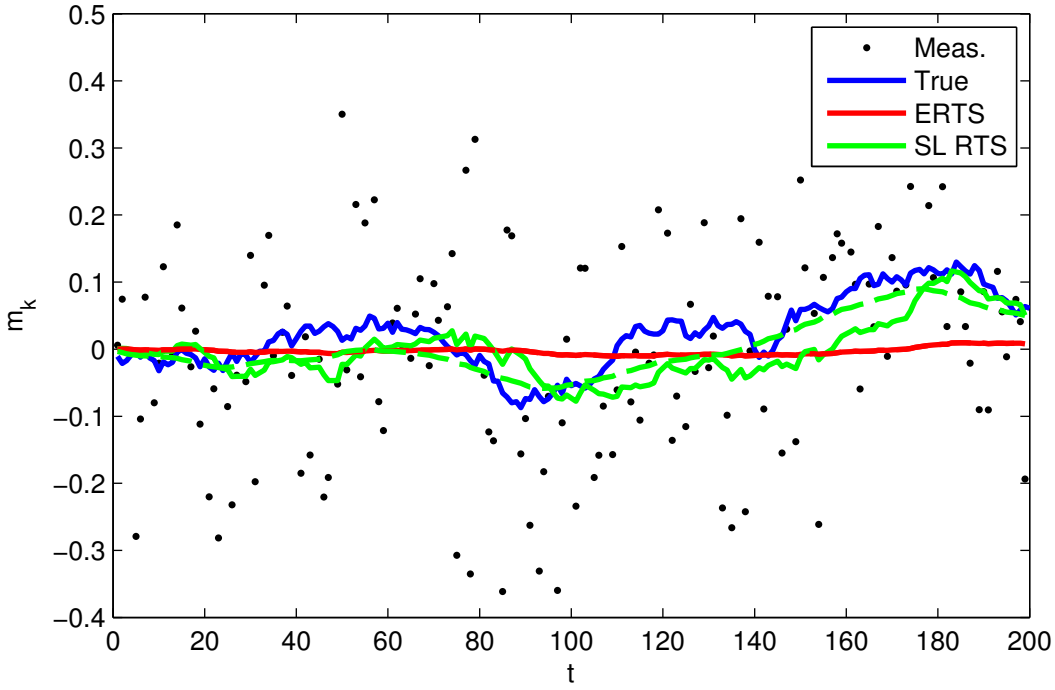


Figure 10: The true signal and the Bootstrap and SIR approximations in exercise 6.2

Listing 9: m-code in exercise 6.2

```
%% bootstrap

m = x0;
P = q;
ms = zeros(1,n);
Ps = zeros(1,n);
N = 700; % number of particles
p = m*ones(1,N); % particles
W = ones(1,N); % weights
W = W/sum(W);

for k=1: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);
    cdf = cumsum(W);
    for l=1:N
        ran = rand;
        p(l) = p(find(cdf > ran,1));
    end
    ms(k) = W*p';
end

plot(x,ms,'-g');
fprintf('BOOTSTRAP %3.4f\n',rmse(s(x),ms));

%% SIR

m = x0;
P = q;
ms = zeros(1,n);
Ps = zeros(1,n);
N = 700; % number of particles
p = m*ones(1,N); % particles
W = ones(1,N); % weights
W = W/sum(W);

sig = zeros(0,3);
alpha = 1; 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];

for k=1:n
    % use UKF to get proposal mean and covariance
    sig = m*ones(3,1)+sqrt(lambda+1)*sqrt(P)*[0 1 -1]';
    sig = f(sig);
    m_ = Wm*sig;
    P_ = Wc*((sig-m_).^2)+q;

    sig_ = m_*ones(3,1)+sqrt(lambda+1)*sqrt(P_)*[0 1 -1]';
    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;

    % sample from the proposal distribution
```

```
    for l=1:N
        pp = normrnd(m,P);
        W(l) = W(l)*normpdf(y(k),h(pp),sqrt(r))*normpdf(pp,f(p(l)),sqrt(q))/normpdf(pp,m
             ,P);
        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
            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) = W*p';

end
```

Exercise 6.3

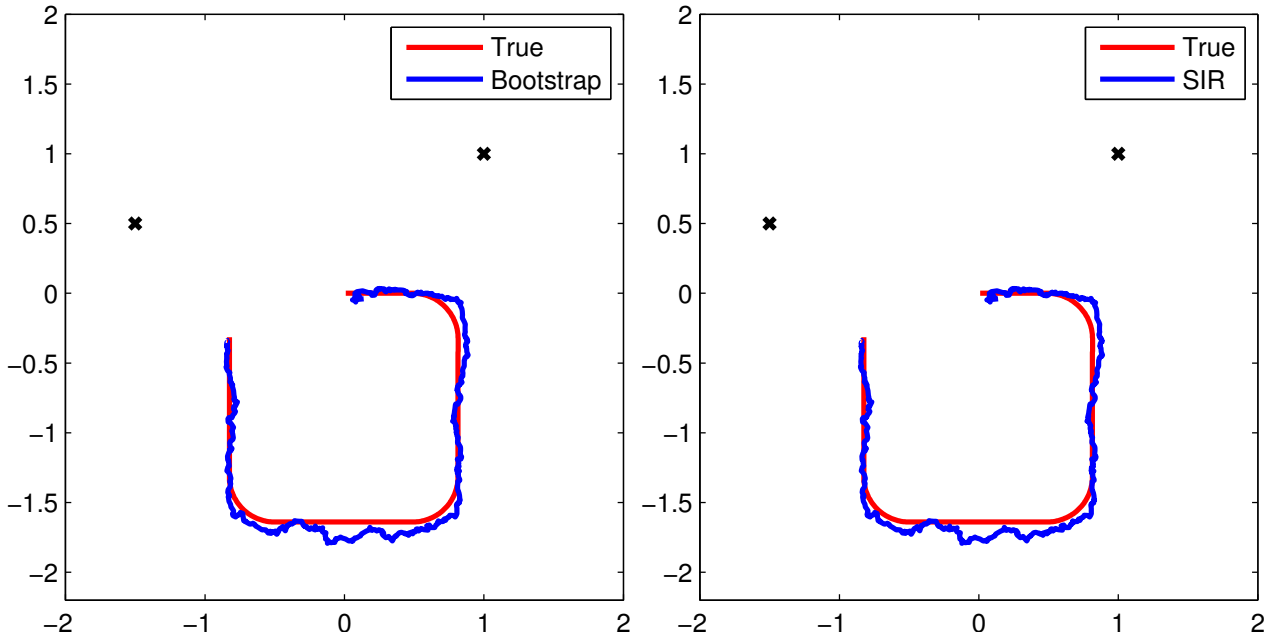
Here we implement the bootstrap filter and SIR with CKF importance distribution for the target tracking problem in exercise 4.3. The graphical results are presented in figure 11, where the UKF/CKF and EKF solutions are compared and the mcode in shown listing 10. The updated RMSE values are presented in table 6. As can be seen, the CKF/UKF approximation is better but only by a very slight margin.

Table 6: The RMSE values in exercise 6.3

Bootstrap	SIR
0.946	0.440

Listing 10: m-code in exercise 6.3

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

        for k=1:steps
            % prediction
            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);
            cdf = cumsum(W);
            po = p;
            for l=1:N
```



(a) Bootstrap approximation

(b) SIR approximation

Figure 11: The true trajectory, the bootstrap approximation and the SIR approximation in exercise 6.3

```
            ran = rand;
            ii(k) = find(cdf ≥ ran,1);
            p(:,l) = po(:,find(cdf ≥ ran,1));
        end
        ms(:,k) = p*W';
        %anim();
    end

    %% Compute error
    fprintf('BOOTSTRAP %3.4f\n',rmse(X,ms));
    showtrace();
end

function ms=sir()
    m = x0;
    P = eye(4);        % Some uncertainty
    ms = zeros(4,steps);
    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=1: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
    fprintf('SIR %3.4f\n',rmse(X,ms));
    showtrace();
end
```

Round 7

Exercise 7.1

A)

B)

C)

Exercise 7.2

Exercise 7.3

A)

B)

C)

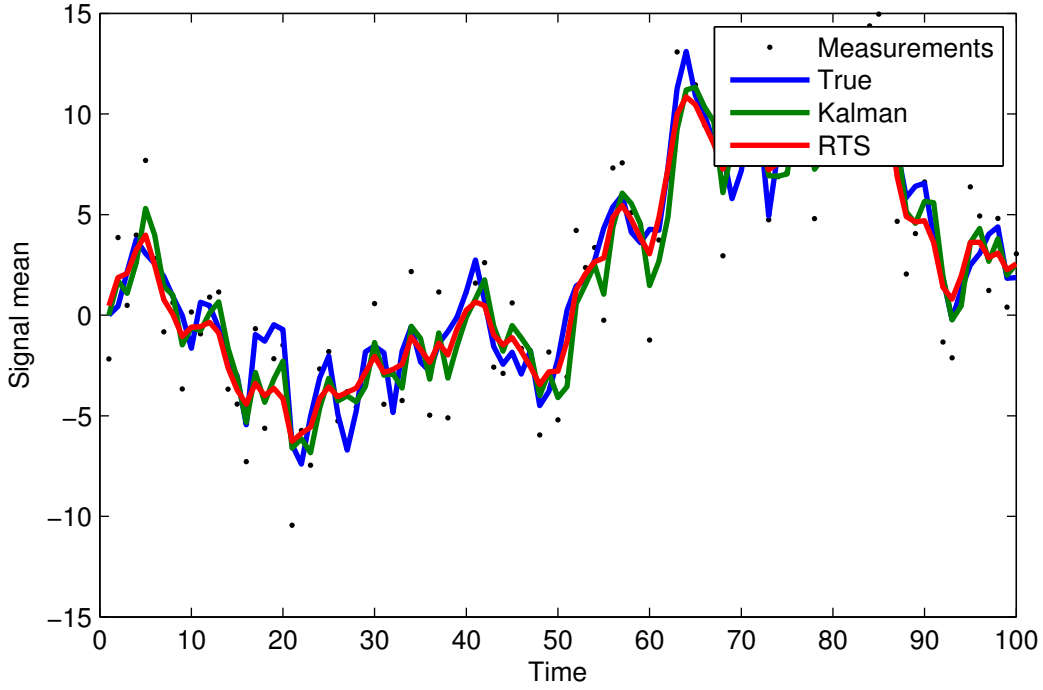


Figure 12: The measurements, the true signal, the Kalman filter solution mean and the RTS smoother solution mean in exercise 7.1A

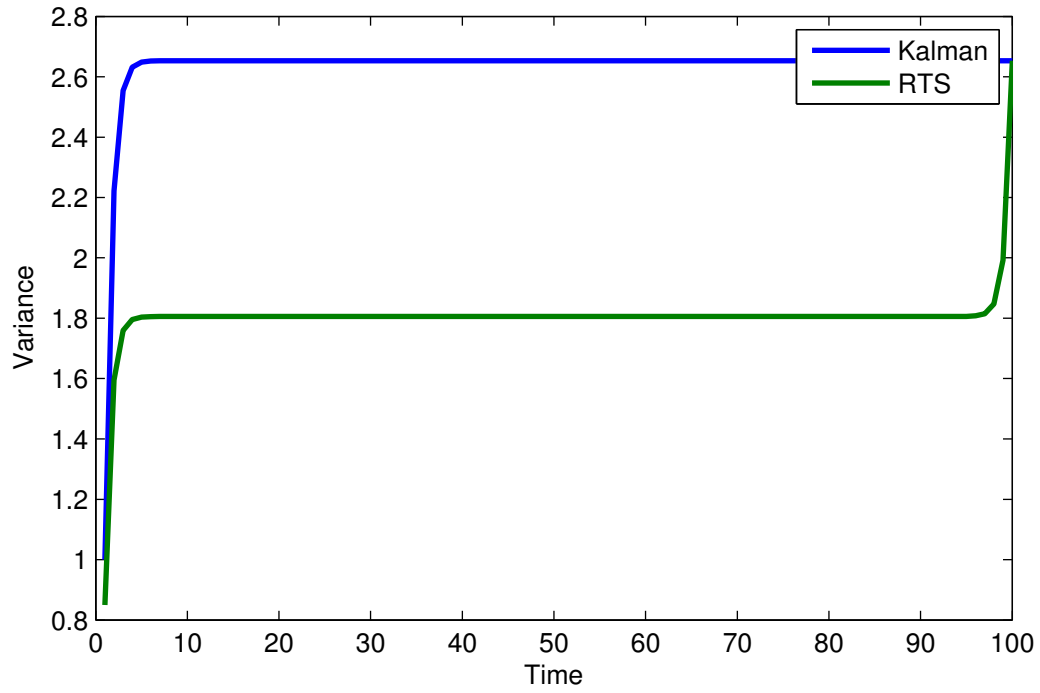


Figure 13: The Kalman filter solution's and the RTS smoother solution's variances in exercise 7.1A

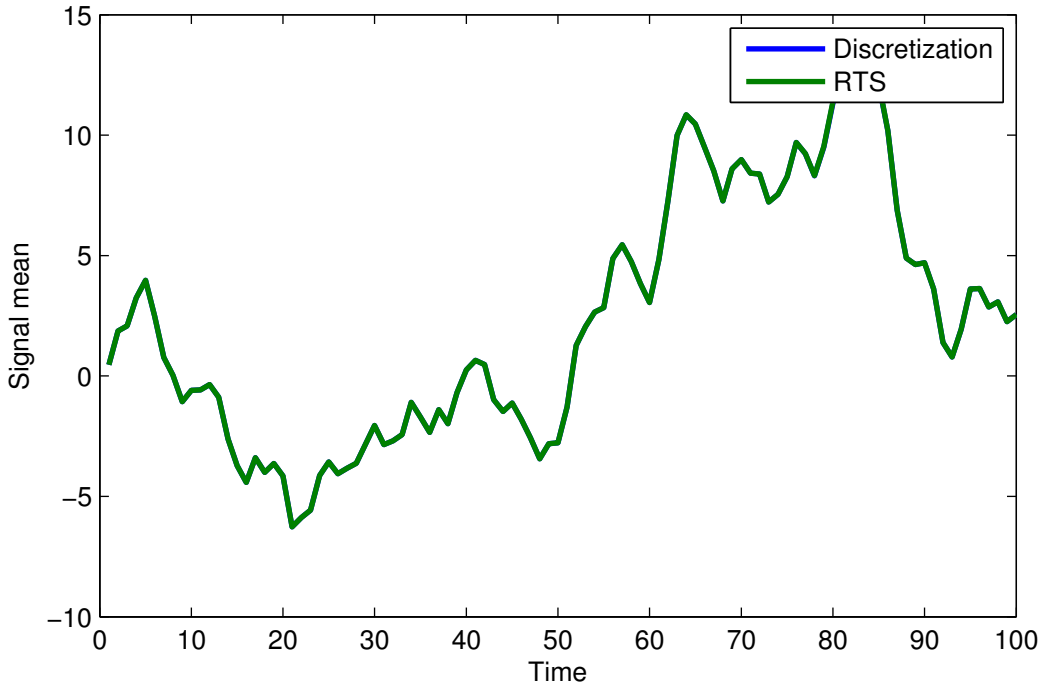


Figure 14: The RTS smoother's and its grid approximation's means in exercise 7.1B

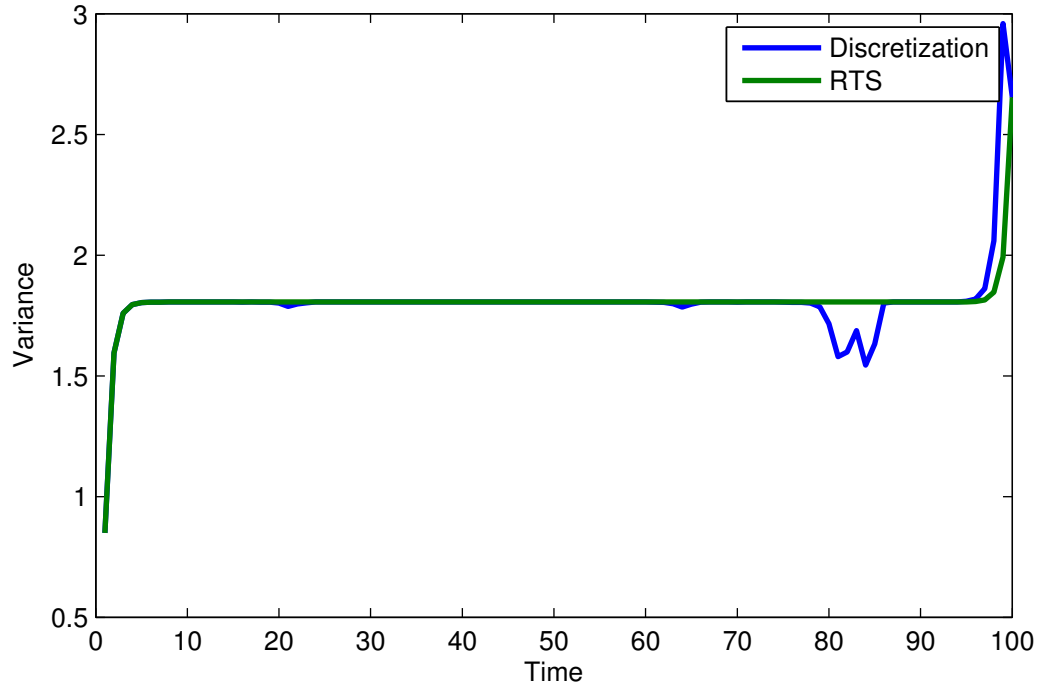


Figure 15: The RTS smoother's and its grid approximation's variances in exercise 7.1B

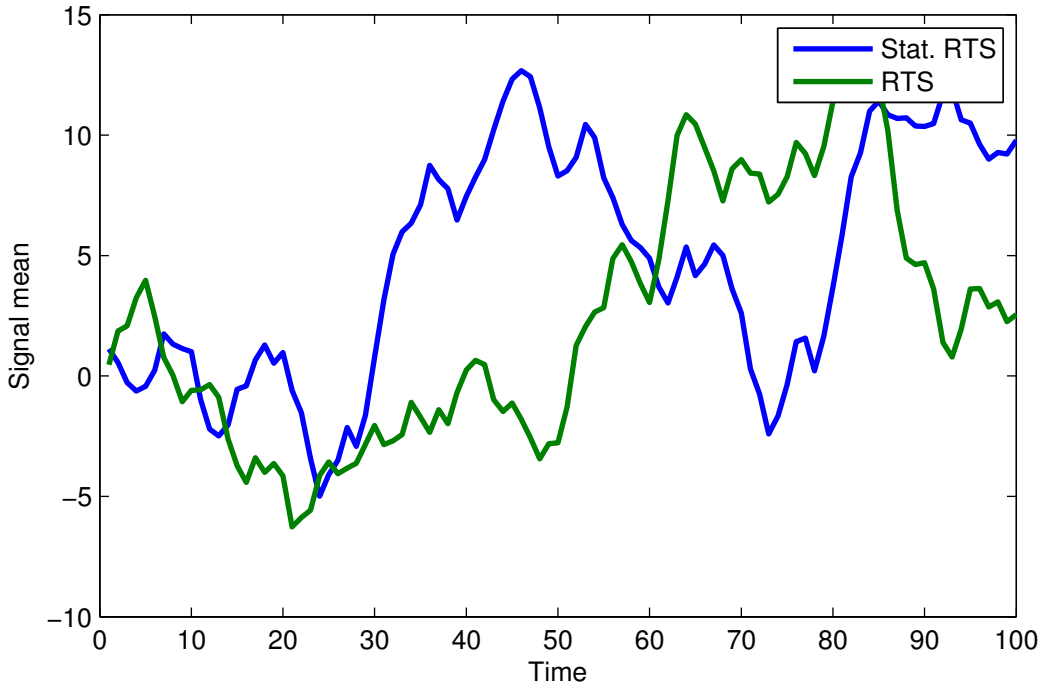


Figure 16: The RTS and the stationary RTS smoother's means in exercise 7.1C

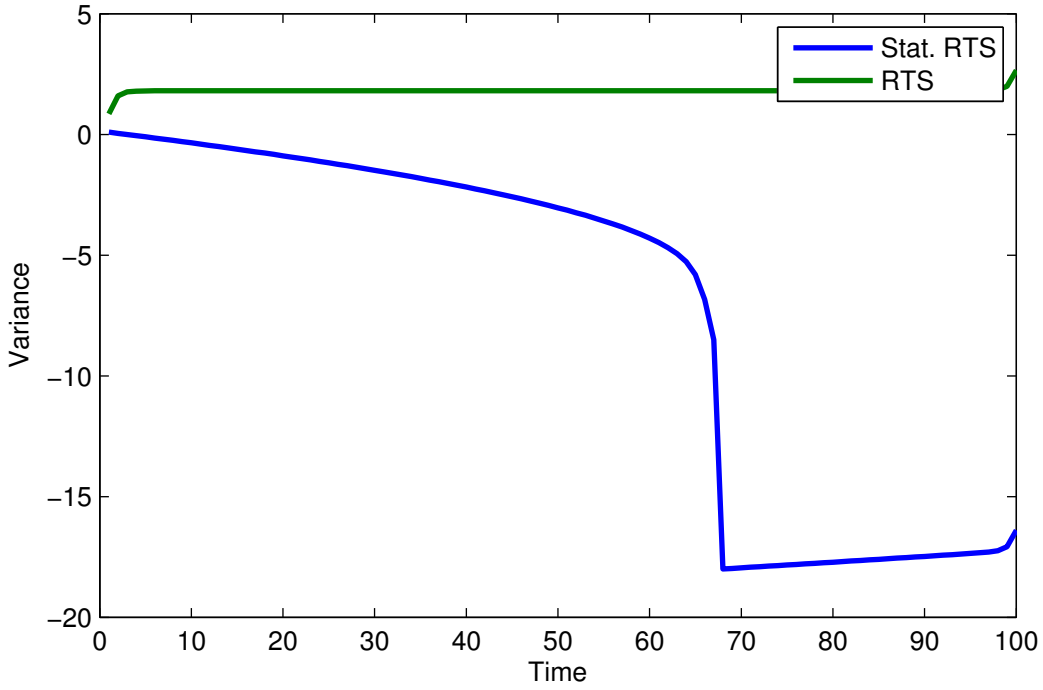


Figure 17: The RTS and the stationary RTS smoother's variances in exercise 7.1C

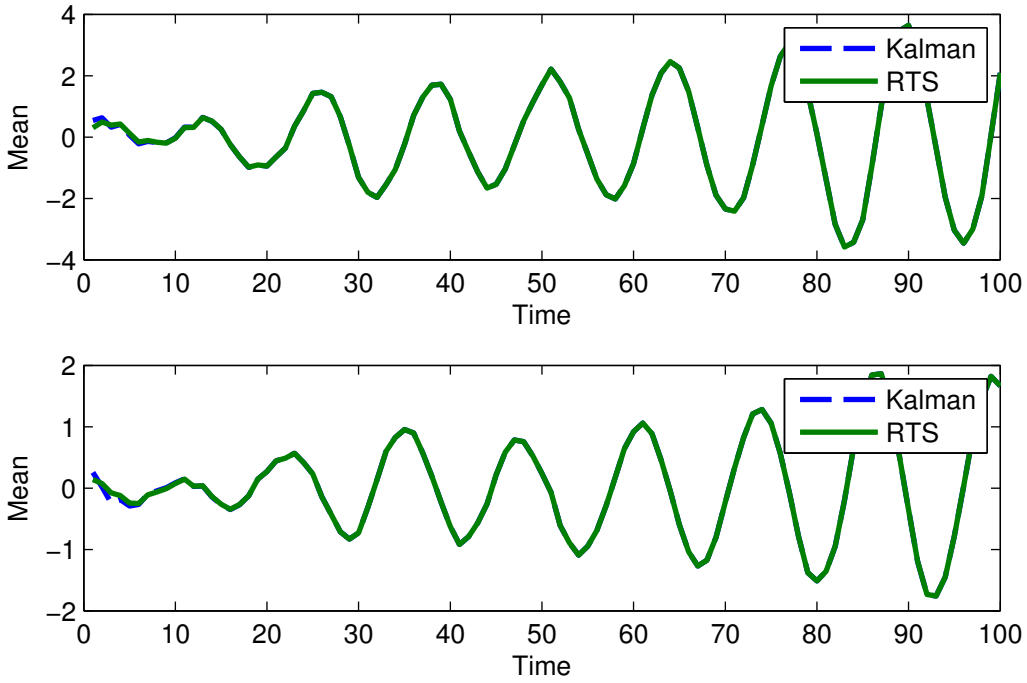


Figure 18: The Kalman filter's and RTS smoother's means in exercise 7.2

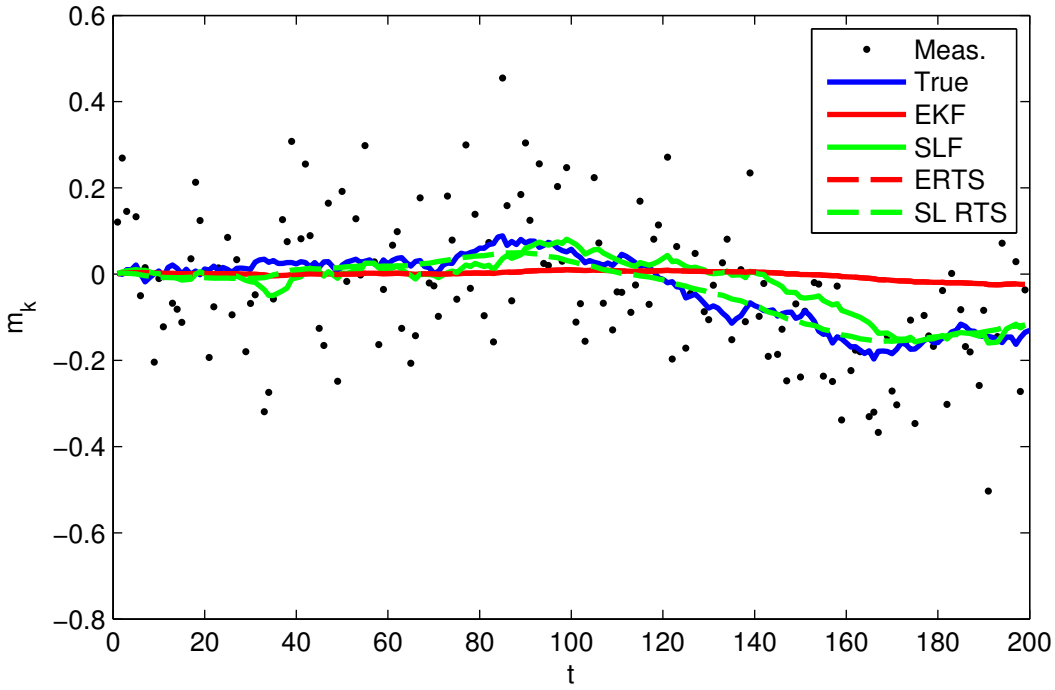


Figure 19: The RTS smoothers for the EKF (ERTS) and SLF (SL RTS) filters in exercise 7.2