Statistical Filtering and Smoothing - II

6. GAUSSIAN APPROXIMATION TO NONLINEAR FILTERING PROBLEMS

6.1 Gaussian Moment Matching

One way to unify various Gaussian approximations to non-linear transforms is to reduce them to approximate computation of Gaussian integrals of the form

$$\int \mathbf{g}(\mathbf{x})\, \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\, d\mathbf{x}.$$

If we can compute these, a straightforward way to form the Gaussian approximation for $(\mathbf{x}, \mathbf{y})$ is to simply match the moments of the distributions, which gives the following algorithm.

Algorithm 6.1 (Gaussian moment matching of an additive transform) *The moment matching based Gaussian approximation to the joint distribution of* $\mathbf{x}$ *and the transformed random variable* $\mathbf{y} = \mathbf{g}(\mathbf{x}) + \mathbf{q}$, *where* $\mathbf{x} \sim \mathrm{N}(\mathbf{m}, \mathbf{P})$ *and* $\mathbf{q} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$, *is given by*

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \mathbf{m} \\ \boldsymbol{\mu}_{\mathrm{M}} \end{pmatrix}, \begin{pmatrix} \mathbf{P} & \mathbf{C}_{\mathrm{M}} \\ \mathbf{C}_{\mathrm{M}}^{\mathsf{T}} & \mathbf{S}_{\mathrm{M}} \end{pmatrix} \right), \tag{6.1}$$

#

where

$$\boldsymbol{\mu}_{\mathrm{M}} = \int \mathbf{g}(\mathbf{x})\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x},$$

$$\mathbf{S}_{\mathrm{M}} = \int (\mathbf{g}(\mathbf{x}) - \boldsymbol{\mu}_{\mathrm{M}})\,(\mathbf{g}(\mathbf{x}) - \boldsymbol{\mu}_{\mathrm{M}})^{\mathsf{T}}\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x} + \mathbf{Q},$$

$$\mathbf{C}_{\mathrm{M}} = \int (\mathbf{x} - \mathbf{m})\,(\mathbf{g}(\mathbf{x}) - \boldsymbol{\mu}_{\mathrm{M}})^{\mathsf{T}}\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x}. \tag{6.2}$$

It is now easy to check by substituting the linear approximation $\mathbf{g}(\mathbf{x}) = \mathbf{g}(\mathbf{m}) + \mathbf{G}_{\mathbf{x}}(\mathbf{m})\,(\mathbf{x} - \mathbf{m})$ into the above expression that the integrals reduce to the linear approximations in Algorithm 5.1. The analogous thing happens with quadratic approximations and statistical linearization. The unscented transform can also be seen as a special case of the above algorithm, as we will see in the following sections.

The non-additive version of this transform is the following:

Algorithm 6.2 (Gaussian moment matching of a non-additive transform) *The moment matching based Gaussian approximation to the joint distribution of* $\mathbf{x}$ *and the transformed random variable* $\mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{q})$, *where* $\mathbf{x} \sim \mathrm{N}(\mathbf{m}, \mathbf{P})$ *and* $\mathbf{q} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$, *is given by*

#

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix} \mathbf{m} \\ \boldsymbol{\mu}_{\mathrm{M}} \end{pmatrix}, \begin{pmatrix} \mathbf{P} & \mathbf{C}_{\mathrm{M}} \\ \mathbf{C}_{\mathrm{M}}^{\mathsf{T}} & \mathbf{S}_{\mathrm{M}} \end{pmatrix} \right), \tag{6.3}$$

where

$$\begin{aligned} \boldsymbol{\mu}_{\mathrm{M}} &= \int \mathbf{g}(\mathbf{x}, \mathbf{q})\, \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\, \mathrm{N}(\mathbf{q} \mid \mathbf{0}, \mathbf{Q})\, \mathrm{d}\mathbf{x}\, \mathrm{d}\mathbf{q}, \\ \mathbf{S}_{\mathrm{M}} &= \int (\mathbf{g}(\mathbf{x}, \mathbf{q}) - \boldsymbol{\mu}_{\mathrm{M}})\, (\mathbf{g}(\mathbf{x}, \mathbf{q}) - \boldsymbol{\mu}_{\mathrm{M}})^{\mathsf{T}} \\ &\qquad \times \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\, \mathrm{N}(\mathbf{q} \mid \mathbf{0}, \mathbf{Q})\, \mathrm{d}\mathbf{x}\, \mathrm{d}\mathbf{q}, \\ \mathbf{C}_{\mathrm{M}} &= \int (\mathbf{x} - \mathbf{m})\, (\mathbf{g}(\mathbf{x}, \mathbf{q}) - \boldsymbol{\mu}_{\mathrm{M}})^{\mathsf{T}}\, \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\, \mathrm{N}(\mathbf{q} \mid \mathbf{0}, \mathbf{Q})\, \mathrm{d}\mathbf{x}\, \mathrm{d}\mathbf{q}. \end{aligned} \tag{6.4}$$

6.2 Gaussian Filter

The idea is to assume that the filtering distributuon is indeed Gaussian,

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \simeq \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k), \tag{6.5}$$

and approximate its mean $\mathbf{m}_k$ and covariance $\mathbf{P}_k$ via moment matching.

#

Algorithm 6.3 (Gaussian filter I) *The prediction and update steps of the additive noise Gaussian (Kalman) filter are:*

- *Prediction:*

$$\begin{aligned}
\mathbf{m}_k^- &= \int \mathbf{f}(\mathbf{x}_{k-1})\, \mathrm{N}(\mathbf{x}_{k-1} \mid \mathbf{m}_{k-1}, \mathbf{P}_{k-1})\, \mathrm{d}\mathbf{x}_{k-1}, \\
\mathbf{P}_k^- &= \int (\mathbf{f}(\mathbf{x}_{k-1}) - \mathbf{m}_k^-)\, (\mathbf{f}(\mathbf{x}_{k-1}) - \mathbf{m}_k^-)^{\mathsf{T}} \\
&\qquad \times \mathrm{N}(\mathbf{x}_{k-1} \mid \mathbf{m}_{k-1}, \mathbf{P}_{k-1})\, \mathrm{d}\mathbf{x}_{k-1} + \mathbf{Q}_{k-1}.
\end{aligned} \tag{6.6}$$

- *Update:*

$$\begin{aligned}
\boldsymbol{\mu}_k &= \int \mathbf{h}(\mathbf{x}_k)\, \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-)\, \mathrm{d}\mathbf{x}_k, \\
\mathbf{S}_k &= \int (\mathbf{h}(\mathbf{x}_k) - \boldsymbol{\mu}_k)\, (\mathbf{h}(\mathbf{x}_k) - \boldsymbol{\mu}_k)^{\mathsf{T}}\, \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-)\, \mathrm{d}\mathbf{x}_k + \mathbf{R}_k, \\
\mathbf{C}_k &= \int (\mathbf{x}_k - \mathbf{m}_k^-)\, (\mathbf{h}(\mathbf{x}_k) - \boldsymbol{\mu}_k)^{\mathsf{T}}\, \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-)\, \mathrm{d}\mathbf{x}_k, \\
\mathbf{K}_k &= \mathbf{C}_k\, \mathbf{S}_k^{-1}, \\
\mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\, (\mathbf{y}_k - \boldsymbol{\mu}_k), \\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\, \mathbf{S}_k\, \mathbf{K}_k^{\mathsf{T}}.
\end{aligned} \tag{6.7}$$

#

The moment matching enables the use of numerical integration methods such as Gauss-Hermite quadrature or Monte-Carlo .

The extension to non-additive noise is as follows.

Algorithm 6.4 (Gaussian filter II) *The prediction and update steps of the non-additive noise Gaussian (Kalman) filter are:*

- *Prediction:*

$$\mathbf{m}_k^- = \int \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1}) \times \mathrm{N}(\mathbf{x}_{k-1} \mid \mathbf{m}_{k-1}, \mathbf{P}_{k-1})\, \mathrm{N}(\mathbf{q}_{k-1} \mid \mathbf{0}, \mathbf{Q}_{k-1})\, \mathrm{d}\mathbf{x}_{k-1}\, \mathrm{d}\mathbf{q}_{k-1},$$

$$\mathbf{P}_k^- = \int (\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1}) - \mathbf{m}_k^-)(\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1}) - \mathbf{m}_k^-)^{\mathsf{T}} \times \mathrm{N}(\mathbf{x}_{k-1} \mid \mathbf{m}_{k-1}, \mathbf{P}_{k-1})\, \mathrm{N}(\mathbf{q}_{k-1} \mid \mathbf{0}, \mathbf{Q}_{k-1})\, \mathrm{d}\mathbf{x}_{k-1}\, \mathrm{d}\mathbf{q}_{k-1}. \quad (6.8)$$

- *Update:*

$$\boldsymbol{\mu}_k = \int \mathbf{h}(\mathbf{x}_k, \mathbf{r}_k) \times \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-)\, \mathrm{N}(\mathbf{r}_k \mid \mathbf{0}, \mathbf{R}_k)\, \mathrm{d}\mathbf{x}_k\, \mathrm{d}\mathbf{r}_k,$$

♯

$$
\begin{aligned}
\mathbf{S}_k &= \int (\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k) - \boldsymbol{\mu}_k)\,(\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k) - \boldsymbol{\mu}_k)^{\mathsf{T}} \\
&\qquad \times \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-)\ \mathrm{N}(\mathbf{r}_k \mid \mathbf{0}, \mathbf{R}_k)\ \mathrm{d}\mathbf{x}_k\ \mathrm{d}\mathbf{r}_k, \\
\mathbf{C}_k &= \int (\mathbf{x}_k - \mathbf{m}_k^-)\,(\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k) - \boldsymbol{\mu}_k)^{\mathsf{T}} \\
&\qquad \times \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-)\ \mathrm{N}(\mathbf{r}_k \mid \mathbf{0}, \mathbf{R}_k)\ \mathrm{d}\mathbf{x}_k\ \mathrm{d}\mathbf{r}_k, \\
\mathbf{K}_k &= \mathbf{C}_k\,\mathbf{S}_k^{-1}, \\
\mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\,(\mathbf{y}_k - \boldsymbol{\mu}_k), \\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\,\mathbf{S}_k\,\mathbf{K}_k^{\mathsf{T}}.
\end{aligned}
\tag{6.9}
$$

Remark:

The above general Gaussian filters are theoretical constructions rather than practical filtering algorithms. However, there exist many models for which the required integrals can indeed be computed in closed form. But a more practical approach is to compute them numerically. This kind of method will be discussed in the next sections.

#

6.3 Gauss-Hermite Integration

In the Gaussian filter (and later in the smoother) we are interested in approximating Gaussian integrals of the form

$$\int \mathbf{g}(\mathbf{x})\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x}$$
$$= \frac{1}{(2\pi)^{n/2}\,|\mathbf{P}|^{1/2}} \int \mathbf{g}(\mathbf{x}) \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^{\mathsf{T}}\,\mathbf{P}^{-1}\,(\mathbf{x}-\mathbf{m})\right) \mathrm{d}\mathbf{x}, \tag{6.10}$$

In its basic form, one-dimensional Gauss–Hermite quadrature integration refers to the special case of Gaussian quadratures with unit Gaussian weight function $w(x) = \mathrm{N}(x \mid 0, 1)$, that is, to approximations of the form

$$\int_{-\infty}^{\infty} g(x)\ \mathrm{N}(x \mid 0, 1)\ \mathrm{d}x \approx \sum_i W_i\, g(x^{(i)}), \tag{6.11}$$

where $W_i, i = 1, \ldots, p$ are the weights and $x^{(i)}$ are the evaluation points or abscissas – also sometimes called sigma points. Note that the quadrature is sometimes defined in terms of the weight function $\exp(-x^2)$, but here we shall use the "probabilists' definition" above. The two versions of the quadrature are related by simple scaling of variables.

#

Obviously, there are an infinite number of possible ways to select the weights and evaluation points. In Gauss–Hermite integration, as in all Gaussian quadratures, the weights and sigma points are chosen such that with a polynomial integrand the approximation becomes exact. It turns out that the polynomial order with a given number of points is maximized if we choose the sigma points to be roots of Hermite polynomials. When using the pth order Hermite polynomial $H_p(x)$, the rule will be exact for polynomials up to order $2p-1$. The required weights can be computed in closed form (see below).

The Hermite polynomial of order p is defined as (these are the so-called "probabilists' Hermite polynomials"):

$$H_p(x) = (-1)^p \exp(x^2/2) \frac{\mathrm{d}^p}{\mathrm{d}x^p} \exp(-x^2/2). \tag{6.12}$$

The first few Hermite polynomials are:

$$\begin{aligned} H_0(x) &= 1, \\ H_1(x) &= x, \\ H_2(x) &= x^2 - 1, \\ H_3(x) &= x^3 - 3x, \\ H_4(x) &= x^4 - 6x^2 + 3, \end{aligned} \tag{6.13}$$

#

and further polynomials can be found from the recursion

$$H_{p+1}(x) = x\, H_p(x) - p\, H_{p-1}(x). \tag{6.14}$$

Using the same weights and sigma points, integrals over non-unit Gaussian weights functions $\mathrm{N}(x \mid m, P)$ can be evaluated using a simple change of integration variable:

$$\int_{-\infty}^{\infty} g(x)\ \mathrm{N}(x \mid m, P)\ \mathrm{d}x = \int_{-\infty}^{\infty} g(P^{1/2}\,\xi + m)\ \mathrm{N}(\xi \mid 0, 1)\ \mathrm{d}\xi. \tag{6.15}$$

Gauss–Hermite integration can be written as the following algorithm.

Algorithm 6.5 (Gauss–Hermite quadrature) *The pth order Gauss–Hermite approximation to the one-dimensional integral*

$$\int_{-\infty}^{\infty} g(x)\ \mathrm{N}(x \mid m, P)\ \mathrm{d}x \tag{6.16}$$

can be computed as follows:

1 *Compute the unit sigma points as the roots* $\xi^{(i)}, i = 1, \ldots, p$ *of the Hermite polynomial* $H_p(x)$. *Note that we do not need to form the polynomial and then compute its roots, but instead it is numerically more stable to*
compute the roots as eigenvalues of a suitable tridiagonal matrix (Golub and Welsch, 1969).

2 Compute the weights as

$$W_i = \frac{p!}{p^2\,[H_{p-1}(\xi^{(i)})]^2}. \tag{6.17}$$

3 Approximate the integral as

$$\int_{-\infty}^{\infty} g(x)\ \mathrm{N}(x \mid m, P)\ \mathrm{d}x \approx \sum_{i=1}^{p} W_i\, g(P^{1/2}\,\xi^{(i)} + m). \tag{6.18}$$

By generalizing the change of variables idea, we can form approximations to multi-dimensional integrals of the form (6.10). First let $\mathbf{P} = \sqrt{\mathbf{P}}\,\sqrt{\mathbf{P}}^{\mathsf{T}}$, where $\sqrt{\mathbf{P}}$ is the Cholesky factor of the covariance matrix $\mathbf{P}$ or some other similar square root of the covariance matrix. If we define new integration variables $\boldsymbol{\xi}$ by

$$\mathbf{x} = \mathbf{m} + \sqrt{\mathbf{P}}\,\boldsymbol{\xi}, \tag{6.19}$$

we get

$$\int \mathbf{g}(\mathbf{x})\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x} = \int \mathbf{g}(\mathbf{m} + \sqrt{\mathbf{P}}\,\boldsymbol{\xi})\ \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\ \mathrm{d}\boldsymbol{\xi}. \tag{6.20}$$

#

The integration over the multi-dimensional unit Gaussian distribution can be written as an iterated integral over one-dimensional Gaussian distributions, and each of the one-dimensional integrals can be approximated with Gauss–Hermite quadrature:

$$\begin{aligned}
&\int \mathbf{g}(\mathbf{m} + \sqrt{\mathbf{P}}\,\boldsymbol{\xi})\ \mathrm{N}(\boldsymbol{\xi} \mid \mathbf{0}, \mathbf{I})\ \mathrm{d}\boldsymbol{\xi} \\
&\quad = \int \cdots \int \mathbf{g}(\mathbf{m} + \sqrt{\mathbf{P}}\,\boldsymbol{\xi})\ \mathrm{N}(\xi_1 \mid 0, 1)\ \mathrm{d}\xi_1 \times \cdots \times \mathrm{N}(\xi_n \mid 0, 1)\ \mathrm{d}\xi_n \\
&\quad \approx \sum_{i_1, \ldots, i_n} W_{i_1} \times \cdots \times W_{i_n} \mathbf{g}(\mathbf{m} + \sqrt{\mathbf{P}}\,\boldsymbol{\xi}^{(i_1, \ldots, i_n)}).
\end{aligned} \tag{6.21}$$

The weights $W_{i_k}, k = 1, \ldots, n$ are simply the corresponding one-dimensional Gauss–Hermite weights and $\boldsymbol{\xi}^{(i_1, \ldots, i_n)}$ is an n-dimensional vector with one-dimensional unit sigma point $\xi^{(i_k)}$ at element k. The algorithm can now be written as follows.

Algorithm 6.6 (Gauss–Hermite cubature) *The pth order Gauss–Hermite approximation to the multi-dimensional integral*

$$\int \mathbf{g}(\mathbf{x})\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x} \tag{6.22}$$

can be computed as follows.

1 *Compute the one-dimensional weights* $W_i, i = 1, \ldots, p$ *and unit sigma points* $\xi^{(i)}$ *as in the one-dimensional Gauss–Hermite quadrature Algorithm 6.5.*

2 *Form multi-dimensional weights as the products of one-dimensional weights:*

$$\begin{aligned} W_{i_1,\ldots,i_n} &= W_{i_1} \times \cdots \times W_{i_n} \\ &= \frac{p!}{p^2\,[H_{p-1}(\xi^{(i_1)})]^2} \times \cdots \times \frac{p!}{p^2\,[H_{p-1}(\xi^{(i_n)})]^2}, \end{aligned} \tag{6.23}$$

where each i_k *takes values* $1, \ldots, p$.

3 *Form multi-dimensional unit sigma points as Cartesian product of the one-dimensional unit sigma points:*

$$\boldsymbol{\xi}^{(i_1,\ldots,i_n)} = \begin{pmatrix} \xi^{(i_1)} \\ \vdots \\ \xi^{(i_n)} \end{pmatrix}. \tag{6.24}$$

4 *Approximate the integral as*

$$\int \mathbf{g}(\mathbf{x})\ \mathrm{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{P})\ \mathrm{d}\mathbf{x} \approx \sum_{i_1,\ldots,i_n} W_{i_1,\ldots,i_n} \mathbf{g}(\mathbf{m} + \sqrt{\mathbf{P}}\,\boldsymbol{\xi}^{(i_1,\ldots,i_n)}), \tag{6.25}$$

#

where $\sqrt{\mathbf{P}}$ *is a matrix square root defined by* $\mathbf{P} = \sqrt{\mathbf{P}}\sqrt{\mathbf{P}}^{\mathsf{T}}$.

The pth order multi-dimensional Gauss–Hermite integration is exact for monomials of the form $x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$, and their arbitrary linear combinations, where each of the orders $d_i \leq 2p - 1$. The number of sigma points required for an n-dimensional integral with pth order rule is p^n, which quickly becomes unfeasible when the number of dimensions grows.

The figure 6.1 illustrates how the sigma points are selected in Gauss-Hermite cubatures:

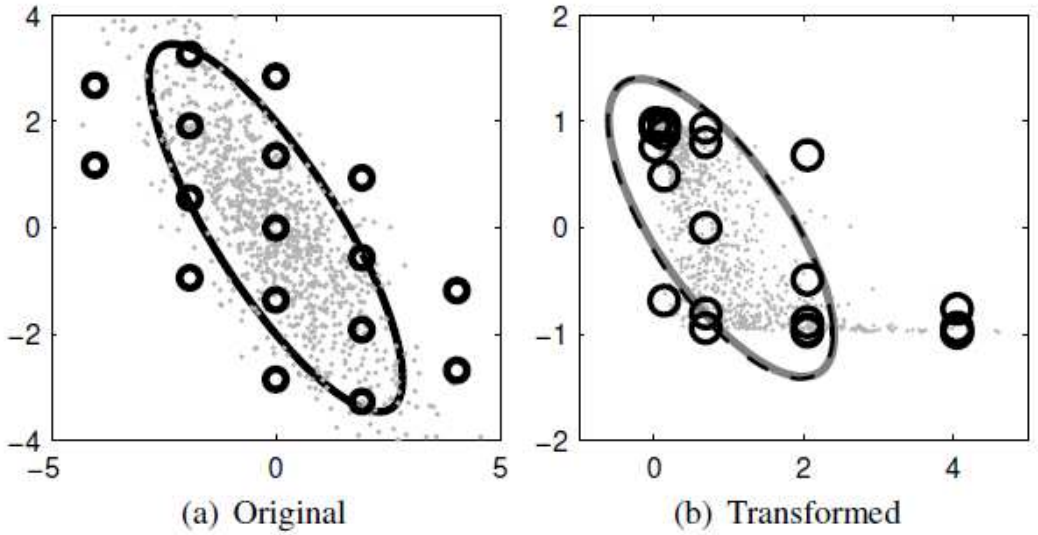


Figure 6.1 Illustration of a fifth order Gauss–Hermite cubature

#

6.4 Gauss-Hermite Kalman Filter

The additive form multi-dimensional Gauss–Hermite cubature based *Gauss–Hermite (Kalman) filter (GHKF)* (Ito and Xiong, 2000) which is also called the *quadrature Kalman filter (QKF)* (Arasaratnam et al., 2007) can be derived by replacing the Gaussian integrals in the Gaussian filter Algorithm 6.3 with the Gauss–Hermite approximations in Algorithm 6.6.

Algorithm 6.7 (Gauss–Hermite Kalman filter) *The additive form Gauss–Hermite Kalman filter (GHKF) algorithm is the following.*

- *Prediction:*

  1 *Form the sigma points as:*

$$\mathcal{X}_{k-1}^{(i_1,\ldots,i_n)} = \mathbf{m}_{k-1} + \sqrt{\mathbf{P}_{k-1}}\,\boldsymbol{\xi}^{(i_1,\ldots,i_n)}, \quad i_1,\ldots,i_n = 1,\ldots,p, \tag{6.26}$$

  where the unit sigma points $\boldsymbol{\xi}^{(i_1,\ldots,i_n)}$ *were defined in Equation* (6.24).

  2 *Propagate the sigma points through the dynamic model:*

$$\hat{\mathcal{X}}_k^{(i_1,\ldots,i_n)} = \mathbf{f}(\mathcal{X}_{k-1}^{(i_1,\ldots,i_n)}), \quad i_1,\ldots,i_n = 1,\ldots,p. \tag{6.27}$$

#

3 *Compute the predicted mean* $\mathbf{m}_k^-$ *and the predicted covariance* $\mathbf{P}_k^-$*:*

$$\mathbf{m}_k^- = \sum_{i_1,\ldots,i_n} W_{i_1,\ldots,i_n} \hat{\mathcal{X}}_k^{(i_1,\ldots,i_n)},$$

$$\mathbf{P}_k^- = \sum_{i_1,\ldots,i_n} W_{i_1,\ldots,i_n} (\hat{\mathcal{X}}_k^{(i_1,\ldots,i_n)} - \mathbf{m}_k^-)\,(\hat{\mathcal{X}}_k^{(i_1,\ldots,i_n)} - \mathbf{m}_k^-)^{\mathsf{T}} + \mathbf{Q}_{k-1},$$

where the weights $W_{i_1,\ldots,i_n}$ *were defined in Equation* (6.23). (6.28)

- *Update:*

  1 *Form the sigma points:*

$$\mathcal{X}_k^{-(i_1,\ldots,i_n)} = \mathbf{m}_k^- + \sqrt{\mathbf{P}_k^-}\,\boldsymbol{\xi}^{(i_1,\ldots,i_n)}, \quad i_1,\ldots,i_n = 1,\ldots,p, \tag{6.29}$$

  where the unit sigma points $\boldsymbol{\xi}^{(i_1,\ldots,i_n)}$ *were defined in Equation* (6.24).

  2 *Propagate sigma points through the measurement model:*

$$\hat{\mathcal{Y}}_k^{(i_1,\ldots,i_n)} = \mathbf{h}(\mathcal{X}_k^{-(i_1,\ldots,i_n)}), \quad i_1,\ldots,i_n = 1,\ldots,p. \tag{6.30}$$

  3 *Compute the predicted mean* $\boldsymbol{\mu}_k$*, the predicted covariance of the measurement* $\mathbf{S}_k$*, and the cross-covariance of the state and the measurement* $\mathbf{C}_k$*:*

#

$$
\begin{aligned}
\boldsymbol{\mu}_k &= \sum_{i_1,\ldots,i_n} W_{i_1,\ldots,i_n} \hat{\mathcal{Y}}_k^{(i_1,\ldots,i_n)}, \\
\mathbf{S}_k &= \sum_{i_1,\ldots,i_n} W_{i_1,\ldots,i_n} (\hat{\mathcal{Y}}_k^{(i_1,\ldots,i_n)} - \boldsymbol{\mu}_k) (\hat{\mathcal{Y}}_k^{(i_1,\ldots,i_n)} - \boldsymbol{\mu}_k)^{\mathsf{T}} + \mathbf{R}_k, \\
\mathbf{C}_k &= \sum_{i_1,\ldots,i_n} W_{i_1,\ldots,i_n} (\mathcal{X}_k^{-(i_1,\ldots,i_n)} - \mathbf{m}_k^-) (\hat{\mathcal{Y}}_k^{(i_1,\ldots,i_n)} - \boldsymbol{\mu}_k)^{\mathsf{T}},
\end{aligned}
\tag{6.31}
$$

where the weights $W_{i_1,\ldots,i_n}$ *were defined in Equation* (6.23).

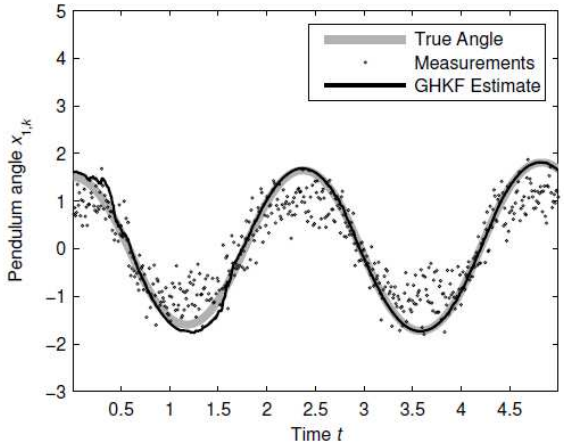
4 *Compute the filter gain* $\mathbf{K}_k$*, the filtered state mean* $\mathbf{m}_k$ *and the covariance* $\mathbf{P}_k$*, conditional on the measurement* $\mathbf{y}_k$*:*

$$
\begin{aligned}
\mathbf{K}_k &= \mathbf{C}_k \, \mathbf{S}_k^{-1}, \\
\mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k \, [\mathbf{y}_k - \boldsymbol{\mu}_k], \\
\mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k \, \mathbf{S}_k \, \mathbf{K}_k^{\mathsf{T}}.
\end{aligned}
\tag{6.32}
$$

The non-additive version can be obtained by applying the Gauss–Hermite quadrature to the non-additive Gaussian filter Algorithm 6.4 in a similar manner. However, due to the rapid growth of computational requirements in state dimension the augmented form is computationally quite heavy, because it requires roughly doubling the dimensionality of the integration variable.

#

Example: Pendulum (or sinusoid) tracking with GHKF



Exercises:

1. Show that one-dimensional Hermite polynomials are orthogonal with respect to the inner product

$$\langle f, g \rangle = \int f(x)\, g(x)\, \mathrm{N}(x \mid 0, 1)\, \mathrm{d}x.$$

2. Show that multivariate Hermite polynomials defined as

$$H_{i_1,\ldots,i_n}(\mathbf{x}) = H_{i_1}(x_1) \times \cdots \times H_{i_n}(x_n)$$

are orthogonal with respect to the inner product

$$\langle f, g \rangle = \int f(\mathbf{x})\, g(\mathbf{x})\, \mathrm{N}(\mathbf{x} \mid \mathbf{0}, \mathbf{I})\, \mathrm{d}\mathbf{x}.$$

7. PARTICLE FILTERING

Although in many filtering problems Gaussian approximations work well, sometimes the filtering distributions can be, for example, multi-modal, or some of the state components might be discrete, in which case Gaussian approximations are not appropriate. In such cases sequential importance resampling based particle filters can be a better alternative. This chapter is concerned with particle filters, which are methods for forming Monte Carlo approximations to the solutions of the Bayesian filtering equations.

7.1 Monte Carlo Approximations in Bayesian Inference

In Bayesian inference, including Bayesian filtering, the main inference problem can often be reduced into computation of the following kind of expectations over the posterior distribution:[1]

$$\mathrm{E}[\mathbf{g}(\mathbf{x}) \mid \mathbf{y}_{1:T}] = \int \mathbf{g}(\mathbf{x})\, p(\mathbf{x} \mid \mathbf{y}_{1:T})\, \mathrm{d}\mathbf{x}, \tag{7.1}$$

where $\mathbf{g} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is an arbitrary function and $p(\mathbf{x} \mid \mathbf{y}_{1:T})$ is the posterior probability density of $\mathbf{x}$ given the measurements $\mathbf{y}_1, \ldots, \mathbf{y}_T$. Now the problem is that such an integral can be evaluated in closed form only in a few special cases and generally, numerical methods have to be used.

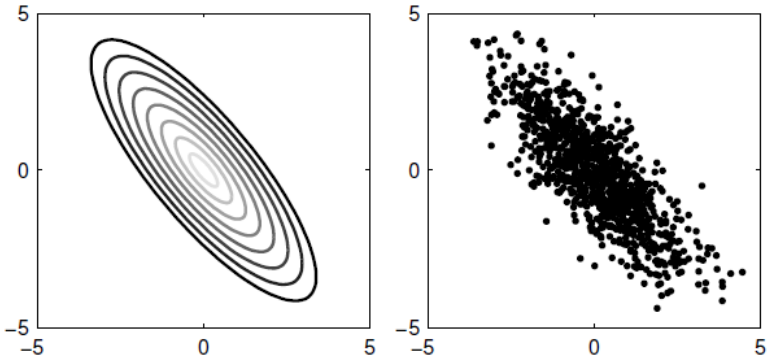
[1] In this section we formally treat $\mathbf{x}$ as a continuous random variable with a density, but the analogous results apply to discrete random variables.

Monte Carlo methods provide a numerical method for calculating integrals of the form (7.1). Monte Carlo refers to a general class of methods where closed form computation of statistical quantities is replaced by drawing samples from the distribution and estimating the quantities by sample averages.

In a (perfect) Monte Carlo approximation, we draw N independent random samples $\mathbf{x}^{(i)} \sim p(\mathbf{x} \mid \mathbf{y}_{1:T}), i = 1, \ldots, N$ and estimate the expectation as

$$\mathrm{E}[\mathbf{g}(\mathbf{x}) \mid \mathbf{y}_{1:T}] \approx \frac{1}{N} \sum_{i=1}^{N} \mathbf{g}(\mathbf{x}^{(i)}). \tag{7.2}$$

Thus Monte Carlo methods approximate the target distribution by a set of samples that are distributed according to the target density. Figure 7.1 represents a two-dimensional Gaussian distribution and its Monte Carlo representation.



#

The convergence of the Monte Carlo approximation is guaranteed by the central limit theorem (CLT, see, e.g., Liu, 2001) and the error term is $O(N^{-1/2})$, regardless of the dimensionality of $\mathbf{x}$. This invariance with respect to dimensionality is unique to Monte Carlo methods and makes them superior to practically all other numerical methods when the dimensionality of $\mathbf{x}$ is considerable. At least in theory, not necessarily in practice (see Daum and Huang, 2003; Snyder et al., 2008).

7.2 Importance Sampling

Often, in practical Bayesian models, it is not possible to obtain samples directly from $p(\mathbf{x} \mid \mathbf{y}_{1:T})$ due to its complicated functional form. In *importance sampling* (IS) (see, e.g., Liu, 2001) we use an approximate distribution called the importance distribution $\pi(\mathbf{x} \mid \mathbf{y}_{1:T})$, from which we can easily draw samples. Importance sampling is based on the following decomposition of the expectation over the posterior probability density $p(\mathbf{x} \mid \mathbf{y}_{1:T})$:

$$\int \mathbf{g}(\mathbf{x})\, p(\mathbf{x} \mid \mathbf{y}_{1:T})\, \mathrm{d}\mathbf{x} = \int \left[\mathbf{g}(\mathbf{x}) \frac{p(\mathbf{x} \mid \mathbf{y}_{1:T})}{\pi(\mathbf{x} \mid \mathbf{y}_{1:T})} \right] \pi(\mathbf{x} \mid \mathbf{y}_{1:T})\, \mathrm{d}\mathbf{x}, \quad (7.3)$$

#

where the importance density $\pi(\mathbf{x} \mid \mathbf{y}_{1:T})$ is required to be non-zero whenever $p(\mathbf{x} \mid \mathbf{y}_{1:T})$ is non-zero, that is, the *support* of $\pi(\mathbf{x} \mid \mathbf{y}_{1:T})$ needs to be greater than or equal to the support of $p(\mathbf{x} \mid \mathbf{y}_{1:T})$. As the above expression is just the expectation of the term in the brackets over the distribution $\pi(\mathbf{x} \mid \mathbf{y}_{1:T})$, we can form a Monte Carlo approximation to it by drawing N samples from the importance distribution:

$$\mathbf{x}^{(i)} \sim \pi(\mathbf{x} \mid \mathbf{y}_{1:T}), \qquad i = 1, \ldots, N, \tag{7.4}$$

and by forming the approximation as

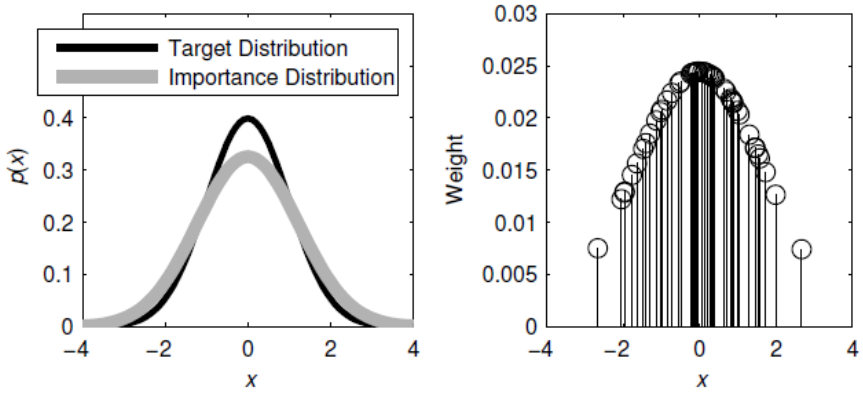
$$\begin{aligned} \mathrm{E}[\mathbf{g}(\mathbf{x}) \mid \mathbf{y}_{1:T}] &\approx \frac{1}{N} \sum_{i=1}^{N} \frac{p(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})}{\pi(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})} \mathbf{g}(\mathbf{x}^{(i)}) \\ &= \sum_{i=1}^{N} \tilde{w}^{(i)} \mathbf{g}(\mathbf{x}^{(i)}), \end{aligned} \tag{7.5}$$

where the weights have been defined as

$$\tilde{w}^{(i)} = \frac{1}{N} \frac{p(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})}{\pi(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})}. \tag{7.6}$$

#

Figure 7.2 illustrates the idea of importance sampling. We sample from the importance distribution which is an approximation to the target distribution. Because the distribution of samples is not exact, we need to correct the approximation by associating a weight with each of the samples.



The disadvantage of this direct importance sampling is that we should be able to evaluate $p(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})$ in order to use it directly. Recall that by Bayes' rule the evaluation of the posterior probability density can be written as

$$p(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T}) = \frac{p(\mathbf{y}_{1:T} \mid \mathbf{x}^{(i)})\, p(\mathbf{x}^{(i)})}{\int p(\mathbf{y}_{1:T} \mid \mathbf{x})\, p(\mathbf{x})\, \mathrm{d}\mathbf{x}}. \tag{7.7}$$

#

The likelihood $p(\mathbf{y}_{1:T} \mid \mathbf{x}^{(i)})$ and prior terms $p(\mathbf{x}^{(i)})$ are usually easy to evaluate but often the integral in the denominator – the normalization constant – cannot be computed. To overcome this problem, we can form an importance sampling approximation to the expectation integral by also approximating the normalization constant by importance sampling. For this purpose we can decompose the expectation integral and form the approximation as follows.

$$
\begin{aligned}
\mathrm{E}[\mathbf{g}(\mathbf{x}) \mid \mathbf{y}_{1:T}] &= \int \mathbf{g}(\mathbf{x})\, p(\mathbf{x} \mid \mathbf{y}_{1:T})\, \mathrm{d}\mathbf{x} \\
&= \frac{\int \mathbf{g}(\mathbf{x})\, p(\mathbf{y}_{1:T} \mid \mathbf{x})\, p(\mathbf{x})\, \mathrm{d}\mathbf{x}}{\int p(\mathbf{y}_{1:T} \mid \mathbf{x})\, p(\mathbf{x})\, \mathrm{d}\mathbf{x}} \\
&= \frac{\int \left[\frac{p(\mathbf{y}_{1:T}|\mathbf{x})\, p(\mathbf{x})}{\pi(\mathbf{x}|\mathbf{y}_{1:T})} \mathbf{g}(\mathbf{x}) \right] \pi(\mathbf{x} \mid \mathbf{y}_{1:T})\, \mathrm{d}\mathbf{x}}{\int \left[\frac{p(\mathbf{y}_{1:T}|\mathbf{x})\, p(\mathbf{x})}{\pi(\mathbf{x}|\mathbf{y}_{1:T})} \right] \pi(\mathbf{x} \mid \mathbf{y}_{1:T})\, \mathrm{d}\mathbf{x}} \\
&\approx \frac{\frac{1}{N} \sum_{i=1}^{N} \frac{p(\mathbf{y}_{1:T}|\mathbf{x}^{(i)})\, p(\mathbf{x}^{(i)})}{\pi(\mathbf{x}^{(i)}|\mathbf{y}_{1:T})} \mathbf{g}(\mathbf{x}^{(i)})}{\frac{1}{N} \sum_{j=1}^{N} \frac{p(\mathbf{y}_{1:T}|\mathbf{x}^{(j)})\, p(\mathbf{x}^{(j)})}{\pi(\mathbf{x}^{(j)}|\mathbf{y}_{1:T})}} \\
&= \sum_{i=1}^{N} \underbrace{\left[\frac{\frac{p(\mathbf{y}_{1:T}|\mathbf{x}^{(i)})\, p(\mathbf{x}^{(i)})}{\pi(\mathbf{x}^{(i)}|\mathbf{y}_{1:T})}}{\sum_{j=1}^{N} \frac{p(\mathbf{y}_{1:T}|\mathbf{x}^{(j)})\, p(\mathbf{x}^{(j)})}{\pi(\mathbf{x}^{(j)}|\mathbf{y}_{1:T})}} \right]}_{w^{(i)}} \mathbf{g}(\mathbf{x}^{(i)}).
\end{aligned}
\tag{7.8}
$$

#

Thus we get the following algorithm.

Algorithm 7.1 (Importance sampling) *Given a measurement model* $p(\mathbf{y}_{1:T} \mid \mathbf{x})$ *and a prior* $p(\mathbf{x})$ *we can form an importance sampling approximation to the posterior as follows.*

1 *Draw* N *samples from the importance distribution:*

$$\mathbf{x}^{(i)} \sim \pi(\mathbf{x} \mid \mathbf{y}_{1:T}), \qquad i = 1, \ldots, N. \tag{7.9}$$

2 *Compute the unnormalized weights by*

$$w^{*(i)} = \frac{p(\mathbf{y}_{1:T} \mid \mathbf{x}^{(i)})\, p(\mathbf{x}^{(i)})}{\pi(\mathbf{x}^{(i)} \mid \mathbf{y}_{1:T})}, \tag{7.10}$$

and the normalized weights by

$$w^{(i)} = \frac{w^{*(i)}}{\sum_{j=1}^{N} w^{*(j)}}. \tag{7.11}$$

3 *The approximation to the posterior expectation of* $\mathbf{g}(\mathbf{x})$ *is then given as*

$$\mathrm{E}[\mathbf{g}(\mathbf{x}) \mid \mathbf{y}_{1:T}] \approx \sum_{i=1}^{N} w^{(i)}\, \mathbf{g}(\mathbf{x}^{(i)}). \tag{7.12}$$

#

The approximation to the posterior probability density formed by the above algorithm can then be formally written as

$$p(\mathbf{x} \mid \mathbf{y}_{1:T}) \approx \sum_{i=1}^{N} w^{(i)}\, \delta(\mathbf{x} - \mathbf{x}^{(i)}), \tag{7.13}$$

where $\delta(\cdot)$ is the Dirac delta function.

7.3 Sequential Importance Sampling

Sequential importance sampling (SIS) (see, e.g., Doucet et al., 2001) is a sequential version of importance sampling. The SIS algorithm can be used for generating importance sampling approximations to filtering distributions of generic state space models of the form

$$\begin{aligned} \mathbf{x}_k &\sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}), \\ \mathbf{y}_k &\sim p(\mathbf{y}_k \mid \mathbf{x}_k), \end{aligned} \tag{7.14}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state at time step k and $\mathbf{y}_k \in \mathbb{R}^m$ is the measurement. The state and measurements may contain both discrete and continuous components.

#

The SIS algorithm uses a weighted set of *particles* $\{(w_k^{(i)}, \mathbf{x}_k^{(i)}) : i = 1, \ldots, N\}$, that is, samples from an importance distribution and their weights, for representing the filtering distribution $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$ such that at every time step k the approximation to the expectation of an arbitrary function $\mathbf{g}(\cdot)$ can be calculated as the weighted sample average

$$\mathrm{E}[\mathbf{g}(\mathbf{x}_k) \mid \mathbf{y}_{1:k}] \approx \sum_{i=1}^{N} w_k^{(i)} \mathbf{g}(\mathbf{x}_k^{(i)}). \tag{7.15}$$

Equivalently, SIS can be interpreted as forming an approximation to the filtering distribution as

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}). \tag{7.16}$$

To derive the algorithm, we consider the full posterior distribution of states $\mathbf{x}_{0:k}$ given the measurements $\mathbf{y}_{1:k}$. By using the Markov properties of the model, we get the following recursion for the posterior distribution:

$$\begin{aligned} p(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k}) &\propto p(\mathbf{y}_k \mid \mathbf{x}_{0:k}, \mathbf{y}_{1:k-1})\, p(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k-1}) \\ &= p(\mathbf{y}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k-1})\, p(\mathbf{x}_{0:k-1} \mid \mathbf{y}_{1:k-1}) \\ &= p(\mathbf{y}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\, p(\mathbf{x}_{0:k-1} \mid \mathbf{y}_{1:k-1}). \end{aligned} \tag{7.17}$$

#

Using a similar rationale as in the previous section, we can now construct an importance sampling method which draws samples from a given importance distribution $\mathbf{x}_{0:k}^{(i)} \sim \pi(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k})$ and computes the importance weights by

$$w_k^{(i)} \propto \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)})\, p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)})\, p(\mathbf{x}_{0:k-1}^{(i)} \mid \mathbf{y}_{1:k-1})}{\pi(\mathbf{x}_{0:k}^{(i)} \mid \mathbf{y}_{1:k})}. \tag{7.18}$$

If we form the importance distribution for the states $\mathbf{x}_k$ recursively as follows:

$$\pi(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k}) = \pi(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})\, \pi(\mathbf{x}_{0:k-1} \mid \mathbf{y}_{1:k-1}), \tag{7.19}$$

then the expression for the weights can be written as

$$w_k^{(i)} \propto \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)})\, p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)})}{\pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k})} \frac{p(\mathbf{x}_{0:k-1}^{(i)} \mid \mathbf{y}_{1:k-1})}{\pi(\mathbf{x}_{0:k-1}^{(i)} \mid \mathbf{y}_{1:k-1})}. \tag{7.20}$$

Let's now assume that we have already drawn the samples $\mathbf{x}_{0:k-1}^{(i)}$ from the importance distribution $\pi(\mathbf{x}_{0:k-1} \mid \mathbf{y}_{1:k-1})$ and computed the corresponding importance weights $w_{k-1}^{(i)}$. We can now draw samples $\mathbf{x}_{0:k}^{(i)}$ from the importance distribution $\pi(\mathbf{x}_{0:k} \mid \mathbf{y}_{1:k})$ by drawing the new state samples for the step k as $\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k})$. The importance weights from
the previous step are proportional to the last term in Equation (7.20):

$$w_{k-1}^{(i)} \propto \frac{p(\mathbf{x}_{0:k-1}^{(i)} \mid \mathbf{y}_{1:k-1})}{\pi(\mathbf{x}_{0:k-1}^{(i)} \mid \mathbf{y}_{1:k-1})}, \tag{7.21}$$

and thus the weights satisfy the recursion

$$w_k^{(i)} \propto \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)})\, p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)})}{\pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k})}\, w_{k-1}^{(i)}. \tag{7.22}$$

The generic sequential importance sampling algorithm can now be described as follows.

Algorithm 7.2 (Sequential importance sampling) *Steps of SIS are the following:*

- *Draw N samples $\mathbf{x}_0^{(i)}$ from the prior*

$$\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0), \qquad i = 1, \ldots, N, \tag{7.23}$$

  and set $w_0^{(i)} = 1/N$, for all $i = 1, \ldots, N$.
- *For each $k = 1, \ldots, T$ do the following.*

  1 Draw samples $\mathbf{x}_k^{(i)}$ from the importance distributions

$$\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k}), \qquad i = 1, \ldots, N. \tag{7.24}$$

#

2 *Calculate new weights according to*

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)})\; p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)})}{\pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{\mathbf{0}:k-1}^{(i)}, \mathbf{y}_{1:k})} \tag{7.25}$$

and normalize them to sum to unity.

Note that it is convenient to select the importance distribution to be Markovian in the sense that

$$\pi(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k}) = \pi(\mathbf{x}_k \mid \mathbf{x}_{k-1}, \mathbf{y}_{1:k}). \tag{7.26}$$

With this form of importance distribution we do not need to store the whole histories $\mathbf{x}_{\mathbf{0}:k}^{(i)}$ in the SIS algorithm, only the current states $\mathbf{x}_k^{(i)}$. This form is also convenient in sequential importance resampling (SIR) discussed in the next section, because we do not need to worry about the state histories during the resampling step as in the SIR particle smoother (see Section 11.1). Thus in the following section we assume that the importance distribution has indeed been selected to have the above Markovian form.

#

7.1 Sequential Importance Resampling

One problem in the SIS algorithm described in the previous section is that we easily encounter the situation that almost all the particles have zero or nearly zero weights. This is called the *degeneracy problem* in particle filtering literature and it prevented practical applications of particle filters for many years.

The degeneracy problem can be solved by using a *resampling* procedure. It refers to a procedure where we draw N new samples from the discrete distribution defined by the weights and replace the old set of N samples with this new set. This procedure can be be written as the following algorithm:

Algorithm 7.3 (Resampling) *The resampling procedure can be described as follows.*

1 *Interpret each weight* $w_k^{(i)}$ *as the probability of obtaining the sample index* i *in the set* $\{\mathbf{x}_k^{(i)} : i = 1, \ldots, N\}$.
2 *Draw* N *samples from that discrete distribution and replace the old sample set with this new one.*
3 *Set all weights to the constant value* $w_k^{(i)} = 1/N$.

#

The idea of the resampling procedure is to remove particles with very small weights and duplicate particles with large weights. Although the theoretical distribution represented by the weighted set of samples does not change, resampling introduces additional variance to estimates. This variance introduced by the resampling procedure can be reduced by proper choice of the resampling method. The *stratified resampling* algorithm (Kitagawa, 1996) is optimal in terms of variance.

Adding a resampling step to the sequential importance sampling algorithm leads to *sequential importance resampling (SIR)*[2] (Gordon et al., 1993; Kitagawa, 1996; Doucet et al., 2001; Ristic et al., 2004), which is the algorithm usually referred to as the *particle filter*. In SIR, resampling is not usually performed at every time step, but only when it is actually needed. One way of implementing this is to do resampling on every nth step, where n is some predefined constant. This method has the advantage that it is unbiased. Another way, which is used here, is *adaptive resampling*. In this method, the "effective" number of particles, which is estimated from the variance of the particle weights (Liu and Chen, 1995), is used for monitoring the need for resampling. The estimate for the effective number of particles can be computed as:

[2] *Sequential importance resampling* is also often referred to as *sampling importance resampling or* sequential importance sampling resampling.

$$n_{\text{eff}} \approx \frac{1}{\sum_{i=1}^{N} \left(w_k^{(i)} \right)^2}, \tag{7.27}$$

where $w_k^{(i)}$ is the normalized weight of particle i at the time step k (Liu and Chen, 1995). Resampling is performed when the effective number of particles is significantly less than the total number of particles, for example, $n_{\text{eff}} < N/10$, where N is the total number of particles.

Algorithm 7.4 (Sequential importance resampling) *The sequential importance resampling (SIR) algorithm, which is also called the particle filter (PF), is the following.*

- *Draw N samples $\mathbf{x}_0^{(i)}$ from the prior*

$$\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0), \qquad i = 1, \ldots, N, \tag{7.28}$$

  and set $w_0^{(i)} = 1/N$, for all $i = 1, \ldots, N$.
- *For each $k = 1, \ldots, T$ do the following:*

1 Draw samples $\mathbf{x}_k^{(i)}$ from the importance distributions

$$\mathbf{x}_k^{(i)} \sim \pi(\mathbf{x}_k \mid \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_{1:k}), \qquad i = 1, \ldots, N. \tag{7.29}$$

#

2 *Calculate new weights according to*

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)})\, p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)})}{\pi(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_{1:k})} \tag{7.30}$$

and normalize them to sum to unity.

3 *If the effective number of particles* (7.27) *is too low, perform resampling.*

The SIR algorithm forms the following approximation to the filtering distribution:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}), \tag{7.31}$$

and thus the expectation of an arbitrary function $\mathbf{g}(\cdot)$ can be approximated as

$$\mathrm{E}[\mathbf{g}(\mathbf{x}_k) \mid \mathbf{y}_{1:k}] \approx \sum_{i=1}^{N} w_k^{(i)} \mathbf{g}(\mathbf{x}_k^{(i)}). \tag{7.32}$$

#

Performance of the SIR algorithm depends on the quality of the importance distribution $\pi(\cdot)$. The importance distribution should be in such a functional form that we can easily draw samples from it and that it is possible to evaluate the probability densities of the sample points. *The optimal importance distribution* in terms of variance (see, e.g., Doucet et al., 2001; Ristic et al., 2004) is

The bootstrap filter (Gordon et al., 1993) is a variation of SIR where the dynamic model $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ is used as the importance distribution. This makes the implementation of the algorithm very easy, but due to the inefficiency of the importance distribution it may require a very large number of Monte Carlo samples for accurate estimation results. In the bootstrap filter the resampling is normally done at each time step.

Algorithm 7.5 (Bootstrap filter) *The bootstrap filter algorithm is as follows.*

1 *Draw a new point* $\mathbf{x}_k^{(i)}$ *for each point in the sample set* $\{\mathbf{x}_{k-1}^{(i)} : i = 1, \ldots, N\}$ *from the dynamic model:*

$$\mathbf{x}_k^{(i)} \sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^{(i)}), \qquad i = 1, \ldots, N. \tag{7.34}$$

#

2 *Calculate the weights*

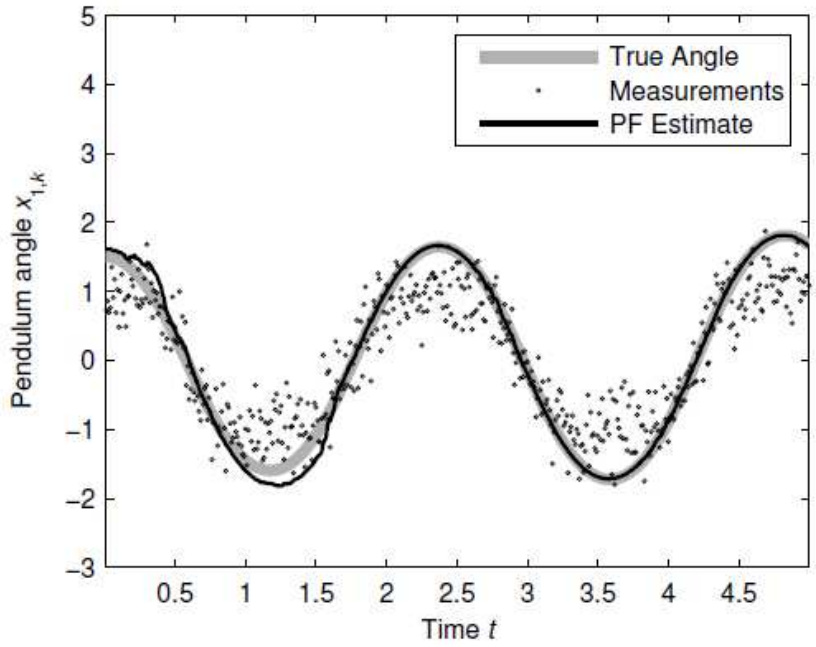
$$w_k^{(i)} \propto p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)}), \qquad i = 1, \ldots, N, \tag{7.35}$$

and normalize them to sum to unity.

3 *Do resampling.*

Example

The result of the bootstrap filter with 10000 particles in the pendulum model is shown:



#

8. BAYESIAN SMOOTHING EQUATIONS AND EXACT SOLUTIONS

8.1 Bayesian Smoothing Equations

The purpose of *Bayesian smoothing*[1] is to compute the marginal posterior distribution of the state $\mathbf{x}_k$ at the time step k after receiving the measurements up to a time step T, where $T > k$:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:T}). \tag{8.1}$$

The difference between filters and smoothers is that *the Bayesian filter* computes its estimates using only the measurements obtained before and at the time step k, but *the Bayesian smoother* uses also the future measurements for computing its estimates. After obtaining the filtering posterior state distributions, the following theorem gives the equations for computing the marginal posterior distributions for each time step conditionally on all the measurements up to the time step T.

Theorem 8.1 (Bayesian optimal smoothing equations) *The backward recursive equations (the* Bayesian smoother*) for computing the* smoothed distributions $p(\mathbf{x}_k \mid \mathbf{y}_{1:T})$ *for any* $k < T$ *are given by the following*

[1] This definition actually applies to the fixed-interval type of smoothing.

Bayesian (fixed-interval) smoothing equations *(Kitagawa, 1987):*

$$p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k}) = \int p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k})\, \mathrm{d}\mathbf{x}_k,$$

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:T}) = p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \int \left[\frac{p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)\, p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T})}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})} \right] \mathrm{d}\mathbf{x}_{k+1}, \tag{8.2}$$

where $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$ *is the filtering distribution of the time step* k*. Note that the term* $p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})$ *is simply the predicted distribution of time step* $k+1$*. The integrations are replaced by summations if some of the state components are discrete.*

Proof Due to the Markov properties the state $\mathbf{x}_k$ is independent of $\mathbf{y}_{k+1:T}$ given $\mathbf{x}_{k+1}$, which gives $p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:T}) = p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k})$. By using *Bayes' rule* the distribution of $\mathbf{x}_k$ given $\mathbf{x}_{k+1}$ and $\mathbf{y}_{1:T}$ can be expressed as

$$\begin{aligned} p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:T}) &= p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k}) \\ &= \frac{p(\mathbf{x}_k, \mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})} \end{aligned}$$

#

The joint distribution of $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ given $\mathbf{y}_{1:T}$ can be now computed as

$$\begin{aligned} p(\mathbf{x}_k, \mathbf{x}_{k+1} \mid \mathbf{y}_{1:T}) &= p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:T})\, p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T}) \\ &= p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k})\, p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T}) \\ &= \frac{p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k})\, p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T})}{p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:k})}, \end{aligned} \tag{8.4}$$

where $p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T})$ is the smoothed distribution of the time step $k+1$. The marginal distribution of $\mathbf{x}_k$ given $\mathbf{y}_{1:T}$ is given by integration (or summation) over $\mathbf{x}_{k+1}$ in Equation (8.4), which gives the desired result. □

8.2.2 Rauch-Tung-Striebel Smoother

The *Rauch–Tung–Striebel smoother* (RTSS, Rauch et al., 1965), which is also called the *Kalman smoother*, can be used for computing the closed form smoothing solution

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:T}) = \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^s, \mathbf{P}_k^s) \tag{8.5}$$

to the linear filtering model (4.17). The difference from the solution computed by the *Kalman filter* is that the smoothed solution is conditional on the whole measurement data $\mathbf{y}_{1:T}$, while the filtering solution is conditional only on the measurements obtained before and at the time step k, that is, on the measurements $\mathbf{y}_{1:k}$.

Theorem 8.2 (RTS smoother) *The backward recursion equations for the (fixed interval) Rauch–Tung–Striebel smoother are given as*

$$\begin{aligned}
\mathbf{m}_{k+1}^{-} &= \mathbf{A}_k\, \mathbf{m}_k, \\
\mathbf{P}_{k+1}^{-} &= \mathbf{A}_k\, \mathbf{P}_k\, \mathbf{A}_k^{\mathsf{T}} + \mathbf{Q}_k, \\
\mathbf{G}_k &= \mathbf{P}_k\, \mathbf{A}_k^{\mathsf{T}}\, [\mathbf{P}_{k+1}^{-}]^{-1}, \\
\mathbf{m}_k^{\mathrm{s}} &= \mathbf{m}_k + \mathbf{G}_k\, [\mathbf{m}_{k+1}^{\mathrm{s}} - \mathbf{m}_{k+1}^{-}], \\
\mathbf{P}_k^{\mathrm{s}} &= \mathbf{P}_k + \mathbf{G}_k\, [\mathbf{P}_{k+1}^{\mathrm{s}} - \mathbf{P}_{k+1}^{-}]\, \mathbf{G}_k^{\mathsf{T}},
\end{aligned} \tag{8.6}$$

where $\mathbf{m}_k$ *and* $\mathbf{P}_k$ *are the mean and covariance computed by the Kalman filter. The recursion is started from the last time step* T*, with* $\mathbf{m}_T^{\mathrm{s}} = \mathbf{m}_T$ *and* $\mathbf{P}_T^{\mathrm{s}} = \mathbf{P}_T$*. Note that the first two of the equations are simply the Kalman filter prediction equations.*

Proof Similarly to the Kalman filter case, by Lemma A.1, the joint distribution of $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ given $\mathbf{y}_{1:k}$ is

$$\begin{aligned}
p(\mathbf{x}_k, \mathbf{x}_{k+1} \mid \mathbf{y}_{1:k}) &= p(\mathbf{x}_{k+1} \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \\
&= \mathrm{N}(\mathbf{x}_{k+1} \mid \mathbf{A}_k\, \mathbf{x}_k, \mathbf{Q}_k)\, \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k) \\
&= \mathrm{N}\left(\begin{pmatrix} \mathbf{x}_k \\ \mathbf{x}_{k+1} \end{pmatrix} \,\middle|\, \tilde{\mathbf{m}}_1, \tilde{\mathbf{P}}_1 \right),
\end{aligned} \tag{8.7}$$

#

where

$$\tilde{\mathbf{m}}_1 = \begin{pmatrix} \mathbf{m}_k \\ \mathbf{A}_k\, \mathbf{m}_k \end{pmatrix}, \qquad \tilde{\mathbf{P}}_1 = \begin{pmatrix} \mathbf{P}_k & \mathbf{P}_k\, \mathbf{A}_k^\mathsf{T} \\ \mathbf{A}_k\, \mathbf{P}_k & \mathbf{A}_k\, \mathbf{P}_k\, \mathbf{A}_k^\mathsf{T} + \mathbf{Q}_k \end{pmatrix}. \tag{8.8}$$

Due to the Markov property of the states we have

$$p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:T}) = p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k}), \tag{8.9}$$

and thus by Lemma A.2 we get the conditional distribution

$$\begin{aligned} p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:T}) &= p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:k}) \\ &= \mathrm{N}(\mathbf{x}_k \mid \tilde{\mathbf{m}}_2, \tilde{\mathbf{P}}_2), \end{aligned} \tag{8.10}$$

where

$$\begin{aligned} \mathbf{G}_k &= \mathbf{P}_k\, \mathbf{A}_k^\mathsf{T}\, (\mathbf{A}_k\, \mathbf{P}_k\, \mathbf{A}_k^\mathsf{T} + \mathbf{Q}_k)^{-1} \\ \tilde{\mathbf{m}}_2 &= \mathbf{m}_k + \mathbf{G}_k\, (\mathbf{x}_{k+1} - \mathbf{A}_k\, \mathbf{m}_k) \\ \tilde{\mathbf{P}}_2 &= \mathbf{P}_k - \mathbf{G}_k\, (\mathbf{A}_k\, \mathbf{P}_k\, \mathbf{A}_k^\mathsf{T} + \mathbf{Q}_k)\, \mathbf{G}_k^\mathsf{T}. \end{aligned} \tag{8.11}$$

The joint distribution of $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ given all the data is

$$\begin{aligned} p(\mathbf{x}_{k+1}, \mathbf{x}_k \mid \mathbf{y}_{1:T}) &= p(\mathbf{x}_k \mid \mathbf{x}_{k+1}, \mathbf{y}_{1:T})\, p(\mathbf{x}_{k+1} \mid \mathbf{y}_{1:T}) \\ &= \mathrm{N}(\mathbf{x}_k \mid \tilde{\mathbf{m}}_2, \tilde{\mathbf{P}}_2)\ \mathrm{N}(\mathbf{x}_{k+1} \mid \mathbf{m}_{k+1}^\mathrm{s}, \mathbf{P}_{k+1}^\mathrm{s}) \\ &= \mathrm{N}\left(\begin{pmatrix} \mathbf{x}_{k+1} \\ \mathbf{x}_k \end{pmatrix} \middle| \, \tilde{\mathbf{m}}_3, \tilde{\mathbf{P}}_3 \right), \end{aligned} \tag{8.12}$$

#

where

$$\tilde{\mathbf{m}}_3 = \begin{pmatrix} \mathbf{m}^s_{k+1} \\ \mathbf{m}_k + \mathbf{G}_k\,(\mathbf{m}^s_{k+1} - \mathbf{A}_k\,\mathbf{m}_k) \end{pmatrix},$$
$$\tilde{\mathbf{P}}_3 = \begin{pmatrix} \mathbf{P}^s_{k+1} & \mathbf{P}^s_{k+1}\,\mathbf{G}_k^\mathsf{T} \\ \mathbf{G}_k\,\mathbf{P}^s_{k+1} & \mathbf{G}_k\,\mathbf{P}^s_{k+1}\,\mathbf{G}_k^\mathsf{T} + \tilde{\mathbf{P}}_2 \end{pmatrix}. \tag{8.13}$$

Thus by Lemma A.2, the marginal distribution of $\mathbf{x}_k$ is given as

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:T}) = \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}^s_k, \mathbf{P}^s_k), \tag{8.14}$$

where

$$\mathbf{m}^s_k = \mathbf{m}_k + \mathbf{G}_k\,(\mathbf{m}^s_{k+1} - \mathbf{A}_k\,\mathbf{m}_k),$$
$$\mathbf{P}^s_k = \mathbf{P}_k + \mathbf{G}_k\,(\mathbf{P}^s_{k+1} - \mathbf{A}_k\,\mathbf{P}_k\,\mathbf{A}_k^\mathsf{T} - \mathbf{Q}_k)\,\mathbf{G}_k^\mathsf{T}. \tag{8.15}$$

□

Example 8.1 (RTS smoother for Gaussian random walk) *The RTS smoother for the random walk model given in Example 4.1 is given by the equations*

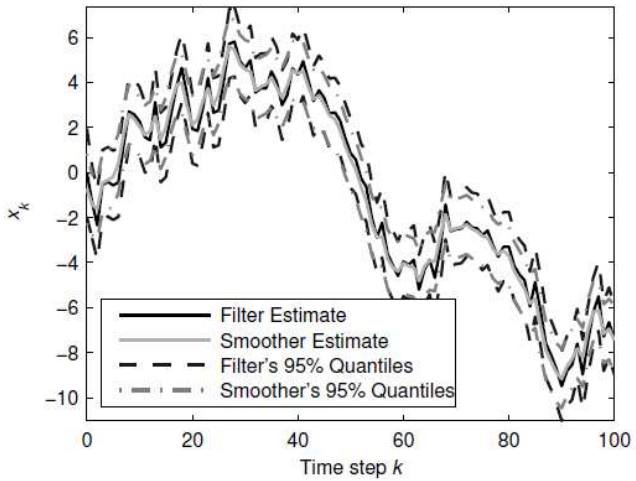
$$m^-_{k+1} = m_k,$$
$$P^-_{k+1} = P_k + Q,$$

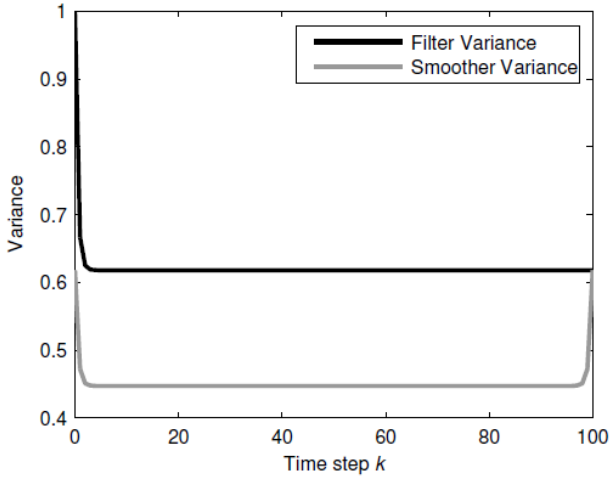
#

$$m_k^s = m_k + \frac{P_k}{P_{k+1}^-}(m_{k+1}^s - m_{k+1}^-),$$

$$P_k^s = P_k + \left(\frac{P_k}{P_{k+1}^-}\right)^2 [P_{k+1}^s - P_{k+1}^-], \qquad (8.16)$$

where m_k and P_k are the updated mean and covariance from the Kalman filter in Example 4.2. The result of applying the smoother to simulated data is shown in Figure 8.1. The evolution of the filter and smoother variances is illustrated in Figure 8.2.





#

Exercises

8.1 Derive the linear RTS smoother for the non-zero-mean noise model in Exercise 4.1.

8.2 Write down the Bayesian smoothing equations for finite-state HMM models described in Exercise 4.2 assuming that the filtering distributions $p(x_k \mid \mathbf{y}_{1:k})$ have already been computed.

APPENDIX: Cholesky Factorization

The Cholesky factor of the symmetric positive definite matrix $\mathbf{P}$ is a lower triangular matrix $\mathbf{A}$ such that

$$\mathbf{P} = \mathbf{A}\,\mathbf{A}^{\mathsf{T}}. \tag{A.6}$$

The matrix $\mathbf{A}$ can be computed by the Cholesky factorization algorithm (see, e.g., Golub and van Loan, 1996) presented below.

#

Algorithm A.1 (Cholesky factorization) *The Cholesky factor* $\mathbf{A}$ *of the matrix* $\mathbf{P} \in \mathbb{R}^{n \times n}$ *can be computed as follows:*

procedure CHOL($\mathbf{P}$)
  for $i \leftarrow 1 \ldots n$ ***do***
    $A_{ii} = \sqrt{P_{ii} - \sum_{k<i} A_{ik}^2}$
    for $j \leftarrow i + 1 \ldots n$ ***do***
      $A_{ji} = \left(P_{ji} - \sum_{k<i} A_{jk}\, A_{ik}\right) / A_{ii}$
    end for
  end for
  return $\mathbf{A}$
end procedure