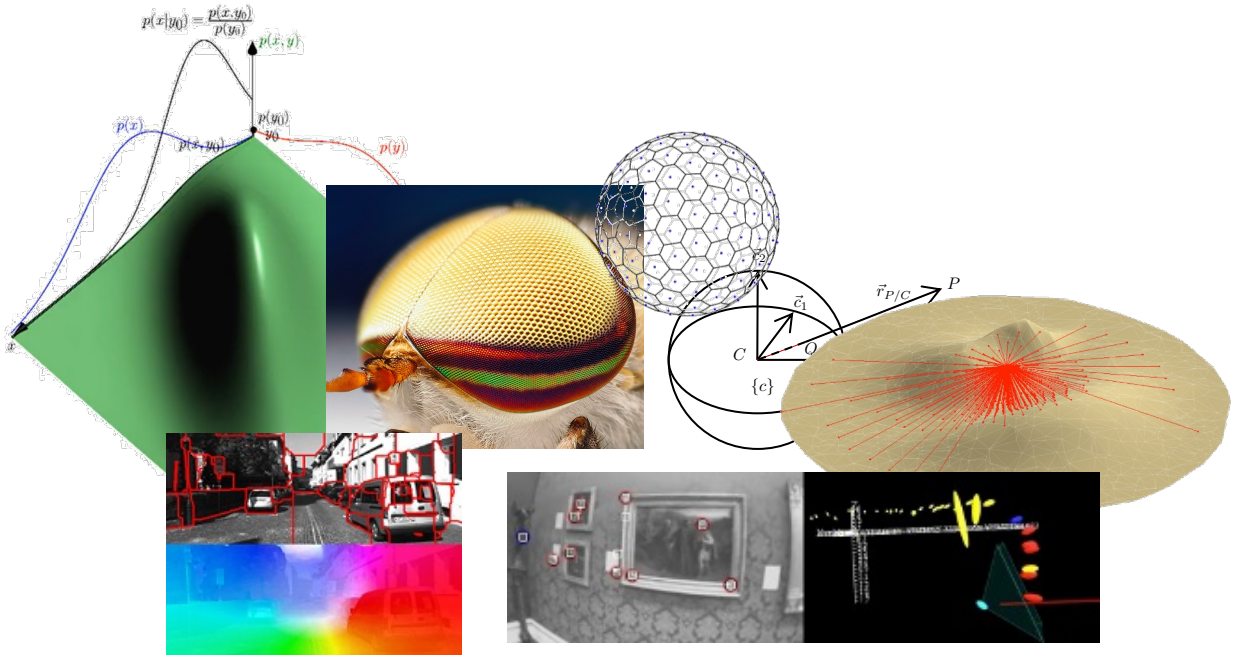
MCHA4400

Vision-based navigation



Dr Chris Renton

School of Engineering

Distributions

Discrete variables

- **Cumulative distribution function** (cdf), is the probability of **X** being *equal to or below* some value **x**.

$$F(\mathbf{x}) = P(\mathbf{X} \leq \mathbf{x})$$

- **Probability mass function** (pmf), is the probability that the discrete variable **X** is equal to a particular value **x**.

$$P(\mathbf{x}) = P(\mathbf{X} = \mathbf{x})$$

Continuous variables

Continuous distribution

- **Probability density function** (pdf), $p(\mathbf{x})$ does not indicate probability, but probability *density*, related to the cdf by

$$F(\mathbf{x}) = P(\mathbf{X} \leq \mathbf{x}) = \int_{-\infty}^{\mathbf{x}} p(\mathbf{z})\,\mathrm{d}\mathbf{z}$$

- It does not make sense to talk about $P(\mathbf{X} = \mathbf{x})$ for continuous distributions, but instead we can ask what the probability is that $\mathbf{X}$ lies within some region

$$P(\mathbf{X} \in \mathcal{X}) = \int_{\mathcal{X}} p(\mathbf{z})\,\mathrm{d}\mathbf{z}$$

Conditional density & Marginalisation

For continuous distributions, given some data D, the conditional cdf is related to the conditional pdf by

$$F(\mathbf{x}|D) = P(\mathbf{X} \leq \mathbf{x}|D) = \frac{P(\mathbf{X} \leq \mathbf{x}, D)}{P(D)} = \int_{-\infty}^{\mathbf{x}} p(\mathbf{z}|D)\,\mathrm{d}\mathbf{z}$$

The pdf $p(\mathbf{x})$ is related to the joint pdf $p(\mathbf{x},\mathbf{y})$ and conditional pdf $p(\mathbf{x}\,|\,\mathbf{y})$ by

$$p(\mathbf{x}) = \int_{-\infty}^{\infty} p(\mathbf{x}, \mathbf{y})\,\mathrm{d}\mathbf{y} = \int_{-\infty}^{\infty} p(\mathbf{x}|\mathbf{y})p(\mathbf{y})\,\mathrm{d}\mathbf{y}$$

Transformation of RVs

Given two random variables, $\mathbf{X}$, $\mathbf{Y}$, their pdfs, $p_{\mathbf{X}}(\bullet)$, $p_{\mathbf{Y}}(\bullet)$ and a functional relation $\mathbf{y} = \mathbf{f}(\mathbf{x})$, where $\mathbf{f}$ is monotonic, find the pdf of $\mathbf{Y}$.

$$P(\mathbf{x} < \mathbf{X} \leq \mathbf{x} + \mathrm{d}\mathbf{x}) = P(\mathbf{y} < \mathbf{Y} \leq \mathbf{y} + \mathrm{d}\mathbf{y})$$

$$\int_{\mathbf{x}}^{\mathbf{x}+\mathrm{d}\mathbf{x}} p_{\mathbf{X}}(\mathbf{z})\,\mathrm{d}\mathbf{z} = \begin{cases} \int_{\mathbf{y}}^{\mathbf{y}+\mathrm{d}\mathbf{y}} p_{\mathbf{Y}}(\mathbf{z})\,\mathrm{d}\mathbf{z} & \text{if } \mathrm{d}\mathbf{y} > 0 \\ -\int_{\mathbf{y}}^{\mathbf{y}+\mathrm{d}\mathbf{y}} p_{\mathbf{Y}}(\mathbf{z})\,\mathrm{d}\mathbf{z} & \text{if } \mathrm{d}\mathbf{y} < 0 \end{cases}$$

$$p_{\mathbf{X}}(\mathbf{x})\,\mathrm{d}\mathbf{x} = p_{\mathbf{Y}}(\mathbf{y})|\mathrm{d}\mathbf{y}|$$

$$p_{\mathbf{Y}}(\mathbf{y}) = \left|\frac{\partial \mathbf{x}}{\partial \mathbf{y}}\right| p_{\mathbf{X}}(\mathbf{x}) = \left|\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right|^{-1} p_{\mathbf{X}}(\mathbf{x})$$

$$p_{\mathbf{Y}}(\mathbf{y}) = \left|\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}\right|^{-1} p_{\mathbf{X}}(\mathbf{f}^{-1}(\mathbf{y})) \qquad |\cdot| = \mathrm{abs}\,\det(\cdot)$$

Note: The derivation above is illustrative, not rigorous

Transformation of RVs

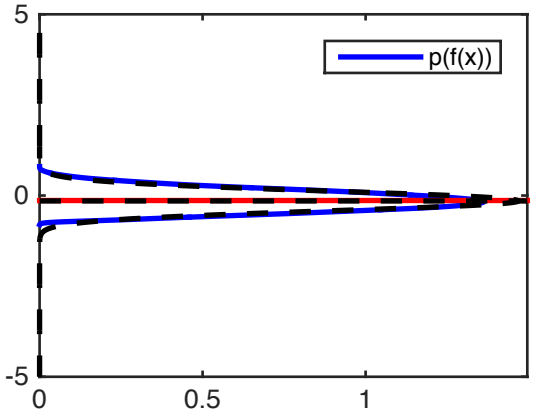
If the function $\mathbf{f}$ is not monotonic, then there is no unique inverse. In this case, we can transform the RV as follows:

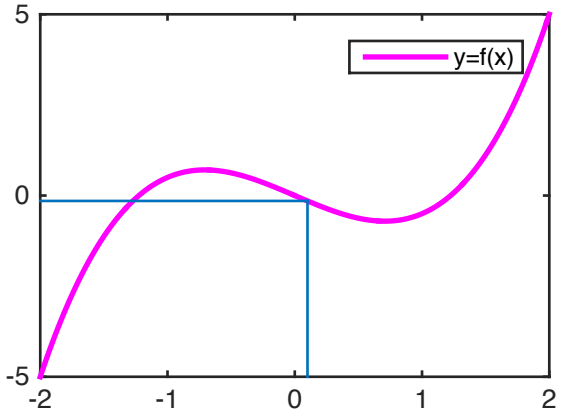
$$p_{\mathbf{Y}}(\mathbf{y}) = \sum_{\mathbf{x} \in \mathbf{f}^{-1}(\mathbf{y})} \left| \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right|^{-1} p_{\mathbf{X}}(\mathbf{x}) \qquad |\cdot| = \operatorname{abs} \det(\cdot)$$

where we treat the inverse of $\mathbf{f}$ as the set of solutions for each $\mathbf{y}$,

$$\mathbf{f}^{-1}(\mathbf{y}) = \{\mathbf{x} \,|\, \mathbf{f}(\mathbf{x}) = \mathbf{y}\}$$

Example

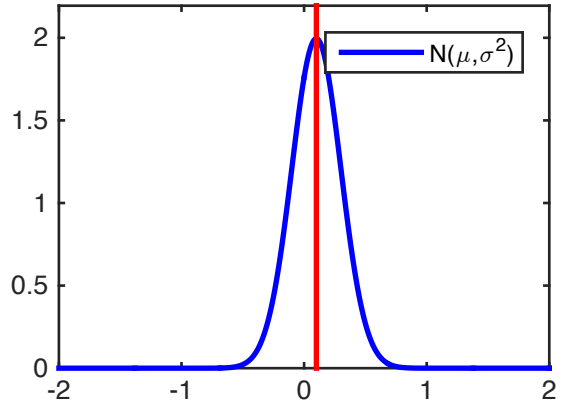




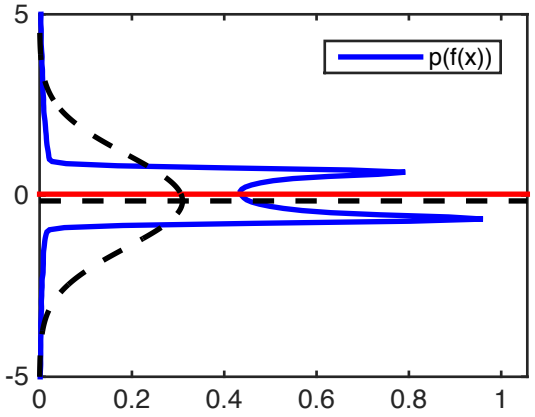
$$y = f(x) = x^3 - \frac{3}{2}x$$

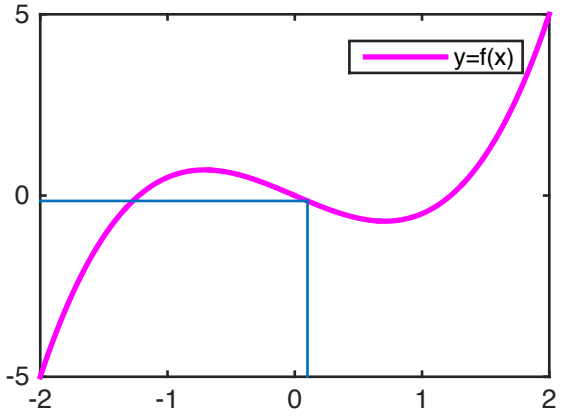
$$p_Y(y) = \sum_{x \in f^{-1}(y)} \left| \frac{\partial f(x)}{\partial x} \right|^{-1} p_X(x)$$

$$f^{-1}(y) = \{x \mid f(x) = y\}$$



Example

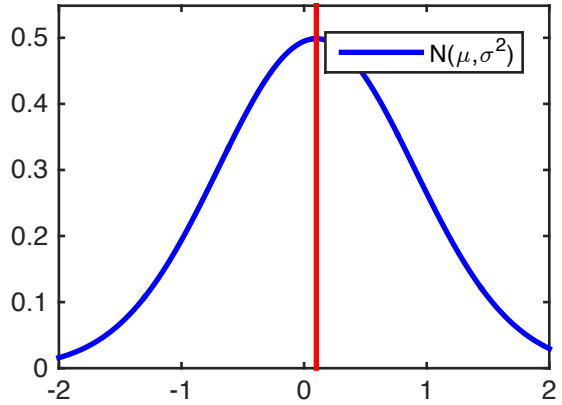




$$y = f(x) = x^3 - \frac{3}{2}x$$

$$p_Y(y) = \sum_{x \in f^{-1}(y)} \left| \frac{\partial f(x)}{\partial x} \right|^{-1} p_X(x)$$

$$f^{-1}(y) = \{x \mid f(x) = y\}$$



Sum of RVs

Given two random variables, **X**, **Y**, and their joint pdf, $p(\mathbf{x}, \mathbf{y})$, the pdf of the sum $\mathbf{Z} = \mathbf{X} + \mathbf{Y}$ is given by

$$p_{\mathbf{Z}}(\mathbf{z}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\mathbf{z}-\mathbf{y}} p(\mathbf{x}, \mathbf{y})\, \mathrm{d}\mathbf{x}\, \mathrm{d}\mathbf{y} = \int_{-\infty}^{\infty} \int_{-\infty}^{\mathbf{z}-\mathbf{x}} p(\mathbf{x}, \mathbf{y})\, \mathrm{d}\mathbf{y}\, \mathrm{d}\mathbf{x}$$

If **X** and **Y** are independent, then $p(\mathbf{x}, \mathbf{y}) = p_{\mathbf{X}}(\mathbf{x})\, p_{\mathbf{Y}}(\mathbf{y})$ and the above simplifies to the convolution of the functions $p_{\mathbf{X}}(\bullet)$ and $p_{\mathbf{Y}}(\bullet)$

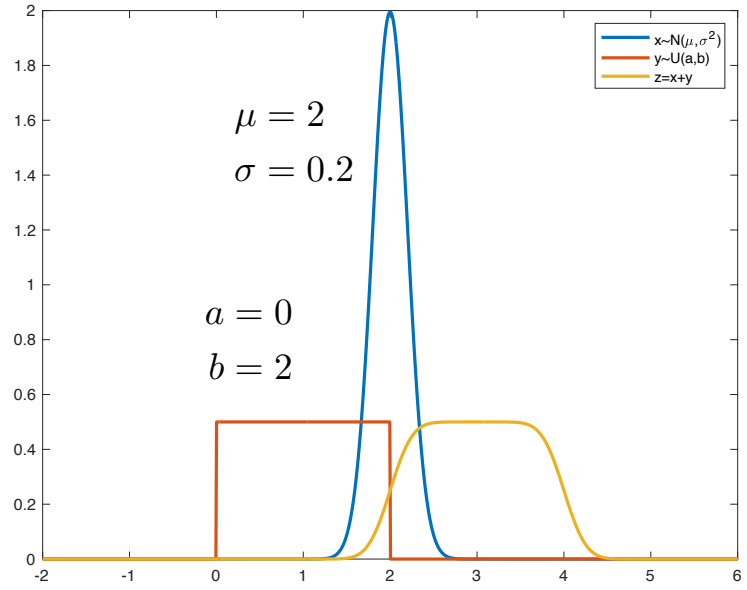
$$p_{\mathbf{Z}}(\mathbf{z}) = (p_{\mathbf{X}} * p_{\mathbf{Y}})(\mathbf{z}) = \int_{-\infty}^{\infty} p_{\mathbf{X}}(\mathbf{z} - \mathbf{y})\, p_{\mathbf{Y}}(\mathbf{y})\, \mathrm{d}\mathbf{y} = \int_{-\infty}^{\infty} p_{\mathbf{X}}(\mathbf{x})\, p_{\mathbf{Y}}(\mathbf{z} - \mathbf{x})\, \mathrm{d}\mathbf{x}$$

Example

Let $z = x + y$, where x and y are independent RVs with $x \sim \mathcal{N}(\mu,\sigma^2)$ and $y \sim \mathcal{U}(a,b)$.

Then,

$$
\begin{aligned}
p_Z(z) &= (p_X * p_Y)(z) \\
&= \int_{-\infty}^{\infty} p_X(z-y)\, p_Y(y)\, \mathrm{d}y \\
&= \int_{-\infty}^{\infty} \mathcal{N}(z-y;\mu,\sigma^2)\, \mathcal{U}(y;a,b)\, \mathrm{d}y \\
&= \frac{1}{b-a} \int_{a}^{b} \mathcal{N}(z-y;\mu,\sigma^2)\, \mathrm{d}y
\end{aligned}
$$



Expectation

The *expectation functional* maps a function and a given pdf into a value

$$\mathrm{E}_p[\mathbf{f}(\cdot)] = \int_{\mathcal{X}} \mathbf{f}(\mathbf{x}) p(\mathbf{x})\, \mathrm{d}\mathbf{x}$$

If **f** = **id**, i.e., **f**(**x**) = **x**, then we map from a pdf to a value—the *expected value*,

$$\boldsymbol{\mu}_{\mathbf{x}} = \mathrm{E}_p[\mathbf{x}] = \int_{\mathcal{X}} \mathbf{x}\, p(\mathbf{x})\, \mathrm{d}\mathbf{x}$$

The expected value is not necessarily what we would expect to see when we run a particular experiment!

Covariance

If we set $\mathbf{f}(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathrm{E}[\mathbf{x}])\,(\mathbf{y} - \mathrm{E}[\mathbf{y}])^{\mathsf{T}}$, then we define the **cross covariance** between **x** and **y** as

$$\mathbf{P}_{\mathbf{xy}} = \mathrm{E}[(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}})^{\mathsf{T}}] = \mathrm{E}[\mathbf{x}\mathbf{y}^{\mathsf{T}}] - \boldsymbol{\mu}_{\mathbf{x}}\boldsymbol{\mu}_{\mathbf{y}}^{\mathsf{T}}$$

We are often interested in the **covariance matrix** for a particular **x**,

$$\mathbf{P}_{\mathbf{x}} = \mathrm{E}[(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})^{\mathsf{T}}] = \mathrm{E}[\mathbf{x}\mathbf{x}^{\mathsf{T}}] - \boldsymbol{\mu}_{\mathbf{x}}\boldsymbol{\mu}_{\mathbf{x}}^{\mathsf{T}}$$

We can say that **x**, has mean $\boldsymbol{\mu}_{\mathbf{x}}$ and covariance $\mathbf{P}_{\mathbf{x}}$ with the notation

$$\mathbf{x} \sim (\boldsymbol{\mu}_{\mathbf{x}}, \mathbf{P}_{\mathbf{x}})$$

If *x* is scalar, then the covariance reduces to the **variance**, equal to the square of the **standard deviation**.

$$x \sim (\mu_x, \sigma_x^2)$$

Estimating mean and covariance

We can estimate the expected value of **x** by taking the **sample mean** over a large number of experiments

$$\hat{\boldsymbol{\mu}}_{\mathbf{x}} = \frac{1}{N}\sum_{k=1}^{N}\mathbf{x}_k$$

The **sample covariance** is given by

$$\hat{\mathbf{P}}_{\mathbf{x}} = \frac{1}{N-1}\sum_{k=1}^{N}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_{\mathbf{x}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_{\mathbf{x}})^{\mathsf{T}}$$

The factor $N-1$, known as Bessel's correction, arises because there is bias introduced by using the sample mean. If the expected value of **x** was known exactly, then

$$\hat{\mathbf{P}}_{\mathbf{x}} = \frac{1}{N}\sum_{k=1}^{N}(\mathbf{x}_k - \boldsymbol{\mu}_{\mathbf{x}})(\mathbf{x}_k - \boldsymbol{\mu}_{\mathbf{x}})^{\mathsf{T}}$$

Bayesian Credible Region

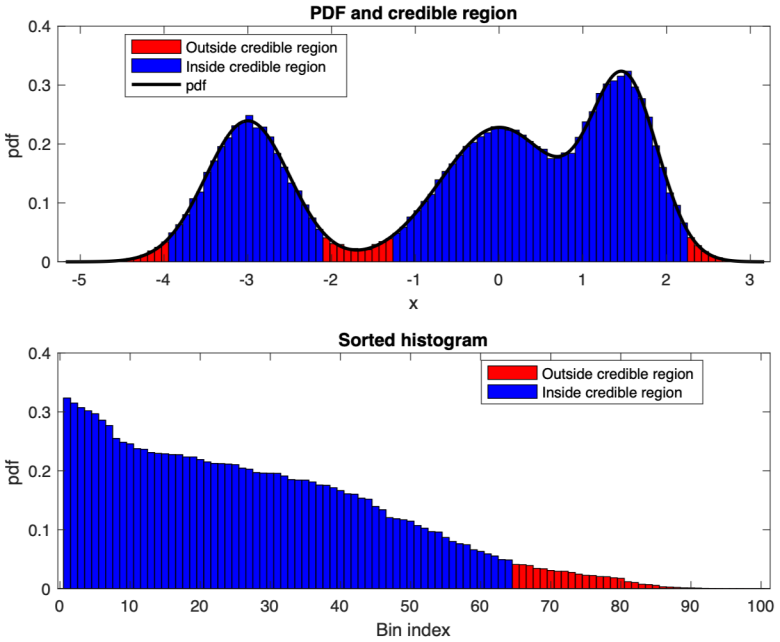
In some applications, it is useful to determine a set of likely values instead of just a single point estimate.

The *Bayesian Credible Region* is given by the set

$$\mathcal{R} = \{\mathbf{x} \in \mathcal{X} \mid p(\mathbf{x}) \geq t\}$$

where t is the unique solution to

$$\int_{p(\mathbf{x}) \geq t} p(\mathbf{x})\, \mathrm{d}\mathbf{x} = c$$



The threshold can be obtained by a direct search, or by computing a histogram and sorting the bins.

Typically, c is taken as 95% or 99.7%, which correspond to the 2σ and 3σ confidence regions of a Gaussian distribution, respectively.

Gaussian distribution

Gaussian pdf and RVs

The **probability density function** (pdf) of a multivariate Gaussian RV

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{P})$$

is given by

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{P}) = \frac{1}{\sqrt{\det 2\pi\mathbf{P}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\mathbf{P}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

A Gaussian pdf is entirely parameterised by its mean $\boldsymbol{\mu}$ and covariance $\mathbf{P}$.

Gaussian RVs and pdfs have some very useful properties:

- Affine transformations of Gaussian RVs are also Gaussian RVs
- Marginal and conditional densities of joint Gaussian pdfs are also Gaussian pdfs

Affine transformation of Gaussian RVs

Consider a Gaussian RV with $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{P})$. Find the pdf of $\mathbf{y} = \mathbf{A}\,\mathbf{x} + \mathbf{b}$. Then, $\mathbf{x} = \mathbf{f}(\mathbf{y}) = \mathbf{A}^{-1}(\mathbf{y} - \mathbf{b})$ and

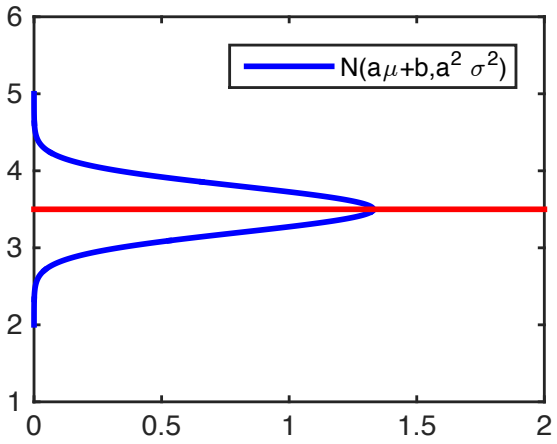
$$|\cdot| = \operatorname{abs}\det(\cdot)$$

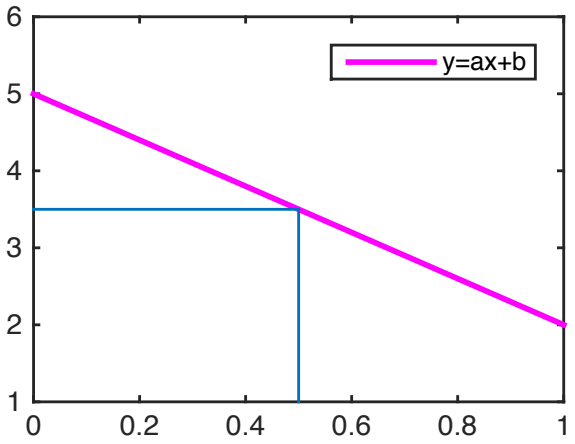
$$\begin{aligned} p(\mathbf{y}) &= \left|\frac{\partial \mathbf{f}(\mathbf{y})}{\partial \mathbf{y}}\right| p(\mathbf{f}(\mathbf{y})) \\ &= \left|\mathbf{A}^{-1}\right| p(\mathbf{A}^{-1}(\mathbf{y}-\mathbf{b})) \\ &= \frac{|\mathbf{A}^{-1}|}{\sqrt{|2\pi\mathbf{P}|}} \exp\left(-\frac{1}{2}(\mathbf{A}^{-1}(\mathbf{y}-\mathbf{b})-\boldsymbol{\mu})^{\mathsf{T}}\mathbf{P}^{-1}(\ldots)\right) \\ &= \frac{1}{\sqrt{|2\pi\mathbf{A}\mathbf{P}\mathbf{A}^{\mathsf{T}}|}} \exp\left(-\frac{1}{2}(\mathbf{y}-\mathbf{b}-\mathbf{A}\boldsymbol{\mu})^{\mathsf{T}}(\mathbf{A}\mathbf{P}\mathbf{A}^{\mathsf{T}})^{-1}(\ldots)\right) \end{aligned}$$

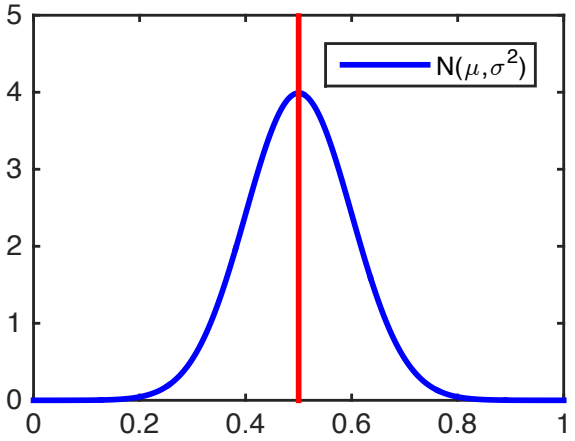
$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y};\ \mathbf{A}\boldsymbol{\mu} + \mathbf{b},\ \mathbf{A}\mathbf{P}\mathbf{A}^{\mathsf{T}})$$

This result also holds for non-invertible **A**, *(e.g., rank-deficient square, tall or wide).*

Affine transformation of Gaussian RVs







Sum of independent Gaussian RVs

The pdf of the sum of two *independent* Gaussian RVs, $\mathbf{z} = \mathbf{x} + \mathbf{y}$, where $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, \mathbf{P}_\mathbf{x})$ and $\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{y}, \mathbf{P}_\mathbf{y})$, can be computed by considering an affine transformation of the joint distribution $p(\mathbf{x}, \mathbf{y})$

$$p(\mathbf{x}, \mathbf{y}) = p\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}_\mathbf{x}\\ \boldsymbol{\mu}_\mathbf{y}\end{bmatrix}, \begin{bmatrix}\mathbf{P}_\mathbf{x} & \mathbf{0}\\ \mathbf{0} & \mathbf{P}_\mathbf{y}\end{bmatrix}\right)$$

$$p(\mathbf{z}) = \mathcal{N}\left(\mathbf{z}; \begin{bmatrix}\mathbf{I} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{\mu}_\mathbf{x}\\ \boldsymbol{\mu}_\mathbf{y}\end{bmatrix}, \begin{bmatrix}\mathbf{I} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{P}_\mathbf{x} & \mathbf{0}\\ \mathbf{0} & \mathbf{P}_\mathbf{y}\end{bmatrix}\begin{bmatrix}\mathbf{I}\\ \mathbf{I}\end{bmatrix}\right)$$

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z};\, \boldsymbol{\mu}_\mathbf{x} + \boldsymbol{\mu}_\mathbf{y},\, \mathbf{P}_\mathbf{x} + \mathbf{P}_\mathbf{y}), \qquad \mathbf{z} = \mathbf{x} + \mathbf{y}$$

Adding Gaussian RVs preserves normality.

Note that we are summing the variables, not the pdfs!

$$p(\mathbf{x} + \mathbf{y}) \neq p(\mathbf{x}) + p(\mathbf{y})$$

The LHS is a pdf, the RHS is an unnormalised mixture.

Joint and marginal Gaussian densities

Consider the joint pdf $p(\mathbf{x}, \mathbf{y})$

$$p(\mathbf{x}, \mathbf{y}) = p\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}_\mathbf{x}\\ \boldsymbol{\mu}_\mathbf{y}\end{bmatrix}, \begin{bmatrix}\mathbf{P}_{\mathbf{xx}} & \mathbf{P}_{\mathbf{xy}}\\ \mathbf{P}_{\mathbf{yx}} & \mathbf{P}_{\mathbf{yy}}\end{bmatrix}\right)$$

The marginal densities $p(\mathbf{x})$ and $p(\mathbf{y})$ are given by

$$p(\mathbf{x}) = \int_{\mathcal{Y}} p(\mathbf{x}, \mathbf{y})\, \mathrm{d}\mathbf{y} = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_\mathbf{x}, \mathbf{P}_{\mathbf{xx}})$$

$$p(\mathbf{y}) = \int_{\mathcal{X}} p(\mathbf{x}, \mathbf{y})\, \mathrm{d}\mathbf{x} = \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_\mathbf{y}, \mathbf{P}_{\mathbf{yy}})$$

These results can be shown by applying linear transformations $\mathbf{C}=[\mathbf{I}, \mathbf{0}]$ *and* $\mathbf{C}=[\mathbf{0}, \mathbf{I}]$ *to select the* $\mathbf{x}$ *and* $\mathbf{y}$ *components, respectively, of the joint Gaussian RV.*

Joint and conditional Gaussian densities

Consider the joint pdf $p(\mathbf{x}, \mathbf{y})$

$$p(\mathbf{x},\mathbf{y}) = p\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}_\mathbf{x}\\ \boldsymbol{\mu}_\mathbf{y}\end{bmatrix}, \begin{bmatrix}\mathbf{P}_{\mathbf{xx}} & \mathbf{P}_{\mathbf{xy}}\\ \mathbf{P}_{\mathbf{yx}} & \mathbf{P}_{\mathbf{yy}}\end{bmatrix}\right)$$

The conditional density $p(\mathbf{x}|\mathbf{y})$ is given by

$$\begin{aligned} p(\mathbf{x}|\mathbf{y}) &= \frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{y})} = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}}, \mathbf{P}_{\mathbf{x}|\mathbf{y}}) \\ &= \mathcal{N}(\mathbf{x}; \underbrace{\boldsymbol{\mu}_\mathbf{x} + \mathbf{P}_{\mathbf{xy}}\mathbf{P}_{\mathbf{yy}}^{-1}(\mathbf{y}-\boldsymbol{\mu}_\mathbf{y})}_{\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}}=\mathrm{E}[\mathbf{x}|\mathbf{y}]}, \underbrace{\mathbf{P}_{\mathbf{xx}} - \mathbf{P}_{\mathbf{xy}}\mathbf{P}_{\mathbf{yy}}^{-1}\mathbf{P}_{\mathbf{yx}}}_{\mathbf{P}_{\mathbf{x}|\mathbf{y}}=\mathrm{cov}(\mathbf{x}|\mathbf{y})}) \end{aligned}$$

The division of a joint Gaussian by one of its marginal densities is equivalent to reducing a system of linear equations using the Schur complement of $\mathbf{P}_{\mathbf{yy}}$.

Bayesian state estimation

Bayesian state estimation

Let us consider a discrete-time nonlinear system described by

Process model: $\mathbf{x}_{k+1} = \mathbf{f}_k(\mathbf{x}_k, \mathbf{w}_k)$

Measurement model: $\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{v}_k)$

W.L.O.G., we can neglect the input as it can be considered a noiseless time-varying model parameter.

where $\mathbf{x}_k$ is the current state, $\mathbf{w}_k$ is the process noise, $\mathbf{y}_k$ is the measurement and $\mathbf{v}_k$ is the measurement noise. *The noises model uncertainty*: $\mathbf{w}_k$ *represents model uncertainty, and* $\mathbf{v}_k$ *represents measurement uncertainty.*

The goal of the Bayesian estimator is to compute the probability density of the state conditioned on the measurements $\mathbf{y}_1$, $\mathbf{y}_2$, …,$\mathbf{y}_k$, that is,

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = p(\mathbf{x}_k|\mathbf{y}_k, \mathbf{y}_{k-1}, \ldots, \mathbf{y}_2, \mathbf{y}_1)$$

where the notation $\mathbf{y}_{1:k}$ indicates all the measurements up to and including time k.

Bayesian state estimation

To make this feasible, we would like to update $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ recursively based on the estimate formed from the previous measurement $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$.

From the properties of joint and conditional pdfs, we have

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

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

Since $\mathbf{x}_k$ is entirely determined by $\mathbf{x}_{k-1}$ and $\mathbf{w}_{k-1}$, this simplifies to

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

Given the process model and the posterior for $k-1$, this tells us how to predict the current state from the posterior of the previous time step.

Bayesian state estimation

Let us return to finding $p(\mathbf{x}_k|\mathbf{y}_{1:k})$. By separating $\mathbf{y}_{1:k}$ into $\mathbf{y}_k$ and $\mathbf{y}_{1:k-1}$, applying Bayes rule on $\mathbf{x}_k$, $\mathbf{y}_k$, yields,

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

Since $\mathbf{y}_k$ is entirely determined by $\mathbf{x}_k$ and $\mathbf{v}_k$, this simplifies to

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{y}_{1:k-1})}{p(\mathbf{y}_k|\mathbf{y}_{1:k-1})}$$

The denominator can be written in terms of $p(\mathbf{y}_k|\mathbf{x}_k)$ and $p(\mathbf{x}_k|\mathbf{y}_{k-1})$,

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

Given the **measurement model** and the **prediction** from the previous slide, this tells us how to correct our prediction once we obtain the current measurement.

Bayesian state estimation

In summary, the recursive Bayesian state estimator consists of the steps:

1. Initialisation: Choose the prior as filtered density

$$p(\mathbf{x}_0) = p(\mathbf{x}_0|\mathbf{y}_{-\infty:0})$$

2. Repeat the following for $k = 1,2,\ldots$
   a. Prediction

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

   b. Correction

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

3. ???
4. Profit

Bayesian state estimation (alternate)

In summary, the recursive Bayesian state estimator consists of the steps:

1. Initialisation: Choose the prior as prediction density

$$p(\mathbf{x}_1) = p(\mathbf{x}_1|\mathbf{y}_{-\infty:0})$$

2. Repeat the following for $k = 1,2,\ldots$
   a. Correction

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

   b. Prediction

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

3. ???
4. Profit

Sun Tzu on Bayesian filtering

"In respect of military method, we have,
firstly, Measurement;
secondly, Estimation of quantity;
thirdly, Calculation;
fourthly, Balancing of chances;
fifthly, Victory."

The Art of War (IV:17)
Sun Tzu
≈ 200 BC ± 350

Continuous-time processes

Continuous-time processes

Models derived from physical processes are typically posed in continuous time.

Therefore, we need tools to integrate stochastic differential equations (SDEs) over each time step within a Gaussian filter.

For continuous-time linear systems with additive white noise, we have tools to discretise these processes exactly.

For nonlinear systems, we can integrate the SDEs numerically and propagate the mean and covariance through the integration step using an affine or unscented transform. To do this, we will use a fixed-step 4^{th} order Runge Kutta integration method.

RK4 SDE integration

Given the following SDE:

(standard form) $$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) + \dot{\mathbf{w}}(t), \qquad \dot{\mathbf{w}}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}\,\delta(t - t'))$$

(infinitesimal form) $$\mathrm{d}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t))\,\mathrm{d}t + \mathrm{d}\mathbf{w}(t), \quad \mathrm{d}\mathbf{w}(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}\,\mathrm{d}t)$$

The RK4 step is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{1}{6}(\mathbf{f}_1 + 2\mathbf{f}_2 + 2\mathbf{f}_3 + \mathbf{f}_4)\Delta t + \Delta\mathbf{w}_k, \qquad \Delta\mathbf{w}_k \sim (\mathbf{0}, \mathbf{Q}\Delta t)$$

where

$$\begin{aligned}
\mathbf{f}_1 &= \mathbf{f}(\mathbf{x}_k) \\
\mathbf{f}_2 &= \mathbf{f}(\mathbf{x}_k + \frac{1}{2}\mathbf{f}_1\Delta t + \frac{1}{2}\Delta\mathbf{w}_k) \\
\mathbf{f}_3 &= \mathbf{f}(\mathbf{x}_k + \frac{1}{2}\mathbf{f}_2\Delta t + \frac{1}{2}\Delta\mathbf{w}_k) \\
\mathbf{f}_4 &= \mathbf{f}(\mathbf{x}_k + \mathbf{f}_3\Delta t + \Delta\mathbf{w}_k) \\
\Delta t &= t_{k+1} - t_k
\end{aligned}$$

RK4 SDE integration

Since noise is injected into the input of the function $\mathbf{f}(\cdot)$ during the RK4 step, the resulting update is not affine in $\Delta\mathbf{w}_k$.

To deal with this in the time update of a Gaussian filter, we can employ **noise augmentation** and apply a transform to the map

$$\boldsymbol{\Phi}: \begin{bmatrix} \mathbf{x}_k \\ \Delta\mathbf{w}_k \end{bmatrix} \mapsto \mathbf{x}_{k+1}$$

to map the mean and covariance

$$\mathcal{T}\{\boldsymbol{\Phi}\}: \left(\begin{bmatrix} \boldsymbol{\mu}_k \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}\Delta t \end{bmatrix} \right) \mapsto (\boldsymbol{\mu}_{k+1}, \mathbf{P}_{k+1})$$

If we use an affine transform, we need the gradients of $\boldsymbol{\Phi}$ w.r.t. $\mathbf{x}_k$ and $\Delta\mathbf{w}_k$.

We could compute the analytical gradients of the RK4 solution, or do something more elegant...

RK4 SDE integration

The gradients of $\mathbf{\Phi}$ w.r.t. $\mathbf{x}_k$ and $\mathbf{\Delta w}_k$ can be computed by the SDE integration itself.

To do this, augment the state and noise increment with the gradients to define the following matrices:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} & \frac{\partial \mathbf{x}}{\partial \mathbf{x}_k} & \frac{\partial \mathbf{x}}{\partial \Delta \mathbf{w}_k} \end{bmatrix}$$

$$\Delta \mathbf{W}_k = \begin{bmatrix} \Delta \mathbf{w}_k & \frac{\partial \Delta \mathbf{w}_k}{\partial \mathbf{x}_k} & \frac{\partial \Delta \mathbf{w}_k}{\partial \Delta \mathbf{w}_k} \end{bmatrix} = \begin{bmatrix} \Delta \mathbf{w}_k & \mathbf{0} & \mathbf{I} \end{bmatrix}$$

Taking the time derivative of $\mathbf{X}$ and applying the chain rule yields the following matrix differential equation

$$\underbrace{\begin{bmatrix} \dot{\mathbf{x}} & \frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{x}_k} & \frac{\partial \dot{\mathbf{x}}}{\partial \Delta \mathbf{w}_k} \end{bmatrix}}_{\dot{\mathbf{X}}} = \underbrace{\begin{bmatrix} \mathbf{f}(\mathbf{x}) & \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{x}_k} & \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \Delta \mathbf{w}_k} \end{bmatrix}}_{\mathbf{F}(\mathbf{X})}$$

RK4 SDE integration

Then evaluate

$$\mathbf{X}_{k+1} = \mathbf{X}_k + \frac{1}{6}(\mathbf{F}_1 + 2\mathbf{F}_2 + 2\mathbf{F}_3 + \mathbf{F}_4)\Delta t + \Delta \mathbf{W}_k,$$

where

$$\mathbf{X}_k = \begin{bmatrix} \mathbf{x}_k & \frac{\partial \mathbf{x}_k}{\partial \mathbf{x}_k} & \frac{\partial \mathbf{x}_k}{\partial \Delta \mathbf{w}_k} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_k & \mathbf{I} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{F}_1 = \mathbf{F}(\mathbf{X}_k)$$

$$\mathbf{F}_2 = \mathbf{F}(\mathbf{X}_k + \frac{1}{2}\mathbf{F}_1 \Delta t + \frac{1}{2}\Delta \mathbf{W}_k)$$

$$\mathbf{F}_3 = \mathbf{F}(\mathbf{X}_k + \frac{1}{2}\mathbf{F}_2 \Delta t + \frac{1}{2}\Delta \mathbf{W}_k)$$

$$\mathbf{F}_4 = \mathbf{F}(\mathbf{X}_k + \mathbf{F}_3 \Delta t + \Delta \mathbf{W}_k)$$

and read off the SDE solution and its gradients from

$$\mathbf{X}_{k+1} = \begin{bmatrix} \mathbf{x}_{k+1} & \frac{\partial \mathbf{x}_{k+1}}{\partial \mathbf{x}_k} & \frac{\partial \mathbf{x}_{k+1}}{\partial \Delta \mathbf{w}_k} \end{bmatrix}$$

Gaussian filter

A problem

If we assume that the initial state and the process and the measurement models are Gaussian

$$p(\mathbf{x}_1) = \mathcal{N}(\mathbf{x}_1; \boldsymbol{\mu}_{1|0}, \mathbf{P}_{1|0})$$
$$p(\mathbf{x}_{k+1}|\mathbf{x}_k) = \mathcal{N}(\mathbf{x}_k; \mathbf{f}_k(\mathbf{x}), \mathbf{Q}_k)$$
$$p(\mathbf{y}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{y}_k; \mathbf{h}_k(\mathbf{x}), \mathbf{R}_k)$$

then the Bayes filter

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

$$p(\mathbf{x}_{k+1}|\mathbf{y}_{1:k}) = \int_{\mathcal{X}} p(\mathbf{x}_{k+1}|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{y}_{1:k})\, \mathrm{d}\mathbf{x}_k,$$

can result in non-Gaussian state distributions due to the nonlinearity in the process and measurement mean functions, $\mathbf{f}_k(\mathbf{x})$ and $\mathbf{h}_k(\mathbf{x})$.

Gaussian filter

In a Gaussian filter, we *assert* that the filtered and predicted state distributions are Gaussian at each time step, i.e.,

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \approx \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k}, \mathbf{P}_{k|k})$$
$$p(\mathbf{x}_{k+1}|\mathbf{y}_{1:k}) \approx \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k+1|k}, \mathbf{P}_{k+1|k})$$

If we accept these assertions,

- How can we obtain the "best" Gaussian approximation to the filtered state distribution?
- Can we extend the filter to handle a non-Gaussian measurement likelihood function?

We will address both questions by finding the "best" Gaussian approximation to the filtered state distribution:

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \propto p(\mathbf{y}_k|\mathbf{x}_k)\, p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) = p(\mathbf{y}_k|\mathbf{x}_k)\, \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1})$$

Gaussian filter

For the process, we consider the following continuous-time SDE:

$$\mathrm{d}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t))\,\mathrm{d}t + \mathrm{d}\mathbf{w}(t), \quad \mathrm{d}\mathbf{w}(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}\,\mathrm{d}t)$$

which may be integrated over each time step to obtain a discrete-time process model

$$\mathbf{x}_{k+1} = \mathbf{f}_k(\mathbf{x}_k, \Delta\mathbf{w}_k), \quad \Delta\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}\,\Delta t_k), \quad \Delta t_k = t_{k+1} - t_k$$

Modelling the process in continuous time and discretising it online:

- avoids non-physical noise structures that can arise by directly posing a discrete-time process model
- enables predictions to be made at any time (e.g., control steps not necessarily at sensor measurement events, multi-rate sensor data fusion)

Nonlinear Kalman filter

If the measurement likelihood is Gaussian, i.e.,

$$p(\mathbf{y}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{y}_k; \mathbf{h}_k(\mathbf{x}_k), \mathbf{R}_k)$$

then we can also write down an explicit measurement model

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k, \quad \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$$

The Gaussian filter with a Gaussian measurement model is a nonlinear Kalman filter (e.g., extended Kalman filter or unscented Kalman filter).

If the measurement likelihood is non-Gaussian, we may not be able to write down an explicit measurement model and so need to work with the measurement likelihood function directly to obtain a Gaussian posterior density, e.g., via the Laplace approximation.

Nonlinear Kalman filter

1. Form the prediction density $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$ by propagating $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$ through the transformation

$$\boldsymbol{\Phi}: \begin{bmatrix} \mathbf{x}_{k-1} \\ \Delta\mathbf{w}_{k-1} \end{bmatrix} \mapsto \mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \Delta\mathbf{w}_{k-1})$$

$$\mathcal{T}\{\boldsymbol{\Phi}\}: \left(\begin{bmatrix} \boldsymbol{\mu}_{k-1|k-1} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{k-1|k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}\,\Delta t_{k-1} \end{bmatrix} \right) \mapsto (\boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1})$$

2. Form the joint pdf $p(\mathbf{x}_k, \mathbf{y}_k|\mathbf{y}_{1:k-1})$ by propagating $p(\mathbf{x}_k, \mathbf{v}_k|\mathbf{y}_{1:k-1})$ through the transformation

$$\boldsymbol{\Psi}: \begin{bmatrix} \mathbf{x}_k \\ \mathbf{v}_k \end{bmatrix} \mapsto \begin{bmatrix} \mathbf{x}_k \\ \mathbf{y}_k \end{bmatrix} = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k \end{bmatrix}$$

$$\mathcal{T}\{\boldsymbol{\Psi}\}: \left(\begin{bmatrix} \boldsymbol{\mu}_{k|k-1} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{k|k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \right) \mapsto \left(\begin{bmatrix} \boldsymbol{\mu}_{k|k-1} \\ \boldsymbol{\mu}_{\mathbf{y}} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{k|k-1} & \mathbf{P}_{\mathbf{xy}} \\ \mathbf{P}_{\mathbf{yx}} & \mathbf{P}_{\mathbf{yy}} \end{bmatrix} \right)$$

3. Compute the conditional pdf $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ by conditioning the joint pdf on $\mathbf{y}_k$

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

$$\boldsymbol{\mu}_{k|k} = \boldsymbol{\mu}_{k|k-1} + \mathbf{P}_{\mathbf{xy}}\mathbf{P}_{\mathbf{yy}}^{-1}(\mathbf{y}_k - \boldsymbol{\mu}_{\mathbf{y}})$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{P}_{\mathbf{xy}}\mathbf{P}_{\mathbf{yy}}^{-1}\mathbf{P}_{\mathbf{yx}}$$

Nonlinear Kalman filter

We have a choice in how to perform the transformations

- Linearise the models about a nominal operating state and then use affine Gaussian transformations.
  This leads to the **Linearised Kalman Filter** (LKF).
- Linearise the process model about the previous filtered mean and linearise the measurement model about the predicted mean and then use affine Gaussian transformations.
  This leads to the **Extended Kalman Filter** (EKF).
- Use the unscented transform (UT) to evaluate the mean and covariance of the prediction density $p(\mathbf{x}_k | \mathbf{y}_{1:k-1})$ and the joint density $p(\mathbf{x}_k, \mathbf{y}_k | \mathbf{y}_{1:k-1})$.
  This leads to the **Unscented Kalman Filter** (UKF).

Laplace approximation

To handle a more general measurement likelihood, let us apply the Laplace approximation to obtain a Gaussian approximation to the filtered density directly

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \propto p(\mathbf{y}_k|\mathbf{x}_k)\,\mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1})$$

which yields

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \approx \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k}, \mathbf{P}_{k|k})$$

where

$$\boldsymbol{\mu}_{k|k} = \arg\min_{\mathbf{x}_k} \mathcal{V}(\mathbf{x}_k), \qquad \mathbf{P}_{k|k} = \left(\frac{\partial^2 \mathcal{V}}{\partial \mathbf{x}_k^2}\bigg|_{\mathbf{x}_k = \boldsymbol{\mu}_{k|k}} \right)^{-1}$$

and

$$\begin{aligned}\mathcal{V}(\mathbf{x}_k) &= -\log p(\mathbf{y}_k|\mathbf{x}_k) - \log \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1}) \\ &= \frac{1}{2}(\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1})^{\mathsf{T}} \mathbf{P}_{k|k-1}^{-1} (\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1}) + \frac{1}{2}\log\det 2\pi \mathbf{P}_{k|k-1} - \log p(\mathbf{y}_k|\mathbf{x}_k)\end{aligned}$$

Laplace filter

1. Form the prediction density $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$ by propagating $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$ through the transformation

$$\boldsymbol{\Phi}: \begin{bmatrix} \mathbf{x}_{k-1} \\ \Delta\mathbf{w}_{k-1} \end{bmatrix} \mapsto \mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \Delta\mathbf{w}_{k-1})$$

$$\mathcal{T}\{\boldsymbol{\Phi}\}: \left(\begin{bmatrix} \boldsymbol{\mu}_{k-1|k-1} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{k-1|k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}\,\Delta t_{k-1} \end{bmatrix} \right) \mapsto (\boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1})$$

2. Compute the filtered pdf $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ using the Laplace approximation

$$\mathcal{V}(\mathbf{x}_k) = -\log \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \mathbf{P}_{k|k-1}) - \log p(\mathbf{y}_k|\mathbf{x}_k)$$

$$\boldsymbol{\mu}_{k|k} = \arg\min_{\mathbf{x}_k} \mathcal{V}(\mathbf{x}_k)$$

$$\mathbf{P}_{k|k} = \left(\frac{\partial^2 \mathcal{V}}{\partial \mathbf{x}_k^2}\bigg|_{\mathbf{x}_k = \boldsymbol{\mu}_{k|k}} \right)^{-1}$$

Square-root implementation

Matrix square roots

Generalise concept of square root to positive definite and positive semidefinite matrices.

Common matrix square roots

- Spectral square root (eigendecomposition)
- Upper and lower Cholesky factorisations

Working with Cholesky factored covariance

- Guarantees positive (semi-)definiteness of **P**
- Guarantees symmetry of **P**
- Improves numerical precision
- Reduces storage requirements

Let **S** denote a square root covariance matrix (**S** for square root, **S** for standard deviation)

We can build Kalman filters to work directly with square root covariance.

Is the square-root implementation used?

Square-root implementations have been essential since the early days of state estimation.

- Potter and Stern (1963) developed a square-root formulation of the Kalman filter to implement on the Apollo onboard computer.
- A square-root information filter was developed in 1969 for JPL's Mariner 10 mission to Venus (Bierman, 1977).

"*A number of practitioners have argued, with considerable logic, that square root filters should always be adopted in preference to the standard Kalman filter recursion.*"
— Maybeck (1982)

Spectral square root of a symmetric positive semi-definite matrix

Consider the eigendecomposition of $\mathbf{M}$

$$\mathbf{M}\mathbf{v}_i = \lambda_i \mathbf{v}_i \implies \mathbf{M}\mathbf{V} = \mathbf{V}\boldsymbol{\Lambda}$$

$$\mathbf{V} = \begin{bmatrix}\mathbf{v}_1, \mathbf{v}_2, \dots\end{bmatrix}, \quad \boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \lambda_2, \dots)$$

$$\mathbf{M} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{-1} = \mathbf{V}\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{\Lambda}^{\frac{1}{2}}\mathbf{V}^{-1}$$

$$= \underbrace{\mathbf{V}\boldsymbol{\Lambda}^{\frac{1}{2}}\mathbf{V}^{-1}}_{\mathbf{M}^{\frac{1}{2}}}\underbrace{\mathbf{V}\boldsymbol{\Lambda}^{\frac{1}{2}}\mathbf{V}^{-1}}_{\mathbf{M}^{\frac{1}{2}}}$$

This is the spectral square root for any square $\mathbf{M}$.

If $\mathbf{M} = \mathbf{M}^\mathsf{T} \succcurlyeq 0$, then the eigenvalues are positive and the matrix of eigenvectors is orthogonal,

$$\mathbf{M}^{\frac{1}{2}} = \mathbf{V}\boldsymbol{\Lambda}^{\frac{1}{2}}\mathbf{V}^\mathsf{T}, \qquad (\boldsymbol{\Lambda}^{\frac{1}{2}})_{ii} = \sqrt{\lambda_i}$$

Cholesky decomposition of a symmetric positive definite matrix

The Cholesky factorisation is a triangular matrix whose product with its transpose equals a given matrix.

Lower Cholesky factorisation

$$\mathbf{M} = \mathbf{L}\mathbf{L}^{\mathsf{T}}, \quad L_{ij} = 0, \forall i > j$$

MATLAB: `L = chol(M, 'lower');`

$$\mathbf{L} = \begin{bmatrix} L_{11} & 0 & \cdots & 0 \\ L_{21} & L_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ L_{n1} & \cdots & L_{n,n-1} & L_{nn} \end{bmatrix}$$

Upper Cholesky factorisation

$$\mathbf{M} = \mathbf{R}^{\mathsf{T}}\mathbf{R}, \quad R_{ij} = 0, \forall i < j$$

MATLAB: `R = chol(M);`

$$\mathbf{R} = \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1n} \\ 0 & R_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & R_{n-1,n} \\ 0 & \cdots & 0 & R_{nn} \end{bmatrix}$$

QR decomposition

Orthogonal-triangular decomposition **A** = **QR** such that

- **Q** is orthogonal
- **R** is upper triangular

$$\mathbf{A} = \mathbf{QR} = \begin{bmatrix} \mathbf{Y} & \mathbf{Z} \end{bmatrix} \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{bmatrix}$$

Relation between upper Cholesky and QR

$$\mathbf{A}^\mathsf{T}\mathbf{A} = \mathbf{R}^\mathsf{T}\mathbf{Q}^\mathsf{T}\mathbf{Q}\mathbf{R} = \mathbf{R}^\mathsf{T}\mathbf{R}$$

$$\mathbf{R} = \mathrm{chol}(\mathbf{A}^\mathsf{T}\mathbf{A}) = \mathrm{qr}(\mathbf{A})$$

MATLAB (R2022a or later):

```
% Assume A is tall (m >= n) and full rank
[m, n]  = size(A);
[Q, R]  = qr(A);          % Full QR (Q is m-by-m, R is m-by-n)
[Y, R1] = qr(A, "econ");  % Thin QR (Y is m-by-n, R1 is n-by-n)
R       = qr(A);          % Q-less full QR
R1      = qr(A, "econ");  % Q-less thin QR
```

Pythagorean QR

Square root of sum of squares

$$\begin{aligned}
\mathbf{R}^\mathsf{T}\mathbf{R} &= \mathbf{B}^\mathsf{T}\mathbf{B} + \mathbf{C}^\mathsf{T}\mathbf{C} \\
&= \begin{bmatrix}\mathbf{B}^\mathsf{T} & \mathbf{C}^\mathsf{T}\end{bmatrix}\begin{bmatrix}\mathbf{B} \\ \mathbf{C}\end{bmatrix} \\
\mathrm{chol}(\mathbf{R}^\mathsf{T}\mathbf{R}) &= \mathrm{chol}\left(\begin{bmatrix}\mathbf{B}^\mathsf{T} & \mathbf{C}^\mathsf{T}\end{bmatrix}\begin{bmatrix}\mathbf{B} \\ \mathbf{C}\end{bmatrix}\right) \\
\mathbf{R} &= \mathrm{qr}\left(\begin{bmatrix}\mathbf{B} \\ \mathbf{C}\end{bmatrix}\right)
\end{aligned}$$

An implementation can be made more efficient by reordering the rows of $\mathbf{B}$ *and* $\mathbf{C}$ *to minimise the number of operations required to upper-triangularise* $\mathbf{R}$.

Square-root moment form

So far, we have been working with Gaussian distributions in their *moment form*, parameterised by the mean $\boldsymbol{\mu}$ and the covariance $\mathbf{P}$,

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{P}) = \frac{1}{\sqrt{\det 2\pi\mathbf{P}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\mathbf{P}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

Let us introduce an upper-triangular matrix $\mathbf{S}$ such that $\mathbf{S}^\mathsf{T}\mathbf{S} = \mathbf{P}$.

Then we can express the Gaussian distribution in *square-root moment form*,

$$\begin{aligned}\mathcal{N}^{\frac{1}{2}}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{S}) &= \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{S}^\mathsf{T}\mathbf{S}) \\ &= \frac{1}{\sqrt{\det 2\pi\mathbf{S}^\mathsf{T}\mathbf{S}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}(\mathbf{S}^\mathsf{T}\mathbf{S})^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)\end{aligned}$$

$$\mathcal{N}^{\frac{1}{2}}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{S}) = \frac{1}{\left|\det \sqrt{2\pi}\mathbf{S}\right|} \exp\left(-\frac{1}{2}\left\|\mathbf{S}^{-\mathsf{T}}(\mathbf{x}-\boldsymbol{\mu})\right\|^2\right)$$

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Log determinants

In MATLAB,

```
>> log(det(1e-4*eye(100)))
ans =
  -Inf

>> log(det(1e4*eye(100)))
ans =
   Inf
```

Do you believe these values?

Example (log determinant): For each of the following cases for the square matrix $\mathbf{M}$, compute $\log\det\mathbf{M}$:

1. $\mathbf{M} = 10^{-4}\mathbf{I}_{100\times 100}$.
2. $\mathbf{M} = 10^{4}\mathbf{I}_{100\times 100}$.

To compute the log of a matrix determinant, we exploit the fact that the determinant of a matrix is equal to the product of the eigenvalues and that the log of a product is equal to the sum of the logs,

$$\log\det\mathbf{M} = \log\prod_{i=1}^{n}\lambda_i = \sum_{i=1}^{n}\log\lambda_i,$$

where $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of $\mathbf{M}$.

Since $\mathbf{M}$ is a diagonal matrix with equal entries along the diagonal in each of cases (1) and (2), the eigenvalues of $\mathbf{M}$ are equal to 10^{-4} and 10^{4}, respectively. Evaluating the RHS yields $\log\det\mathbf{M} = 100\log 10^{-4} \approx -921.03$ for case (1) and $\log\det\mathbf{M} = 100\log 10^{4} \approx 921.03$ for case (2).

Note that naïvely computing the determinant and then taking the logarithm can yield incorrect results in IEEE754 double precision floating point arithmetic. In case (1), computing $\det\mathbf{M} = 10^{-400}$ underflows to zero—the log of which yields the erroneous result $\log\det\mathbf{M} = -\infty$. In case (2), computing $\det\mathbf{M} = 10^{400}$ overflows to positive infinity—the log of which yields the erroneous result $\log\det\mathbf{M} = \infty$.

Log of a Gaussian

Given an upper triangular matrix $\mathbf{S}$ such that $\mathbf{S}^\mathsf{T}\mathbf{S} = \mathbf{P}$

$$
\begin{aligned}
\log \mathcal{N}^{\frac{1}{2}}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{S}) &= \log\left((\det 2\pi\mathbf{S}^\mathsf{T}\mathbf{S})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}(\mathbf{S}^\mathsf{T}\mathbf{S})^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)\right) \\
&= -\frac{1}{2}\log(\det 2\pi\mathbf{S}^\mathsf{T}\mathbf{S}) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\mathbf{S}^{-1}\mathbf{S}^{-\mathsf{T}}(\mathbf{x}-\boldsymbol{\mu}) \\
&= -\frac{1}{2}\log\left((2\pi)^n(\det \mathbf{S})^2\right) - \frac{1}{2}\left(\mathbf{S}^{-\mathsf{T}}(\mathbf{x}-\boldsymbol{\mu})\right)^\mathsf{T}\mathbf{S}^{-\mathsf{T}}(\mathbf{x}-\boldsymbol{\mu}) \\
&= -\frac{n}{2}\log 2\pi - \log|\det \mathbf{S}| - \frac{1}{2}\left\|\mathbf{S}^{-\mathsf{T}}(\mathbf{x}-\boldsymbol{\mu})\right\|^2 \\
&= -\frac{n}{2}\log 2\pi - \log\left|\prod_{i=1}^{n}\mathbf{S}_{ii}\right| - \frac{1}{2}\|\mathbf{w}\|^2 \\
&= -\frac{n}{2}\log 2\pi - \sum_{i=1}^{n}\log|\mathbf{S}|_{ii} - \frac{1}{2}\mathbf{w}^\mathsf{T}\mathbf{w}
\end{aligned}
$$

where $|\cdot|_{ii}$ denotes the absolute value of the i^{th} diagonal element and $\mathbf{w}$ is the solution to the triangular system of equations $\mathbf{S}^\mathsf{T}\mathbf{w} = \mathbf{x} - \boldsymbol{\mu}$

Square-root affine transformation

Consider the following

$$\boldsymbol{\Psi}\colon \mathbf{x} \mapsto \mathbf{y} = \mathbf{h}(\mathbf{x}), \qquad \mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}}, \mathbf{P}_{\mathbf{xx}})$$

$$\mathcal{T}\{\boldsymbol{\Psi}\}\colon (\boldsymbol{\mu}_{\mathbf{x}}, \mathbf{P}_{\mathbf{xx}}) \mapsto (\boldsymbol{\mu}_{\mathbf{y}}, \mathbf{P}_{\mathbf{yy}}) = (\mathbf{h}(\boldsymbol{\mu}_{\mathbf{x}}),\ \mathbf{C}\mathbf{P}_{\mathbf{xx}}\mathbf{C}^{\mathsf{T}}), \quad \mathbf{C} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\boldsymbol{\mu}_{\mathbf{x}}}$$

Let $\mathbf{S}_{\mathbf{xx}}$, and $\mathbf{S}_{\mathbf{yy}}$ be the upper Cholesky factors of $\mathbf{P}_{\mathbf{xx}}$ and $\mathbf{P}_{\mathbf{yy}}$, respectively. Then,

$$\begin{aligned}
\mathbf{P}_{\mathbf{yy}} &= \mathbf{C}\mathbf{P}_{\mathbf{xx}}\mathbf{C}^{\mathsf{T}} \\
\mathbf{S}_{\mathbf{yy}}^{\mathsf{T}}\mathbf{S}_{\mathbf{yy}} &= \mathbf{C}\mathbf{S}_{\mathbf{xx}}^{\mathsf{T}}\mathbf{S}_{\mathbf{xx}}\mathbf{C}^{\mathsf{T}} = (\mathbf{S}_{\mathbf{xx}}\mathbf{C}^{\mathsf{T}})^{\mathsf{T}}\mathbf{S}_{\mathbf{xx}}\mathbf{C}^{\mathsf{T}} \\
\mathbf{S}_{\mathbf{yy}} &= \mathrm{qr}\left(\mathbf{S}_{\mathbf{xx}}\mathbf{C}^{\mathsf{T}}\right)
\end{aligned}$$

The square-root affine transform of $\boldsymbol{\Psi}$: $\mathbf{x} \to \mathbf{y}$ is

$$\mathcal{T}^{\frac{1}{2}}\{\boldsymbol{\Psi}\}\colon (\boldsymbol{\mu}_{\mathbf{x}}, \mathbf{S}_{\mathbf{xx}}) \mapsto (\boldsymbol{\mu}_{\mathbf{y}}, \mathbf{S}_{\mathbf{yy}})$$

Square-root affine transformation

Inside `@Gaussian/affineTransform.m`:

```
function out = affineTransform(obj, h)

[muy, C] = h(obj.mu);   % Evaluate function at mean
Syy = qr(obj.S*C.', "econ");
out = Gaussian(muy, Syy);
```

Square-root time update (continuous)

To perform the RK4 step for a continuous-time process model, we need to perform a transformation through the map

$$\boldsymbol{\Phi}: \begin{bmatrix} \mathbf{x}_{k-1} \\ \Delta\mathbf{w}_{k-1} \end{bmatrix} \mapsto \mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \Delta\mathbf{w}_{k-1})$$

to map the mean and sqrt covariance of the joint previous state and noise increment into their predicted values for the current time step,

$$\mathcal{T}^{\frac{1}{2}}\{\boldsymbol{\Phi}\}: \left(\begin{bmatrix} \boldsymbol{\mu}_{k-1|k-1} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{S}_{k-1|k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{\mathbf{Q}}\sqrt{\Delta t_{k-1}} \end{bmatrix} \right) \mapsto (\boldsymbol{\mu}_{k|k-1}, \mathbf{S}_{k|k-1})$$

where $\mathbf{S}_{\mathbf{Q}}$ is an upper triangular matrix such that $\mathbf{S}_{\mathbf{Q}}^{\mathsf{T}}\,\mathbf{S}_{\mathbf{Q}} = \mathbf{Q}$ is the power spectral density of the continuous-time process noise.

Square-root time update (continuous)

Inside `@SystemEstimator/predict.m`:

```
function systemNext = predict(obj, timeNext)
dt = timeNext - obj.time;
systemNext = obj;
systemNext.time = timeNext;

% Augment state density with independent noise increment dw ~ N(0, Q*dt)
% [ x] ~ N([mu] , [P,    0])
% [dw]    ([ 0]   [0, Q*dt])
[pdw, idxQ] = obj.processNoise(dt);
pxdw = obj.density.join(pdw); % p(x, dw) = p(x)*p(dw)

% Propagate p(x[k], dw[k]) through RK4 to obtain p(x[k+1])
t = obj.time;
u = obj.input(t);
systemNext.density = pxdw.affineTransform( ...
    @(xdw) obj.RK4SDEHelper(t, xdw, u, dt, idxQ));
```

Square-root measurement correction

Recall $$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{y})} = \frac{p(\mathbf{y},\mathbf{x})}{p(\mathbf{y})} = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}}, \mathbf{P}_{\mathbf{x}|\mathbf{y}})$$

$$= \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{\mathbf{x}} + \mathbf{P}_{\mathbf{xy}}\mathbf{P}_{\mathbf{yy}}^{-1}(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}}), \mathbf{P}_{\mathbf{xx}} - \mathbf{P}_{\mathbf{xy}}\mathbf{P}_{\mathbf{yy}}^{-1}\mathbf{P}_{\mathbf{yx}})$$

$$p(\mathbf{y},\mathbf{x}) = p\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{x}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{x}\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}_{\mathbf{y}}\\ \boldsymbol{\mu}_{\mathbf{x}}\end{bmatrix}, \begin{bmatrix}\mathbf{P}_{\mathbf{yy}} & \mathbf{P}_{\mathbf{yx}}\\ \mathbf{P}_{\mathbf{xy}} & \mathbf{P}_{\mathbf{xx}}\end{bmatrix}\right)$$

Upper Cholesky factorisation of the joint covariance ($\mathbf{S}_1$ and $\mathbf{S}_3$ are upper triangular)

$$\begin{bmatrix}\mathbf{S}_1^\mathsf{T} & \mathbf{0}\\ \mathbf{S}_2^\mathsf{T} & \mathbf{S}_3^\mathsf{T}\end{bmatrix}\begin{bmatrix}\mathbf{S}_1 & \mathbf{S}_2\\ \mathbf{0} & \mathbf{S}_3\end{bmatrix} = \begin{bmatrix}\mathbf{P}_{\mathbf{yy}} & \mathbf{P}_{\mathbf{yx}}\\ \mathbf{P}_{\mathbf{xy}} & \mathbf{P}_{\mathbf{xx}}\end{bmatrix}$$

$$\mathbf{S}_1^\mathsf{T}\mathbf{S}_1 = \mathbf{P}_{\mathbf{yy}}$$
$$\mathbf{S}_1^\mathsf{T}\mathbf{S}_2 = \mathbf{P}_{\mathbf{yx}}$$
$$\mathbf{S}_2^\mathsf{T}\mathbf{S}_1 = \mathbf{P}_{\mathbf{xy}}$$
$$\mathbf{S}_2^\mathsf{T}\mathbf{S}_2 + \mathbf{S}_3^\mathsf{T}\mathbf{S}_3 = \mathbf{P}_{\mathbf{xx}}$$

$$\mathbf{P}_{\mathbf{xy}}\mathbf{P}_{\mathbf{yy}}^{-1} = \mathbf{S}_2^\mathsf{T}\mathbf{S}_1(\mathbf{S}_1^\mathsf{T}\mathbf{S}_1)^{-1} = \mathbf{S}_2^\mathsf{T}\mathbf{S}_1\mathbf{S}_1^{-1}\mathbf{S}_1^{-\mathsf{T}} = \mathbf{S}_2^\mathsf{T}\mathbf{S}_1^{-\mathsf{T}}$$

$$\mathbf{P}_{\mathbf{x}|\mathbf{y}} = \mathbf{P}_{\mathbf{xx}} - \mathbf{P}_{\mathbf{xy}}\mathbf{P}_{\mathbf{yy}}^{-1}\mathbf{P}_{\mathbf{yx}} = \mathbf{P}_{\mathbf{xx}} - \mathbf{P}_{\mathbf{xy}}(\mathbf{S}_1^\mathsf{T}\mathbf{S}_1)^{-1}\mathbf{P}_{\mathbf{yx}}$$

$$= \mathbf{P}_{\mathbf{xx}} - \mathbf{P}_{\mathbf{xy}}\mathbf{S}_1^{-1}\mathbf{S}_1^{-\mathsf{T}}\mathbf{P}_{\mathbf{yx}} = \mathbf{P}_{\mathbf{xx}} - \mathbf{S}_2^\mathsf{T}\mathbf{S}_2 = \mathbf{S}_3^\mathsf{T}\mathbf{S}_3$$

$$\boldsymbol{\mu}_{\mathbf{x}|\mathbf{y}} = \boldsymbol{\mu}_{\mathbf{x}} + \mathbf{S}_2^\mathsf{T}\mathbf{S}_1^{-\mathsf{T}}(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}})$$
$$\mathbf{S}_{\mathbf{x}|\mathbf{y}} = \mathbf{S}_3$$

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Square-root measurement correction

Inside `@MeasurementGaussianLikelihood/update.m`:

```
function [obj, system] = update(obj, system)
nx = system.density.dim();

% Apply transform through the measurement model augmented with the
identity function
jointFunc = @(x) obj.augmentedPredictDensity(x, system);
pyx = system.density.affineTransform(jointFunc); % Or unscentedTransform

ny = length(obj.y);
idxX = ny+1:nx+ny;
idxY = 1:ny;
system.density = pyx.conditional(idxX, idxY, obj.y);
```

Square-root measurement correction

Inside `@Gaussian/conditional.m`:

```
% Given joint density p(x), return conditional density p(x(idxA) | x(idxB) = xB)
function out = conditional(obj, idxA, idxB, xB)

SS = qr([obj.S(:, idxB), obj.S(:, idxA)], "econ");

nB = length(idxB);
SBB = SS(1:nB, 1:nB);
SBA = SS(1:nB, nB+1:end);
SAA = SS(nB+1:end, nB+1:end);

muA = obj.mu(idxA);
muB = obj.mu(idxB);

muc = muA + SBA.'*(SBB.'\(xB - muB)); % triangular forward substitution
Sc = SAA;
out = Gaussian(muc, Sc);
```

Square-root nonlinear Kalman filter

1. Form the prediction density $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$ by propagating $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$ through the transformation

$$\boldsymbol{\Phi}: \begin{bmatrix} \mathbf{x}_{k-1} \\ \Delta\mathbf{w}_{k-1} \end{bmatrix} \mapsto \mathbf{x}_k$$

$$\mathcal{T}^{\frac{1}{2}}\{\boldsymbol{\Phi}\}: \left(\begin{bmatrix} \boldsymbol{\mu}_{k-1|k-1} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{S}_{k-1|k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_\mathbf{Q}\sqrt{\Delta t} \end{bmatrix} \right) \mapsto (\boldsymbol{\mu}_{k|k-1}, \mathbf{S}_{k|k-1})$$

2. Form the joint pdf $p(\mathbf{y}_k, \mathbf{x}_k|\mathbf{y}_{1:k-1})$ by propagating $p(\mathbf{x}_k, \mathbf{v}_k|\mathbf{y}_{1:k-1})$ through the transformation

$$\boldsymbol{\Psi}: \begin{bmatrix} \mathbf{x}_k \\ \mathbf{v}_k \end{bmatrix} \mapsto \begin{bmatrix} \mathbf{y}_k \\ \mathbf{x}_k \end{bmatrix} = \begin{bmatrix} \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k \\ \mathbf{x}_k \end{bmatrix}$$

$$\mathcal{T}^{\frac{1}{2}}\{\boldsymbol{\Psi}\}: \left(\begin{bmatrix} \boldsymbol{\mu}_{k|k-1} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{S}_{k|k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_\mathbf{R} \end{bmatrix} \right) \mapsto \left(\begin{bmatrix} \boldsymbol{\mu}_\mathbf{y} \\ \boldsymbol{\mu}_{k|k-1} \end{bmatrix}, \begin{bmatrix} \mathbf{S}_1 & \mathbf{S}_2 \\ \mathbf{0} & \mathbf{S}_3 \end{bmatrix} \right)$$

3. Compute the conditional pdf $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ by conditioning the joint pdf on $\mathbf{y}_k$

$$\boldsymbol{\mu}_{k|k} = \boldsymbol{\mu}_{k|k-1} + \mathbf{S}_2^\mathsf{T}\mathbf{S}_1^{-\mathsf{T}}(\mathbf{y}_k - \boldsymbol{\mu}_\mathbf{y})$$

$$\mathbf{S}_{k|k} = \mathbf{S}_3$$

Square-root Laplace filter

1. Form the prediction density $p(\mathbf{x}_k|\mathbf{y}_{1:k-1})$ by propagating $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$ through the transformation

$$\boldsymbol{\Phi}: \begin{bmatrix} \mathbf{x}_{k-1} \\ \Delta\mathbf{w}_{k-1} \end{bmatrix} \mapsto \mathbf{x}_k$$

$$\mathcal{T}^{\frac{1}{2}}\{\boldsymbol{\Phi}\}: \left(\begin{bmatrix} \boldsymbol{\mu}_{k-1|k-1} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{S}_{k-1|k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{\mathbf{Q}}\sqrt{\Delta t} \end{bmatrix} \right) \mapsto (\boldsymbol{\mu}_{k|k-1}, \mathbf{S}_{k|k-1})$$

2. Compute the filtered pdf $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ from the Laplace approximation

$$\mathcal{V}(\mathbf{x}_k) = -\log \mathcal{N}^{\frac{1}{2}}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \mathbf{S}_{k|k-1}) - \log p(\mathbf{y}_k|\mathbf{x}_k)$$

$$\boldsymbol{\mu}_{k|k} = \arg\min_{\mathbf{x}_k} \mathcal{V}(\mathbf{x}_k)$$

$$\mathbf{S}_{k|k}^{\mathsf{T}}\mathbf{S}_{k|k} = \left(\frac{\partial^2 \mathcal{V}}{\partial \mathbf{x}_k^2}\bigg|_{\mathbf{x}_k=\boldsymbol{\mu}_{k|k}} \right)^{-1}$$

References

- G.H. Golub and C.F. Van Loan, 2012. *Matrix Computations*, 4th Ed., JHU Press.
- Potter, J., Stern, R., 1963. *Statistical filtering of space navigation measurements*. In: Guidance and Control Conference. p. 333.
- Bierman, G. J., 1977. *Factorization Methods for Discrete Sequential Estimation*. Academic Press.
- Maybeck, P. S., 1982. *Stochastic models, estimation, and control*. Academic press.