Model-Based Estimation Notes

Lecturer: Mark Psiaki, Cornell MAE
Scribe: Kevin Kircher, Cornell MAE

These are class notes from MAE 6780, Model-Based Estimation, at Cornell University in the spring of 2014. We consider the problem of estimating the parameters and states of a stochastic dynamical system from noisy or incomplete measurements.

The general process for solving such a problem:

1. Use physics, geometry and kinematics to model the system
    - encode dynamics in a differential or difference equation
    - understand how model parameters influence measurements
    - characterize model uncertainty
2. Analyze observability: assuming our sensors are perfectly accurate, can we uniquely determine the states/parameters from our measurements?
3. If the system is observable, apply an estimation technique to approximate the states/parameters based on the data. Also determine the accuracy of the estimate
4. Use the estimates to answer questions about the physical system

These notes cover step 3 and, to a lesser extent, step 2. The primary source is Prof. Mark Psiaki's lecture notes. The secondary source is *Estimation with Applications to Tracking and Navigation* by Yaakov Bar-Shalom, X. Rong Li and Thiagalingam Kirubarajan.

Contents

1 Parameter estimation basics **4**
- 1.1 Fisher estimation . . . 4
- 1.2 Bayesian estimation . . . 5
- 1.3 Estimator precision . . . 6

2 Linear least squares parameter estimation **8**
- 2.1 Single observation, Gaussian prior . . . 8
- 2.2 Multiple observations, no prior . . . 9

3 Nonlinear least squares parameter estimation **13**
- 3.1 Numerical solution . . . 13
- 3.2 Solution uniqueness and observability . . . 16
- 3.3 Covariance matrix . . . 17

4 Stochastic linear dynamical systems **18**
- 4.1 Continuous-time . . . 18
- 4.2 Discrete-time . . . 19
- 4.3 Pre-whitening colored noise . . . 22

5 State estimation in linear Gaussian dynamical systems **23**
- 5.1 The LG system . . . 23
- 5.2 Definitions and notation . . . 24
- 5.3 The KF algorithm . . . 25
- 5.4 Some KF properties . . . 26
- 5.5 KF derivation 1: MMSE . . . 28
- 5.6 KF derivation 2: MAP . . . 28
- 5.7 Tests of KF correctness . . . 29
- 5.8 KF initialization . . . 35
- 5.9 Model mismatch . . . 38

6 The square-root information filter **40**
- 6.1 The IF algorithm . . . 40
- 6.2 SRIF background . . . 42
- 6.3 The SRIF algorithm . . . 46

7 Smoothing **49**
- 7.1 Notation . . . 49
- 7.2 Covariance-based fixed interval smoothing . . . 50
- 7.3 Information-based fixed interval smoothing . . . 53

8 State estimation in stochastic nonlinear dynamical systems **56**
8.1 Discretization of nonlinear differential equations 56
8.2 The extended Kalman filter 61
8.3 The iterated EKF 71
8.4 The unscented Kalman filter 76
8.5 The particle filter 80

1 Parameter estimation basics

Imagine you have a big vector of observations

$$\mathbf{z}^k = \begin{bmatrix} \mathbf{z}(1) \\ \vdots \\ \mathbf{z}(k) \end{bmatrix} \in \mathbf{R}^{kn_z}$$

where each $\mathbf{z}(i) \in \mathbf{R}^{n_z}$ is a realization of a random vector whose distribution has an unknown parameter $\mathbf{x} \in \mathbf{R}^{n_x}$. How would you use $\mathbf{z}^k$ to estimate $\mathbf{x}$? This section presents a few techniques:

- maximum likelihood
- least squares
- maximum *a posteriori*
- minimum mean squared error

It also covers some metrics for quantifying "how good" an estimator is:

- error
- bias
- covariance
- mean squared error

1.1 Fisher estimation

- in **Fisher** estimation (named after the British statistician Ronald Fisher, 1890-1962), we treat $\mathbf{x}$ as an unknown, deterministic constant
- the primary tool of Fisher estimation is the **likelihood function** $p(\mathbf{z}^k|\mathbf{x})$, which measures how likely the observations are for a given parameter value
- the **maximum likelihood estimator** (MLE) of $\mathbf{x}$ is

$$\hat{\mathbf{x}}_{\mathrm{ML}}(\mathbf{z}^k) = \arg\max_{\mathbf{x}} p(\mathbf{z}^k|\mathbf{x})$$

- if the likelihood function can be separated according to

$$p(\mathbf{z}^k|\mathbf{x}) = f_1(g(\mathbf{z}^k), \mathbf{x}) f_2(\mathbf{z}^k)$$

then

$$\hat{\mathbf{x}}_{\mathrm{ML}}(\mathbf{z}^k) = \arg\max_{\mathbf{x}} p(\mathbf{z}^k|\mathbf{z}) = \arg\max_{\mathbf{x}} f_1(g(\mathbf{z}^k), \mathbf{x})$$

So $f_2(\mathbf{z}^k)$ does not affect the minimization of $p(\mathbf{z}^k|\mathbf{x})$. In such cases, the MLE will depend only on $g(\mathbf{z}^k)$, which we call the **sufficient statistic**

- if each observation is modeled as a function of the parameter plus some measurement noise,
$$\mathbf{z}(i) = \mathbf{h}(i, \mathbf{x}) + \mathbf{w}(i) \quad i \in \{1, \dots, k\}$$
where $\mathbf{E}[\mathbf{w}(i)] = \mathbf{0}$ and $\mathbf{cov}(\mathbf{w}(i)) = R$ for all i, then we can define the least squares cost
$$C_{\text{LS}}(\mathbf{x}, \mathbf{z}^k) = \frac{1}{2} \sum_{i=1}^{k} \big(\mathbf{z}(i) - \mathbf{h}(i, \mathbf{x})\big)^T R^{-1} \big(\mathbf{z}(i) - \mathbf{h}(i, \mathbf{x})\big)$$
which penalizes deviations of the observations from the deterministic model $\mathbf{h}(i, \mathbf{x})$. The **least squares estimator** (LSE) of $\mathbf{x}$ minimizes this cost:
$$\hat{\mathbf{x}}_{\text{LS}}(\mathbf{z}^k) = \arg\min_{\mathbf{x}} C_{\text{LS}}(\mathbf{x}, \mathbf{z}^k)$$
If $\mathbf{h}(i, \mathbf{x}) = H(i)\mathbf{x}$ for some matrices $H(i) \in \mathbf{R}^{n_z \times n_x}$, then this is called a **linear least squares** problem.

- if the measurement noises $\mathbf{w}(i)$ are Gaussian and the measurement maps are linear, then the MLE and LSE coincide

1.2 Bayesian estimation

- in **Bayesian** estimation (named after the British statistician Thomas Bayes, 1702-1761), we treat $\mathbf{x}$ as a random variable. The *a priori* distribution $p(\mathbf{x})$ encodes our prior assumptions or knowledge about $\mathbf{x}$.

- the primary tool of Bayesian estimation is the **posterior distribution**,
$$p(\mathbf{x}|\mathbf{z}^k) = \frac{p(\mathbf{z}^k|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z}^k)}$$
where $p(\mathbf{z}^k)$ can be (but rarely *needs* to be) calculated using the Law of Total Probability

- the **maximum *a posteriori* estimator** (MAPE) of $\mathbf{x}$ is
$$\hat{\mathbf{x}}_{\text{MAP}}(\mathbf{z}^k) = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{z}^k)$$

- define the mean squared error cost
$$C_{\text{MSE}}(\hat{\mathbf{x}}, \mathbf{z}^k) = \mathbf{E}[(\hat{\mathbf{x}} - \mathbf{x})^T(\hat{\mathbf{x}} - \mathbf{x})|\mathbf{z}^k]$$
then the **minimum mean squared error estimator** (MMSEE) of $\mathbf{x}$ is
$$\begin{aligned}\hat{\mathbf{x}}_{\text{MMSE}}(\mathbf{z}^k) &= \arg\min_{\hat{\mathbf{x}}} C_{\text{MSE}}(\hat{\mathbf{x}}, \mathbf{z}^k) \\ &= \mathbf{E}[\mathbf{x}|\mathbf{z}^k]\end{aligned}$$

- if the posterior distribution is Gaussian, then the MAP and MSE agree. This is because the mean and mode of a Gaussian *pdf* coincide.

1.3 Estimator precision

- the **estimation error** of an estimate $\hat{\mathbf{x}}$ of $\mathbf{x}$ is

$$\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$$

- the **bias** of $\hat{\mathbf{x}}$ is

$$\mathbf{bias}(\hat{\mathbf{x}}) = \mathbf{E}[\tilde{\mathbf{x}}]$$

  $\hat{\mathbf{x}}$ is called **unbiased** if

$$\begin{aligned} & \mathbf{bias}(\hat{\mathbf{x}}) = \mathbf{0} \\ \iff & \mathbf{E}[\hat{\mathbf{x}}] = \mathbf{x} \qquad \text{(Fisher)} \\ \iff & \mathbf{E}[\hat{\mathbf{x}}] = \mathbf{E}[\mathbf{x}] \quad \text{(Bayes)} \end{aligned}$$

  $\hat{\mathbf{x}}$ is called **asymptotically unbiased** if it's unbiased in the limit $k \to \infty$

- the **covariance** of $\hat{\mathbf{x}}$ is

$$\mathbf{cov}(\hat{\mathbf{x}}) = \mathbf{E}[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T] - \mathbf{E}[\tilde{\mathbf{x}}]\,\mathbf{E}[\tilde{\mathbf{x}}]^T$$

- the **mean squared error** (MSE) of $\hat{\mathbf{x}}$ is

$$\mathbf{MSE}(\hat{\mathbf{x}}) = \mathbf{E}[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T]$$

  For an unbiased estimator, $\mathbf{MSE}(\hat{\mathbf{x}}) = \mathbf{cov}(\hat{\mathbf{x}})$.

- $\hat{\mathbf{x}}$ is **consistent** if

$$\lim_{k\to\infty} \mathbf{MSE}(\hat{\mathbf{x}}) = 0$$

  "the estimator improves with the number of observations, and becomes perfect in the MSE sense as $k \to \infty$"

- in Fisher estimation, the **Fisher information matrix** (FIM) is

$$\begin{aligned} J_f &= -\,\mathbf{E}[\frac{\partial^2}{\partial \mathbf{x}^2} \ln p(\mathbf{z}^k|\mathbf{x})\Big|_{\mathbf{x}}] \\ &= \mathbf{E}[(\frac{\partial}{\partial \mathbf{x}} \ln p(\mathbf{z}^k|\mathbf{x})\Big|_{\mathbf{x}})^T (\frac{\partial}{\partial \mathbf{x}} \ln p(\mathbf{z}^k|\mathbf{x})\Big|_{\mathbf{x}})] \end{aligned}$$

  where $\mathbf{x}$ is the true (deterministic) parameter value, and expectation is taken over the randomness in $\mathbf{z}^k$.

- in Bayesian estimation, the FIM is

$$\begin{aligned} J_b &= -\,\mathbf{E}[\frac{\partial^2}{\partial \mathbf{x}^2} \ln p(\mathbf{x}, \mathbf{z}^k)] \\ &= \mathbf{E}[\big(\frac{\partial}{\partial \mathbf{x}} \ln p(\mathbf{z}^k|\mathbf{x})\big)^T \big(\frac{\partial}{\partial \mathbf{x}} \ln p(\mathbf{z}^k|\mathbf{x})\big)] \end{aligned}$$

  where expectation is taken over the randomness in both $\mathbf{z}^k$ and in $\mathbf{x}$, using the prior distribution $p(\mathbf{x})$.

- the **Cramér-Rao lower bound** (CRLB) of an *unbiased* estimator is

$$\mathbf{cov}(\hat{\mathbf{x}}) \succeq J^{-1}$$

  where $J = J_f$ or $J = J_b$, depending on the estimation approach. There's a different version of the CRLB for biased estimators.

- an estimator that achieves the CRLB is called **efficient**

2 Linear least squares parameter estimation

Consider the problem of estimating the parameter $\mathbf{x} \in \mathbf{R}^{n_x}$ based on the measurement model

$$\mathbf{z}(i) = H(i)\mathbf{x} + \mathbf{w}(i) \in \mathbf{R}^{n_z}$$

where, by assumption, $\mathbf{w}(i) \sim \mathcal{N}(\mathbf{0}, R)$ and $\mathbf{w}(i) \perp\!\!\!\perp \mathbf{x}$ for all i.

This section solves this problem in the case of a single observation with a Gaussian prior on $\mathbf{x}$, and then in the case of multiple observations with no prior. We solve the case of multiple observations by two approaches: batch and recursive.

2.1 Single observation, Gaussian prior

In the case of a single observation $\mathbf{z}$, the measurement model is

$$\mathbf{z} = H\mathbf{x} + \mathbf{w} \in \mathbf{R}^{n_z}$$

where $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, R)$, and $\mathbf{w} \perp\!\!\!\perp \mathbf{x}$.

We take the Bayesian approach, assuming a *Gaussian* prior distribution $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \bar{\mathbf{x}}, P_{xx})$, and derive the MAPE.

- the expectation and covariances of the measurement are

$$\begin{aligned} \mathbf{E}[\mathbf{z}] &\equiv \bar{\mathbf{z}} = H\bar{\mathbf{x}} \\ \mathbf{cov}(\mathbf{x}, \mathbf{z}) &\equiv P_{xz} = P_{xx}H^T \\ \mathbf{cov}(\mathbf{z}, \mathbf{z}) &\equiv P_{zz} = HP_{xx}H^T + R \end{aligned}$$

- the MAPE is

$$\hat{\mathbf{x}}_{\text{MAP}}(\mathbf{z}) = \bar{\mathbf{x}} + P_{xz}P_{zz}^{-1}(\mathbf{z} - \bar{\mathbf{z}})$$

  "the *a priori* mean plus a feedback term, based on the difference between the measurement and its expectation"

- the MAPE is unbiased with $\mathbf{cov}(\hat{\mathbf{x}}_{\text{MAP}}(\mathbf{z})) \equiv P_{\hat{x}\hat{x}}$, where

$$P_{\hat{x}\hat{x}} = P_{xx} - P_{xz}P_{zz}^{-1}P_{xz}^T$$

- **NB.** For the problem $\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{w}$, where $\mathbf{w}$ and $\mathbf{x}$ are general random vectors, *not necessarily Gaussian*, the estimator

$$\boxed{\begin{aligned} \hat{\mathbf{x}}(\mathbf{z}) &= \bar{\mathbf{x}} + P_{xz}P_{zz}^{-1}(\mathbf{z} - \bar{\mathbf{z}}) \\ P_{\hat{x}\hat{x}} &= P_{xx} - P_{xz}P_{zz}^{-1}P_{xz}^T \end{aligned}} \tag{1}$$

  is the **linear minimum mean-square error** (LMMSE) estimator of $\mathbf{x}$, i.e. the best unbiased estimator (in the MSE sense) of the form

$$\hat{\mathbf{x}}(\mathbf{z}) = A\mathbf{z} + \mathbf{b}$$

  for some $A \in \mathbf{R}^{n_x \times n_z}$ and $\mathbf{b} \in \mathbf{R}^{n_x}$. The LMMSE equations (1) are called the **fundamental equations of linear estimation.**

2.2 Multiple observations, no prior

In the case of multiple observations $\mathbf{z}(1), \ldots, \mathbf{z}(k) \in \mathbf{R}^{n_z}$, the measurement model is

$$\mathbf{z}(i) = H(i)\mathbf{x} + \mathbf{w}(i) \in \mathbf{R}^{n_z}$$

Assume that $\mathbf{w}(i) \sim \mathcal{N}(\mathbf{0}, R(i))$, $\mathbf{w}(i) \perp\!\!\!\perp \mathbf{x}$ for all i, and $\mathbf{E}[\mathbf{w}(i)\mathbf{w}(j)^T] = \mathbf{0}$ for all $i \neq j$.

We'll use the Fisher approach, assuming no prior information about $\mathbf{x}$, and derive the MLE. Since the noise is Gaussian and the measurement map is linear, the MLE is also the LSE.

Define the cost

$$C_{\mathrm{LS}}(k) = \frac{1}{2}\sum_{i=1}^{k} \big(\mathbf{z}(i) - H(i)\mathbf{x}\big)^T R(i)^{-1} \big(\mathbf{z}(i) - H(i)\mathbf{x}\big)$$

and the stacked vector of observations

$$\mathbf{z}^k = \begin{bmatrix} \mathbf{z}(1) \\ \vdots \\ \mathbf{z}(k) \end{bmatrix} \in \mathbf{R}^{kn_z}$$

The likelihood of $\mathbf{z}^k$ for a given $\mathbf{x}$ is

$$\begin{aligned} & p(\mathbf{z}^k|\mathbf{x}) = ce^{-C_{\mathrm{LS}}(k)} \\ \iff \quad & C_{\mathrm{LS}}(k) = \ln c - \ln p(\mathbf{z}^k|\mathbf{x})) \end{aligned}$$

where c is independent of $\mathbf{x}$.

Maximizing the likelihood is therefore equivalent to minimizing $C_{\mathrm{LS}}(k)$:

$$\hat{\mathbf{x}}_{\mathrm{ML}}(\mathbf{z}^k) = \arg\max_{\mathbf{x}} p(\mathbf{z}^k|\mathbf{x}) = \arg\min_{\mathbf{x}} C_{\mathrm{LS}}(k) \tag{2}$$

We'll solve problem (2) by two approaches: batch, where we lift the problem into a higher dimension and solve it on one shot, and recursive, where we iteratively solve k subproblems.

2.2.1 Batch solution

Define the lifted matrices

$$H^k = \begin{bmatrix} H(1) \\ \vdots \\ H(k) \end{bmatrix} \in \mathbf{R}^{kn_z \times n_x}, \quad \mathbf{w}^k = \begin{bmatrix} \mathbf{w}(1) \\ \vdots \\ \mathbf{w}(k) \end{bmatrix} \in \mathbf{R}^{kn_z}, \quad R^k = \begin{bmatrix} R(1) & & \\ & \ddots & \\ & & R(k) \end{bmatrix} \in \mathbf{R}^{kn_z \times kn_z}$$

Then

$$\mathbf{z}^k = H^k\mathbf{x} + \mathbf{w}^k$$

where $\mathbf{w}^k \sim \mathcal{N}(\mathbf{0}, R^k)$, and

$$C_{\mathrm{LS}}(k) = \frac{1}{2}\big(\mathbf{z}^k - H^k\mathbf{x}\big)^T(R^k)^{-1}\big(\mathbf{z}^k - H^k\mathbf{x}\big)$$

Minimizing $C_{\mathrm{LS}}(k)$ gives

$$\boxed{\hat{\mathbf{x}}_{\mathrm{ML}}(\mathbf{z}^k) = \big((H^k)^T(R^k)^{-1}H^k\big)^{-1}(H^k)^T(R^k)^{-1}\mathbf{z}^k}$$

The MLE is unbiased, with $\mathbf{cov}(\hat{\mathbf{x}}_{\mathrm{ML}}(\mathbf{z}^k)) \equiv P_{\hat{x}\hat{x}}$, where

$$\boxed{P_{\hat{x}\hat{x}} = \big((H^k)^T(R^k)^{-1}H^k\big)^{-1}}$$

NB. To ensure that $(H^k)^T(R^k)^{-1}H^k$ exists and is invertible, we require that $R^k \succ 0$ and that H^k have full column rank, i.e. $\mathrm{rank}(H^k) = n_x$.

2.2.2 On computing $\big((H^k)^T(R^k)^{-1}H^k\big)^{-1}$

Suppose you're given an H^k with goofy units – some measurements are in microns, others are in parsecs. Then $\big((H^k)^T(R^k)^{-1}H^k\big)$ will probably be ill-conditioned, i.e. the conditioning number

$$\mathrm{cond}\,\big((H^k)^T(R^k)^{-1}H^k\big) = \frac{\lambda_{\max}}{\lambda_{\min}}$$

will be large. Here $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalues of $\big((H^k)^T(R^k)^{-1}H^k\big)$, respectively.

Inverting an ill-conditioned matrix is a bad idea, because only

$$\sim 16 - \log_{10}\mathrm{cond}\,\big((H^k)^T(R^k)^{-1}H^k\big)$$

useful digits are retained in the calculation (for a computer with 16-digit precision).

Square root methods – Cholesky and QR factorization – avoid these difficulties.

1. **Cholesky factorization**

   Since $R^k \succ 0$, a Cholesky factorization R_a exists, where R_a is upper triangular and nonsingular and $R_a^T R_a = R^k$. Let $R_a^{-T} = (R_a^T)^{-1}$, and define

   $$\mathbf{z}_a = R_a^{-T}\mathbf{z}^k, \quad H_a = R_a^{-T}H^k, \quad \mathbf{w}_a = R_a^{-T}\mathbf{w}^k$$

   Under this transformation, the measurement model and cost become

   $$\mathbf{z}_a = H_a\mathbf{x} + \mathbf{w}_a$$
   $$C_{\mathrm{LS}}(k) = (\mathbf{z}_a - H_a\mathbf{x})^T(\mathbf{z}_a - H_a\mathbf{x})$$

   so we've exchanged our weighted LS problem for an unweighted one.

2. **QR factorization**

The QR factorization of H_a is

$$H_a = Q_b \begin{bmatrix} R_b \\ 0 \end{bmatrix}$$

where Q_b is orthonormal and $R_b \in \mathbf{R}^{n_x \times n_x}$ is upper triangular and, because H^k is full column rank, also invertible. Define

$$Q_b^T \mathbf{z}_a = \begin{bmatrix} \mathbf{z}_{b1} \\ \mathbf{z}_{b2} \end{bmatrix}, \quad Q_b^T \mathbf{w}_a = \begin{bmatrix} \mathbf{w}_{b1} \\ \mathbf{w}_{b2} \end{bmatrix}$$

then the measurement model and cost become

$$\begin{bmatrix} \mathbf{z}_{b1} \\ \mathbf{z}_{b2} \end{bmatrix} = \begin{bmatrix} R_b \\ 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{w}_{b1} \\ \mathbf{w}_{b2} \end{bmatrix}$$
$$C_{\mathrm{LS}}(k) = (\mathbf{z}_{b1} - R_b \mathbf{x})^T (\mathbf{z}_{b1} - R_b \mathbf{x}) + \mathbf{z}_{b2}^T \mathbf{z}_{b2}$$

Minimizing $C_{\mathrm{LS}}(k)$ gives the MLE and its covariance,

$$\boxed{\begin{aligned} \hat{\mathbf{x}}_{\mathrm{ML}}(\mathbf{z}^k) &= R_b^{-1} \mathbf{z}^k \\ P_{\hat{x}\hat{x}} &= R_b^{-1} R_b^{-T} \end{aligned}}$$

2.2.3 Recursive solution

Suppose you've taken k observations and used them to compute the MLE of $\mathbf{x}$ and its covariance. Could you incorporate the incremental observation

$$\mathbf{z}(k+1) = H(k+1)\mathbf{x} + \mathbf{w}(k+1)$$

without reprocessing the entire batch of data? The answer, unsurprisingly, is yes. There are simple update formulas for the estimator and its covariance.

For clarity, denote the MLE given k observations by $\hat{\mathbf{x}}_{\mathrm{ML}}(k, \mathbf{z}^k))$, and its error covariance by $\hat{P}(k, \mathbf{z}^k) \equiv \mathbf{cov}(\hat{\mathbf{x}}_{\mathrm{ML}}(k, \mathbf{z}^k))$. Our goal is to write $\hat{\mathbf{x}}_{\mathrm{ML}}(k+1, \mathbf{z}^{k+1})$ and $\hat{P}(k+1, \mathbf{z}^{k+1})$ in terms of $\hat{\mathbf{x}}_{\mathrm{ML}}(k, \mathbf{z}^k)$ and $\hat{P}(k, \mathbf{z}^k)$.

The augmented vector of observations is

$$\mathbf{z}^{k+1} = \begin{bmatrix} \mathbf{z}(1) \\ \vdots \\ \mathbf{z}(k) \\ \mathbf{z}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{z}^k \\ \mathbf{z}(k+1) \end{bmatrix}$$

Similarly,

$$H^{k+1} = \begin{bmatrix} H^k \\ H(k+1) \end{bmatrix}, \quad \mathbf{w}^{k+1} = \begin{bmatrix} \mathbf{w}^k \\ \mathbf{w}(k+1) \end{bmatrix}, \quad R^{k+1} = \begin{bmatrix} R^k & \\ & R(k+1) \end{bmatrix}$$

With these definitions (and after a lot of algebra), the cost becomes

$$\begin{aligned} C_{\mathrm{LS}}(k+1) &= C_{\mathrm{LS}}(k) + (\mathbf{z}(k+1) - H(k+1)\mathbf{x})^T R(k+1)^{-1}(\mathbf{z}(k+1) - H(k+1)\mathbf{x}) \\ &= \dots \\ &= (\mathbf{x} - \hat{\mathbf{x}}_{\mathrm{ML}}(k, \mathbf{z}^k))^T \hat{P}(k, \mathbf{z}^k)^{-1}(\mathbf{x} - \hat{\mathbf{x}}_{\mathrm{ML}}(k, \mathbf{z}^k)) \\ &\quad + (\mathbf{z}(k+1) - H(k+1)\mathbf{x})^T R(k+1)^{-1}(\mathbf{z}(k+1) - H(k+1)\mathbf{x}) + c \end{aligned}$$

where c is constant with respect to $\mathbf{x}$.

Minimizing $C_{\mathrm{LS}}(k+1)$ gives the estimator update formula

$$\boxed{\hat{\mathbf{x}}_{\mathrm{ML}}(k+1, \mathbf{z}^{k+1}) = \hat{\mathbf{x}}_{\mathrm{ML}}(k, \mathbf{z}^k) + W(k+1)\big(\mathbf{z}(k+1) - H(k+1)\hat{\mathbf{x}}_{\mathrm{ML}}(k, \mathbf{z}^k)\big)}$$

The MLE remains unbiased, and the covariance update formula is given by either of

$$\boxed{\begin{aligned} \hat{P}(k+1, \mathbf{z}^{k+1}) &= \big(\hat{P}(k, \mathbf{z}^k)^{-1} + H(k+1)^T R(k+1)^{-1} H(k+1)\big)^{-1} \\ &= \big(I - W(k+1)H(k+1)\big)\hat{P}(k, \mathbf{z}^k) \end{aligned}}$$

The **correction gain matrix** in the above formulas is given by either of

$$\boxed{\begin{aligned} W(k+1) &= \hat{P}(k, \mathbf{z}^k) H(k+1)^T \big(H(k+1)\hat{P}(k, \mathbf{z}^k)H(k+1)^T + R(k+1)\big)^{-1} \\ &= \hat{P}(k+1, \mathbf{z}^{k+1}) H(k+1)^T R(k+1)^{-1} \end{aligned}}$$

3 Nonlinear least squares parameter estimation

Consider the problem of estimating the parameter $\mathbf{x} \in \mathbf{R}^{n_x}$ based on the measurement model

$$\tilde{\mathbf{z}} = \tilde{\mathbf{h}}(\mathbf{x}) + \tilde{\mathbf{w}} \in \mathbf{R}^{n_z}$$

where, by assumption, $\tilde{\mathbf{w}} \sim \mathcal{N}(\mathbf{0}, R)$ and $\tilde{\mathbf{w}} \perp\!\!\!\perp \mathbf{x}$. (Why the tildes? Because we're going to transform the problem shortly.)

Real estimation problems are almost always **overdetermined**, meaning $n_z > n_x$. In overdetermined systems, no $\mathbf{x}$ will ever satisfy all the n_z equations in $\tilde{\mathbf{z}} = \tilde{\mathbf{h}}(\mathbf{x})$ exactly, so we need to choose an $\mathbf{x}$ that, on average, satisfies as many equations as possible, as closely as possible.

With that in mind, we define the nonlinear weighted least squares cost

$$C_{\text{NLS}}(\mathbf{x}) = \frac{1}{2}\big(\tilde{\mathbf{z}} - \tilde{\mathbf{h}}(\mathbf{x})\big)^T R^{-1}\big(\tilde{\mathbf{z}} - \tilde{\mathbf{h}}(\mathbf{x})\big)$$

Since $R \succ 0$, $C_{\text{NLS}}(\mathbf{x})$ is nonnegative for all $\mathbf{x}$ and zero iff $\tilde{\mathbf{z}} = \tilde{\mathbf{h}}(\mathbf{x})$. This suggests that minimizing $C_{\text{NLS}}(\mathbf{x})$ should give us a good estimate of $\mathbf{x}$.

The first step in solving a *weighted* least squares problem is to reformulate it as an *unweighted* least squares cost by Cholesky factorization as in Section 2.2.2. Let $R_a^T R_a = R$ and $R_a^{-T} = (R_a^{-1})^T$. Under the transformation $\mathbf{z} = R_a^{-T}\tilde{\mathbf{z}}$, $\mathbf{h}(\mathbf{x}) = R_a^{-T}\tilde{\mathbf{h}}(\mathbf{x})$, and $\mathbf{w} = R_a^{-T}\tilde{\mathbf{w}}$, the measurement model becomes

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{w} \in \mathbf{R}^{n_z}$$

where $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, I_{n_z})$ and $\mathbf{w} \perp\!\!\!\perp \mathbf{x}$. The transformed cost is now unweighted:

$$C_{\text{NLS}}(\mathbf{x}) = \frac{1}{2}\big(\mathbf{z} - \mathbf{h}(\mathbf{x})\big)^T\big(\mathbf{z} - \mathbf{h}(\mathbf{x})\big)$$

and the NLS estimator is

$$\hat{\mathbf{x}}_{\text{NLS}}(\mathbf{z}) = \arg\min_{\mathbf{x}} C_{\text{NLS}}(\mathbf{x})$$

3.1 Numerical solution

The first-order condition (FOC)

$$\Big(\frac{\partial}{\partial \mathbf{x}} C_{\text{NLS}}(\mathbf{x})\Big)^T\Big|_{\hat{\mathbf{x}}_{\text{NLS}}(\mathbf{z})} = \mathbf{0}$$

is necessary (but not sufficient) for the optimality of $\hat{\mathbf{x}}_{\text{NLS}}(\mathbf{z})$.

The Jacobian of $C_{\text{NLS}}(\mathbf{x})$ is

$$\frac{\partial}{\partial \mathbf{x}} C_{\text{NLS}}(\mathbf{x}) = -\big(\mathbf{z} - \mathbf{h}(\mathbf{x})\big)^T \frac{\partial \mathbf{h}}{\partial \mathbf{x}}$$

Let $D(\mathbf{x}) = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}$ and $\mathbf{f}(\mathbf{x}) = -D(\mathbf{x})^T(\mathbf{z} - \mathbf{h}(\mathbf{x}))$. Then the FOC is

$$\mathbf{f}(\hat{\mathbf{x}}_{\text{NLS}}(\mathbf{z})) = \mathbf{0}$$

which is a nonlinear root-finding problem.

3.1.1 Newton's method

The go-to algorithm for nonlinear root-finding problems is Newton's method, where we make an initial guess $\mathbf{x}_\text{g}$, linearize $\mathbf{f}(\hat{\mathbf{x}}_\text{NLS}(\mathbf{z}))$ about $\mathbf{x}_\text{g}$, solve the corresponding linear problem, update $\mathbf{x}_\text{g}$ and iterate. Assuming the initial guess is good, Newton's method will converge to the true solution, i.e. $\mathbf{x}_\text{g} \to \hat{\mathbf{x}}_\text{NLS}(\mathbf{z})$.

The linearization of $\mathbf{f}(\hat{\mathbf{x}}_\text{NLS}(\mathbf{z}))$ about $\mathbf{x}_\text{g}$ is

$$\begin{aligned} & \mathbf{f}(\hat{\mathbf{x}}_\text{NLS}(\mathbf{z})) \approx \mathbf{f}(\mathbf{x}_\text{g}) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\mathbf{x}_\text{g}} \big(\hat{\mathbf{x}}_\text{NLS}(\mathbf{z}) - \mathbf{x}_\text{g}\big) \\ \iff \quad & \mathbf{f}(\mathbf{x}_\text{g} + \Delta\mathbf{x}) \approx \mathbf{f}(\mathbf{x}_\text{g}) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\mathbf{x}_\text{g}} \Delta\mathbf{x} \end{aligned}$$

where $\Delta\mathbf{x} = \hat{\mathbf{x}}_\text{NLS}(\mathbf{z}) - \mathbf{x}_\text{g}$. Combining the FOC and Taylor approximation gives

$$\Delta\mathbf{x} = -\Big(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\mathbf{x}_\text{g}}\Big)^{-1} \mathbf{f}(\mathbf{x}_\text{g})$$

so the update equation is

$$\mathbf{x}_{\text{g, new}} = \mathbf{x}_\text{g} + \Delta\mathbf{x} = \mathbf{x}_\text{g} - \Big(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\mathbf{x}_\text{g}}\Big)^{-1} \mathbf{f}(\mathbf{x}_\text{g})$$

In the NLS problem,

$$\begin{aligned} \frac{\partial \mathbf{f}}{\partial \mathbf{x}} &= \frac{\partial}{\partial \mathbf{x}}\big(- D(\mathbf{x})^T(\mathbf{z} - \mathbf{h}(\mathbf{x}))\big) \\ &= -\frac{\partial}{\partial \mathbf{x}}\big(\frac{\partial \mathbf{h}}{\partial \mathbf{x}}^T (\mathbf{z} - \mathbf{h}(\mathbf{x}))\big) \\ &= -\frac{\partial^2 \mathbf{h}}{\partial \mathbf{x}^2}^T \big(\mathbf{z} - \mathbf{h}(\mathbf{x})\big) + \frac{\partial \mathbf{h}}{\partial \mathbf{x}}^T \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \\ &= -H(\mathbf{x})^T\big(\mathbf{z} - \mathbf{h}(\mathbf{x})\big) + D(\mathbf{x})^T D(\mathbf{x}) \end{aligned}$$

where $H(\mathbf{x}) = \frac{\partial^2 \mathbf{h}}{\partial \mathbf{x}^2}$ is the Hessian of $\mathbf{h}(\mathbf{x})$.

The update equation is therefore

$$\begin{aligned} \mathbf{x}_{\text{g, new}} &= \mathbf{x}_\text{g} + \Delta\mathbf{x} = \mathbf{x}_\text{g} - \Big(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\mathbf{x}_\text{g}}\Big)^{-1} \mathbf{f}(\mathbf{x}_\text{g}) \\ &= \mathbf{x}_\text{g} + \Big[- H(\mathbf{x}_\text{g})^T\big(\mathbf{z} - \mathbf{h}(\mathbf{x}_\text{g})\big) + D(\mathbf{x}_\text{g})^T D(\mathbf{x}_\text{g})\Big]^{-1} D(\mathbf{x}_\text{g})^T\big(\mathbf{z} - \mathbf{h}(\mathbf{x}_\text{g})\big) \end{aligned}$$

3.1.2 The Gauss-Newton method

Newton's method is powerful and general. In the neighborhood of the true solution, it converges extremely quickly. However, Newton's method poses two difficulties for practical implementation.

1. The Hessian $H(\mathbf{x}) = \frac{\partial^2 \mathbf{h}}{\partial \mathbf{x}^2}$ is costly to compute, and may lose its positive definiteness due to roundoff error.

2. For a bad initial guess, Newton's method may fail to converge to the global optimum.

The second problem is a fundamental difficulty of global optimization; see Section 3.2 for details. The first problem is more easily dealt with; Gauss's solution was to make two changes to Newton's method.

1. Ignore the term
$$-H(\mathbf{x}_{\mathrm{g}})\big(\mathbf{z} - \mathbf{h}(\mathbf{x}_{\mathrm{g}})\big)$$
when computing $\mathbf{x}_{\mathrm{g,\ new}}$.

   This is a plausible thing to do, since in the neighborhood of the true solution $\hat{\mathbf{x}}_{\mathrm{NLS}}(\mathbf{z})$, the term $\mathbf{z} - \mathbf{h}(\mathbf{x}_{\mathrm{g}})$) is small. Why? Because $\hat{\mathbf{x}}_{\mathrm{NLS}}(\mathbf{z})$, the minimizer of $C_{\mathrm{NLS}}(\mathbf{x})$, was specifically designed to make $\mathbf{z} - \mathbf{h}(\hat{\mathbf{x}}_{\mathrm{NLS}}(\mathbf{z}))$ as small as possible.

2. Scale $\Delta\mathbf{x}$ in the update equation by a factor $\alpha \in (0, 1]$.

   Here α is chosen such that each iteration incurs a lower cost, i.e.
$$C_{\mathrm{NLS}}(\mathbf{x}_{\mathrm{g,\ new}}) < C_{\mathrm{NLS}}(\mathbf{x}_{\mathrm{g}})$$

With these modifications, the Gauss-Newton update equation is

$$\mathbf{x}_{\mathrm{g,\ new}} = \mathbf{x}_{\mathrm{g}} + \alpha\Big[D(\mathbf{x}_{\mathrm{g}})^T D(\mathbf{x}_{\mathrm{g}})\Big]^{-1} D(\mathbf{x}_{\mathrm{g}})^T\big(\mathbf{z} - \mathbf{h}(\mathbf{x}_{\mathrm{g}})\big)$$

NB. The best practice for calculating the inverse matrix square $(D(\mathbf{x}_{\mathrm{g}})^T D(\mathbf{x}_{\mathrm{g}}))^{-1}$ is to use Cholesky and QR factorizations as in Section 2.2.2.

How to choose α in the Gauss-Newton algorithm? There's an optimal but computationally intensive method (see Gill, Murray and Wright: "optimal line searches"). A simpler, effective approach is to start a loop with $\alpha = 1$ and keep halving α until it yields a cost decrease.

3.1.3 The Levenberg-Marquardt method

Another modification of Newton's method is the Levenberg-Marquardt method. It's similar to the Gauss-Newton method, except rather than scaling $\Delta\mathbf{x}_{\mathrm{g}}$ by a parameter α, we insert a matrix λI into the matrix inversion.

The Levenberg-Marquardt update equation is

$$\mathbf{x}_{\mathrm{g,\ new}} = \mathbf{x}_{\mathrm{g}} + \Big[D(\mathbf{x}_{\mathrm{g}})^T D(\mathbf{x}_{\mathrm{g}}) + \lambda I\Big]^{-1} D(\mathbf{x}_{\mathrm{g}})^T\big(\mathbf{z} - \mathbf{h}(\mathbf{x}_{\mathrm{g}})\big)$$

If $\alpha = 1$ and $\lambda = 0$, then the Gauss-Newton and Levenberg-Marquardt update equations agree.

The challenge of implementing the Levenberg-Marquardt method is deciding what λ to use. It can be shown that there always exists a $\lambda > 0$ (possibly very large) that yields a cost decrease, but finding a good λ is not as easy as finding a good α in Gauss-Newton. There are sophisticated rules for updating λ (see Gill, Murray and Wright "trust region" for details).

3.2 Solution uniqueness and observability

The nonlinear least squares estimation problem is called **observable** if the problem

$$\hat{\mathbf{x}}_{\text{NLS}}(\mathbf{z}) = \arg\min_{\mathbf{x}} C_{\text{NLS}}(\mathbf{x}) = \arg\min_{\mathbf{x}} \frac{1}{2}(\mathbf{z} - \mathbf{h}(\mathbf{x}))^T(\mathbf{z} - \mathbf{h}(\mathbf{x}))$$

has a unique global minimum. Since $\mathbf{h}(\mathbf{x})$ is an arbitrary nonlinear function of $\mathbf{x}$, $C_{\text{NLS}}(\mathbf{x})$ is not generally convex and the NLS problem is not generally observable.

In unobservable problems, the quality of the numerical solution depends strongly on the initial guess. The art of NLS batch estimation often boils down to using physics and intuition to arrive at a "good" initial guess: one that will result, with high probability, in convergence to the global minimum.

In unobservable systems, we can still verify that the result of our root-finding algorithm is locally unique. A local minimum $\hat{\mathbf{x}}$ of $C_{\text{NLS}}(\mathbf{x})$ is called **locally unique** if there exists an $\epsilon > 0$ such that for all $\mathbf{x} \in \{\mathbf{x} \mid \|\mathbf{x} - \hat{\mathbf{x}}\| < \epsilon\}$, $C_{\text{NLS}}(\mathbf{x}) > C_{\text{NLS}}(\hat{\mathbf{x}})$. In English: every point in the punctured ϵ-ball centered at $\hat{\mathbf{x}}$ incurs a cost that's strictly greater than the cost of $\hat{\mathbf{x}}$.

A sufficient condition for the local uniqueness of $\hat{\mathbf{x}}$ is that the Hessian of $C_{\text{NLS}}(\mathbf{x})$ be positive definite:

$$\frac{\partial^2}{\partial \mathbf{x}^2} C_{\text{NLS}}(\mathbf{x})\Big|_{\hat{\mathbf{x}}} \succ 0 \quad \Rightarrow \quad \hat{\mathbf{x}} \text{ locally unique}$$

Expanding this condition,

$$\begin{aligned}\frac{\partial^2}{\partial \mathbf{x}^2} C_{\text{NLS}}(\mathbf{x}) &= \frac{\partial^2}{\partial \mathbf{x}^2} \frac{1}{2}(\mathbf{z} - \mathbf{h}(\mathbf{x}))^T(\mathbf{z} - \mathbf{h}(\mathbf{x})) \\ &= -H(\mathbf{x})(\mathbf{z} - \mathbf{h}(\mathbf{x})) + D(\mathbf{x})^T D(\mathbf{x})\end{aligned}$$

where $D(\mathbf{x})$ and $H(\mathbf{x})$ are the Jacobian and Hessian of $\mathbf{h}(\mathbf{x})$, respectively. So

$$-H(\hat{\mathbf{x}})(\mathbf{z} - \mathbf{h}(\hat{\mathbf{x}})) + D(\hat{\mathbf{x}})^T D(\hat{\mathbf{x}}) \succ 0 \quad \Rightarrow \quad \hat{\mathbf{x}} \text{ locally unique} \tag{3}$$

To rigorously determine the local uniqueness of $\hat{\mathbf{x}}$, we need to check condition (3). However, $H(\hat{\mathbf{x}})$ is costly to compute. In many practical cases, we make the simplifying assumption that the term $H(\hat{\mathbf{x}})(\mathbf{z} - \mathbf{h}(\hat{\mathbf{x}}))$ is negligible. This assumption gives the "first-cut" uniqueness check

$$D(\hat{\mathbf{x}})^T D(\hat{\mathbf{x}}) \succ 0 \quad \Longleftrightarrow \quad \text{rank}(D(\hat{\mathbf{x}})) = n_x \quad \Rightarrow \quad \hat{\mathbf{x}} \text{ locally unique}$$

We can justify neglecting the $H(\hat{\mathbf{x}})(\mathbf{z} - \mathbf{h}(\hat{\mathbf{x}}))$ term in two ways:

1. At a local minimum $\hat{\mathbf{x}}$, $\mathbf{z} - \mathbf{h}(\hat{\mathbf{x}})$ should be small. This is the same reason we neglect this term in the Gauss-Newton method.

2. If $D(\hat{\mathbf{x}})^T D(\hat{\mathbf{x}}) \not\succ 0$, the system likely has problems for other reasons. This is a good check to do regardless.

3.3 Covariance matrix

Given a unique global minimum $\hat{\mathbf{x}}_{\text{NLS}}(\mathbf{z})$ of $C_{\text{NLS}}(\mathbf{x})$, the estimator error covariance is

$$\begin{aligned} P_{\hat{x}\hat{x}} &= \mathbf{E}[(\mathbf{x} - \hat{\mathbf{x}}_{\text{NLS}}(\mathbf{z}))(\mathbf{x} - \hat{\mathbf{x}}_{\text{NLS}}(\mathbf{z}))^T] \\ &\equiv J^{-1} \\ &\approx \left[D(\hat{\mathbf{x}}_{\text{NLS}}(\mathbf{z}))^T D(\hat{\mathbf{x}}_{\text{NLS}}(\mathbf{z}))\right]^{-1} \end{aligned}$$

where $D(\hat{\mathbf{x}}_{\text{NLS}}(\mathbf{z}))^T D(\hat{\mathbf{x}}_{\text{NLS}}(\mathbf{z}))$ approximates J, the FIM. The inverse of the matrix square is best calculated using QR factorization as in Section 2.2.2.

4 Stochastic linear dynamical systems

A quick recap of stochastic linear systems follows, as a precursor to state estimation.

4.1 Continuous-time

Consider the continuous-time linear stochastic dynamical system

$$\begin{aligned} \dot{\mathbf{x}}(t) &= A(t)\mathbf{x}(t) + B(t)\mathbf{u}(t) + D(t)\tilde{\mathbf{v}}(t) \\ \mathbf{z}(t) &= C(t)\mathbf{x}(t) + \tilde{\mathbf{w}}(t) \end{aligned} \tag{4}$$

where

- $\mathbf{x}(t) \in \mathbf{R}^{n_x}$ is the **state**
- $\mathbf{u}(t) \in \mathbf{R}^{n_u}$ is the **input** or **control**
- $\tilde{\mathbf{v}}(t) \in \mathbf{R}^{n_v}$ is the **disturbance** or **process noise**
- $\mathbf{z}(t) \in \mathbf{R}^{n_z}$ is the **output** or **measurement**
- $\tilde{\mathbf{w}}(t) \in \mathbf{R}^{n_z}$ is the **noise** or **measurement noise**

4.1.1 Solution

Given an initial condition $\mathbf{x}(t_0)$, the solution to (4) is

$$\mathbf{x}(t) = F(t, t_0)\mathbf{x}(t_0) + \int_{t_0}^{t} F(t, \tau)\big[B(\tau)\mathbf{u}(\tau) + D(\tau)\tilde{\mathbf{v}}(\tau)\big]\mathrm{d}\tau$$

where $F(t, \tau)$ is the **state transition matrix** (STM) from τ to t. The STM solves the matrix initial value problem (IVP)

$$\begin{aligned} \frac{\partial}{\partial t}F(t, \tau) &= A(t)F(t, \tau) \\ F(\tau, \tau) &= I \end{aligned} \tag{5}$$

Solving this IVP is the only way to produce the STM for the general, time-varying system (4).

Properties of the STM:

$$\begin{aligned} F(t_2, t_0) &= F(t_1, t_0)F(t_2, t_1) \\ F(t, t_0) &= F(t_0, t)^{-1} \end{aligned}$$

In the special case of **time-invariant** open-loop dynamics, the STM is the matrix exponential:

$$A(t) \equiv A \text{ for all } t \in [0, \tau] \quad \Rightarrow \quad F(t, \tau) = F(t - \tau, 0) = e^{A(t-\tau)}$$

which can be calculated by series expansion, Laplace transform, or a dozen other ways (*expm(A)* in Matlab).

4.1.2 Mean and covariance of the state

Consider system (4), driven by (non-stationary, non-zero-mean) white process noise,

$$\mathbf{E}[\tilde{\mathbf{v}}(t)] = \bar{\tilde{\mathbf{v}}}(t)$$
$$\mathbf{E}[(\tilde{\mathbf{v}}(t) - \bar{\tilde{\mathbf{v}}}(t))(\tilde{\mathbf{v}}(\tau) - \bar{\tilde{\mathbf{v}}}(\tau))^T] = V(t)\delta(t-\tau)$$

The mean of the state evolves according to

$$\begin{aligned}
\bar{\mathbf{x}}(t) &= \mathbf{E}[\mathbf{x}(t)] \\
&= F(t,t_0)\bar{\mathbf{x}}(t_0) + \int_{t_0}^{t} F(t,\tau)\big[B(\tau)\mathbf{u}(\tau) + D(\tau)\bar{\tilde{\mathbf{v}}}(\tau)\big]\mathrm{d}\tau \\
\Rightarrow \quad \dot{\bar{\mathbf{x}}}(t) &= A(t)\bar{\mathbf{x}}(t) + B(t)\mathbf{u}(t) + D(t)\bar{\tilde{\mathbf{v}}}(t)
\end{aligned}$$

The covariance of the state evolves according to

$$\begin{aligned}
P_{xx}(t) &= \mathbf{E}[(\mathbf{x}(t) - \bar{\mathbf{x}}(t))(\mathbf{x}(t) - \bar{\mathbf{x}}(t))^T] \\
&= F(t,t_0)P_{xx}(t,t_0)F(t,t_0)^T + \int_{t_0}^{t} F(t,\tau)D(\tau)V(\tau)D(\tau)^T F(t,\tau)\mathrm{d}\tau \\
\Rightarrow \quad \dot{P}_{xx}(t) &= A(t)P_{xx}(t) + P_{xx}(t)A(t)^T + D(t)V(t)D(t)^T
\end{aligned}$$

In the special case of

1. stable unforced dynamics: $\lambda_i < 0$ for all eigenvalues λ_i of A
2. time-invariant dynamics: $A(t) \equiv A$, $D(t) \equiv D$, $V(t) \equiv V$

the state covariance will tend to the steady-state value P_{xx}^{ss} as $t \to \infty$. That value is given by the Lyapunov equation

$$AP_{xx}^{\text{ss}} + P_{xx}^{\text{ss}}A^T + DVD^T = 0$$

which is solvable in Matlab by $P_{xx}^{\text{ss}} = \text{lyap}(A, DVD^T)$.

4.2 Discrete-time

4.2.1 Discretization of dynamics

If the control $\mathbf{u}(t)$ in (4) is a **zero-order hold** (staircase), i.e. $\mathbf{u}(t) = \mathbf{u}(t_k)$ for all $t \in [t_k, t_{k+1})$, then we can develop an equivalent discrete-time model. For simplicity, let $\mathbf{x}(t_k) \equiv \mathbf{x}(k)$, $\mathbf{u}(t_k) \equiv \mathbf{u}(k)$, $\mathbf{v}(t_k) \equiv \mathbf{v}(k)$, $F(t_{k+1}, t_k) \equiv F(k)$, and $G(t_{k+1}, t_k) \equiv G(k)$. Then the discrete-time dynamics are

$$\mathbf{x}(k+1) = F(k)\mathbf{x}(k) + G(k)\mathbf{u}(k) + \mathbf{v}(k)$$

where

$$G(k) = \int_{t_k}^{t_{k+1}} F(t_{k+1}, \tau) B(\tau) \mathrm{d}\tau$$

$$\mathbf{E}[\mathbf{v}(k)] = \bar{\mathbf{v}}(k) = \int_{t_k}^{t_{k+1}} F(t_{k+1}, \tau) D(\tau) \bar{\bar{\mathbf{v}}}(\tau) \mathrm{d}\tau$$

$$\begin{aligned} \mathbf{E}[(\mathbf{v}(k) - \bar{\mathbf{v}}(k))(\mathbf{v}(j) - \bar{\mathbf{v}}(j))^T] &= \delta_{jk} Q(k) \\ &= \delta_{jk} \int_{t_k}^{t_{k+1}} F(t_{k+1}, \tau) D(\tau) V(\tau) D(\tau)^T F(t_{k+1}, \tau)^T \mathrm{d}\tau \end{aligned}$$

and $F(k)$ is the STM from t_k to t_{k+1}.

For a general time-varying stochastic system, the process of discretization involves calculating the above integrals. For real-time control, that means online numerical integration, for instance with Runge-Kutta methods. This may or may not be feasible, depending on the application and the available computing power.

In the special case of

1. constant step sizes: $t_{k+1} - t_k = \Delta t$ for all k
2. time-invariant dynamics in the underlying continuous-time system: $A(t) \equiv A$, $B(t) \equiv B$, $D(t) \equiv D$, and $V(t) \equiv V$ for all t

the corresponding discrete-time system is also time-invariant:

$$\begin{aligned} F(k) \equiv F &= e^{A\Delta t} \\ G(k) \equiv G &= \int_0^{\Delta t} e^{A(\Delta t - \tau)} B \mathrm{d}\tau \\ Q(k) \equiv Q &= \int_0^{\Delta t} e^{A(\Delta t - \tau)} D V D^T e^{A^T(\Delta t - \tau)} \mathrm{d}\tau \end{aligned}$$

These integrals can be done offline, which reduces the real-time calculations to matrix algebra.

4.2.2 Discretization of observations

The measurements in (4) can also be discretized, although sampling data from a continuous-time sensor requires an anti-aliasing filter.

Consider the case of zero-mean, continuous white sensor noise:

$$\begin{aligned} \mathbf{E}[\tilde{\mathbf{w}}(t)] &= \mathbf{0} \\ \mathbf{E}[\tilde{\mathbf{w}}(t)\tilde{\mathbf{w}}(\tau)^T] &= \tilde{R}(t)\delta(t - \tau) \end{aligned}$$

A good anti-aliasing filter produces (approximately) a noise sequence

$$\mathbf{w}(k) = \frac{1}{2\Delta t}\int_{t_k-\Delta t}^{t_k+\Delta t} \tilde{\mathbf{w}}(\tau)\mathrm{d}\tau$$
$$\Rightarrow \quad \mathbf{E}[\mathbf{w}(k)\mathbf{w}(k)^T] \approx \frac{1}{2\Delta t}\tilde{R}(t_k) = R(k)$$

which gives the discrete-time measurement model

$$\mathbf{z}(k) = H(k)\mathbf{x}(k) + \mathbf{w}(k)$$

where

$$H(k) = \frac{1}{\Delta t}\int_{t_{k-1}}^{t_k} F(t_k, t_{k-1})C(t)\mathrm{d}t$$

4.2.3 Solution

Consider general discrete-time linear stochastic system

$$\begin{aligned} \mathbf{x}(k+1) &= F(k)\mathbf{x}(k) + G(k)\mathbf{u}(k) + \mathbf{v}(k) \\ \mathbf{z}(k) &= H(k)\mathbf{x}(k) + \mathbf{w}(k) \end{aligned} \tag{6}$$

The solution to (6) is

$$\mathbf{x}(k) = F(k-1)F(k-2)\cdots F(0)\mathbf{x}(0) + \sum_{i=0}^{k-1} F(k-1)F(k-2)\cdots F(i+1)(G(i)\mathbf{u}(i) + \mathbf{v}(i))$$

If $\mathbf{v}(k)$, $\mathbf{w}(k)$ and $\mathbf{x}(0)$ are mutually independent and $\mathbf{v}(k)$ and $\mathbf{w}(k)$ are white, then the mean and covariance of the state evolve according to

$$\begin{aligned} \bar{\mathbf{x}}(k+1) &= F(k)\bar{\mathbf{x}}(k) + G(k)\mathbf{u}(k) + \bar{\mathbf{v}}(k) \\ P_{xx}(k+1) &= F(k)P_{xx}(k)F(k)^T + Q(k) \end{aligned}$$

In the special case of

1. time-invariant dynamics: $F(k) \equiv F$ and $Q(k) \equiv Q$ for all k
2. stable unforced dynamics: $|\lambda_i| < 1$ for all eigenvalues λ_i of F

the state covariance will tend to the steady-state value P_{xx}^{ss} as $k \to \infty$. That value is given by the discrete-time Lyapunov equation

$$P_{xx}^{\text{ss}} = FP_{xx}^{\text{ss}}F^T + Q$$

which is solvable in Matlab by $P_{xx}^{\text{ss}} = \text{dylap}(F, Q)$.

4.3 Pre-whitening colored noise

If $\tilde{\mathbf{v}}(t)$ is white noise, then the state $\mathbf{x}(t)$ is Markovian and the above results (both continuous- and discrete-time) hold. If $\tilde{\mathbf{v}}(t)$ is not white noise, then they generally do not. In the case of colored noise, the typical approach is to add some clever variables to $\mathbf{x}(t)$ to produce a system driven by a different noise sequence, now white. This process of "Markov-ization" is one application of **state augmentation**.

Consider system (4) driven by stationary, colored noise with autocorrelation

$$\mathbf{E}[\tilde{\mathbf{v}}(t)\tilde{\mathbf{v}}(\tau)^T] = R_{\tilde{\mathbf{v}}\tilde{\mathbf{v}}}(t-\tau)$$

and power spectral density

$$S_{\tilde{\mathbf{v}}\tilde{\mathbf{v}}}(\omega) = \int_{-\infty}^{\infty} R_{\tilde{\mathbf{v}}\tilde{\mathbf{v}}}(\tau) e^{-i\omega\tau} \mathrm{d}\tau$$

Suppose that by studying the structure of $\tilde{\mathbf{v}}(t)$, we can derive[1] the linear model

$$\dot{\tilde{\mathbf{v}}}(t) = C(t)\tilde{\mathbf{v}}(t) + \mathbf{q}(t)$$

where $\mathbf{q}(t)$ is stationary white noise with autocorrelation

$$\mathbf{E}[\mathbf{q}(t)\mathbf{q}(\tau)^T] = V\delta(t-\tau)$$

Then the state of the augmented system

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\tilde{\mathbf{v}}}(t) \end{bmatrix} = \begin{bmatrix} A(t) & D(t) \\ 0 & C(t) \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ \tilde{\mathbf{v}}(t) \end{bmatrix} + \begin{bmatrix} B(t) \\ 0 \end{bmatrix} \mathbf{u}(t) + \begin{bmatrix} 0 \\ I \end{bmatrix} \mathbf{q}(t)$$

is a Markov process.

[1] How to derive such a model? See *spectral factorization* in Bar-Shalom, e.g. the example on p. 193-4.

5 State estimation in linear Gaussian dynamical systems

So far, these notes have dealt with static parameter estimation. In this section, we shift our focus to estimating the state of a linear dynamical system.

There are three generic state estimation problems in dynamical systems:

1. **filtering:** given observations $\mathbf{z}^k$, find an estimator $\hat{\mathbf{x}}(k|\mathbf{z}^k)$ of the current state $\mathbf{x}(k)$
2. **smoothing:** given observations $\mathbf{z}^k$, find an estimator $\hat{\mathbf{x}}(j|\mathbf{z}^k)$ of the state $\mathbf{x}(j)$ at some previous time $j < k$ (a non-causal problem)
3. **prediction:** given observations $\mathbf{z}^k$, find an estimator $\mathbf{x}_k(j|\mathbf{z}^k)$ of the state $\mathbf{x}(j)$ at some future time $j > k$

In general, prediction is a harder problem than filtering, which is a harder problem than smoothing. In terms of algorithm performance, smoothing > filtering > prediction.

5.1 The LG system

Consider the discrete-time stochastic linear time-varying dynamical system

$$\begin{aligned}\mathbf{x}(k+1) &= F(k)\mathbf{x}(k) + G(k)\mathbf{u}(k) + \Gamma(k)\mathbf{v}(k), \quad k \in \mathcal{T}_0 \\ \mathbf{z}(k) &= H(k)\mathbf{x}(k) + \mathbf{w}(k), \quad k \in \mathcal{T}_1\end{aligned}$$

where $\mathcal{T}_0 = \{0, 1, 2, 3, \ldots, N-1\}$ and $\mathcal{T}_1 = \{1, 2, 3, \ldots, N\}$.

Let the process and measurement noise sequences be zero-mean, Gaussian and white:

$$\begin{aligned}\mathbf{E}[\mathbf{v}(k)] &= \mathbf{0} \\ \mathbf{E}[\mathbf{v}(k)\mathbf{v}(j)^T] &= \delta_{jk}Q(k) \text{ for all } j, k \in \mathcal{T}_0 \\ \mathbf{E}[\mathbf{w}(k)] &= \mathbf{0} \\ \mathbf{E}[\mathbf{w}(k)\mathbf{w}(j)^T] &= \delta_{jk}R(k) \text{ for all } j, k \in \mathcal{T}_1\end{aligned}$$

Let the initial state $\mathbf{x}(0)$ be Gaussian with known mean $\hat{\mathbf{x}}(0)$ and covariance $P(0)$:

$$\begin{aligned}\mathbf{E}[\mathbf{x}(0)] &= \hat{\mathbf{x}}(0) \\ \mathbf{E}[(\mathbf{x}(0) - \hat{\mathbf{x}}(0))(\mathbf{x}(0) - \hat{\mathbf{x}}(0)^T)] &= P(0)\end{aligned}$$

Finally, let all the random variables be mutually uncorrelated:

$$\begin{aligned}\mathbf{E}[\mathbf{w}(k)\mathbf{v}(j)^T] &= 0 \text{ for all } j \in \mathcal{T}_0, k \in \mathcal{T}_1 \\ \mathbf{E}[\mathbf{w}(k)(\mathbf{x}(0) - \hat{\mathbf{x}}(0))^T] &= 0 \text{ for all } k \in \mathcal{T}_1 \\ \mathbf{E}[\mathbf{v}(k)(\mathbf{x}(0) - \hat{\mathbf{x}}(0))^T] &= 0 \text{ for all } k \in \mathcal{T}_0\end{aligned}$$

This is called the linear Gaussian (LG) system: linear dynamics, Gaussian randomness. It's an extremely common model because it's

1. **tractable:** linear systems and Gaussian random variables are the easiest dynamical systems and random variables to work with

2. **broadly applicable:** nonlinear systems can be linearized, and the CLT justifies treating many (not all, but many) random variables as Gaussian

Since linear transformation preserves normality, the LG system is a Gauss-Markov process. The optimal MMSE and MAP estimator for the LG system is the **Kalman filter** (KF). The KF is a recursive algorithm that uses feedback on the measurement prediction error to continually update the state estimate.

5.2 Definitions and notation

For each of the basic quantities in the Kalman filter, there are a few different notations used in the estimation literature. These notes will use the notation on the right-hand side.

- current estimate
$$\mathbf{E}[\mathbf{x}(k)|\mathbf{z}^k] \equiv \hat{\mathbf{x}}(k|\mathbf{z}^k) \equiv \hat{\mathbf{x}}(k|k) \equiv \hat{\mathbf{x}}(k)$$

- covariance of current estimate
$$\begin{aligned}\mathbf{E}[(\mathbf{x}(k) - \hat{\mathbf{x}}(k|\mathbf{z}^k))(\mathbf{x}(k) - \hat{\mathbf{x}}(k|\mathbf{z}^k))^T \mid \mathbf{z}^k] \\ \equiv P(k|\mathbf{z}^k) \equiv P(k|k) \equiv P(k)\end{aligned}$$

- predicted state
$$\mathbf{E}[\mathbf{x}(k+1)|\mathbf{z}^k] \equiv \hat{\mathbf{x}}(k+1|\mathbf{z}^k) \equiv \hat{\mathbf{x}}(k+1|k) \equiv \bar{\mathbf{x}}(k+1)$$

- covariance of predicted state
$$\begin{aligned}\mathbf{E}[(\mathbf{x}(k+1) - \hat{\mathbf{x}}(k+1|\mathbf{z}^k))(\mathbf{x}(k+1) - \hat{\mathbf{x}}(k+1|\mathbf{z}^k))^T \mid \mathbf{z}^k] \\ \equiv P(k+1|\mathbf{z}^k) \equiv P(k+1|k) \equiv \bar{P}(k+1)\end{aligned}$$

- predicted measurement
$$\mathbf{E}[\mathbf{z}(k+1)|\mathbf{z}^k] \equiv \hat{\mathbf{z}}(k+1|\mathbf{z}^k) \equiv \hat{\mathbf{z}}(k+1|k) \equiv \bar{\mathbf{z}}(k+1)$$

- covariance of predicted measurement
$$\begin{aligned}\mathbf{E}[(\mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1|\mathbf{z}^k))(\mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1|\mathbf{z}^k))^T \mid \mathbf{z}^k] \\ \equiv P_{zz}(k+1|\mathbf{z}^k) \equiv P_{zz}(k+1|k) \equiv \bar{P}_{zz}(k+1)\end{aligned}$$

- covariance between predicted state and predicted measurement
$$\begin{aligned}\mathbf{E}[(\mathbf{x}(k+1) - \hat{\mathbf{x}}(k+1|\mathbf{z}^k))(\mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1|\mathbf{z}^k))^T \mid \mathbf{z}^k] \\ \equiv P_{xz}(k+1|\mathbf{z}^k) \equiv P_{xz}(k+1|k) \equiv \bar{P}_{xz}(k+1)\end{aligned}$$

5.3 The KF algorithm

Given the mean $\hat{\mathbf{x}}(0)$ and covariance $P(0)$ of the initial state, the KF algorithm is

1. set $k = 0$
2. **dynamic propagation**
   (a) predict the next state
   $$\bar{\mathbf{x}}(k+1) = F(k)\hat{\mathbf{x}}(k) + G(k)\mathbf{u}(k)$$
   (b) calculate the covariance of this prediction
   $$\bar{P}(k+1) = F(k)P(k)F(k)^T + \Gamma(k)Q(k)\Gamma(k)^T$$
3. **measurement update**
   (a) calculate the measurement error covariance
   $$\bar{P}_{zz}(k+1) \equiv S(k+1) = H(k+1)\bar{P}(k+1)H(k+1)^T + R(k+1)$$
   (b) calculate the filter gain
   $$W(k+1) = \bar{P}(k+1)H(k+1)^T S(k+1)^{-1}$$
   (c) measure $\mathbf{z}(k+1)$ and calculate the innovation
   $$\boldsymbol{\nu}(k+1) = \mathbf{z}(k+1) - H(k+1)\bar{\mathbf{x}}(k+1)$$
   (d) update the state estimate
   $$\hat{\mathbf{x}}(k+1) = \bar{\mathbf{x}}(k+1) + W(k+1)\boldsymbol{\nu}(k+1)$$
   (e) update the state estimate error covariance
   $$P(k+1) = \bar{P}(k+1) - W(k+1)S(k+1)W(k+1)^T$$
4. increment k and return to step 2

Note that $\bar{P}(k+1)$, $S(k+1)$, $W(k+1)$ and $P(k+1)$ depend only on $P(0)$ and the system matrices, so they can be calculated offline.

5.3.1 Interpretation of the filter gain

Note that the filter gain

$$\begin{aligned} W(k+1) &= \bar{P}(k+1)H(k+1)S(k+1)^{-1} \\ &= \bar{P}(k+1)H(k+1)\big[H(k+1)\bar{P}(k+1)H(k+1)^T + R(k+1)\big]^{-1} \end{aligned}$$

is "small" (in some matrix sense) if the covariance of the predicted state is "small." Interpretation: if the state prediction is very accurate, then the estimator update doesn't change much.

On the other hand, $W(k+1)$ is large if $R(k+1)$, the covariance of the measurement noise, is small (meaning its inverse is large). Interpretation: if the measurements are very accurate, then the estimator update makes a big change.

The idea here is that the filter gain weights the estimator update based on the relative accuracies of the estimator and the measurements.

5.3.2 Alternate formulas

There are a couple of alternate formulas for the estimator error covariance update:

$$\begin{aligned} P(k+1) &= \big[\bar{P}(k+1)^{-1} + H(k+1)^T R(k+1)^{-1} H(k+1)\big]^{-1} \\ &= \big[I - W(k+1)H(k+1)\big]\bar{P}(k+1)\big[I - W(k+1)H(k+1)\big]^T \\ &\quad + W(k+1)R(k+1)W(k+1)^T \end{aligned}$$

These are useful for numerically stable covariance updates when the measurement is very accurate. In that case, the formula given in the KF algorithm,

$$P(k+1) = \bar{P}(k+1) - W(k+1)S(k+1)W(k+1)^T$$

suffers from roundoff error because $W(k+1)S(k+1)W(k+1)^T \approx \bar{P}(k+1)$.

There's also an alternate formula for the gain matrix,

$$W(k+1) = P(k+1)H(k+1)^T R(k+1)^{-1}$$

which avoids inverting $S(k+1)$.

5.4 Some KF properties

5.4.1 Error stability

If $\mathbf{v}(k) \equiv \mathbf{0}$ for all $k \in \mathcal{T}_0$ and $\mathbf{w}(k) \equiv \mathbf{0}$ for all $k \in \mathcal{T}_1$, and if the LG system is detectable, then the estimation error is BIBO stable.

To see this, define the estimation error $\mathbf{e}(k) = \mathbf{x}(k) - \hat{\mathbf{x}}(k)$. Then the error dynamics are $\mathbf{e}(k+1) = \big[I - W(k+1)H(k+1)\big]F(k)\mathbf{e}(k)$. We can show the stability of the error dynamics using the Lyapunov function $V(k, \mathbf{e}(k)) = \mathbf{e}(k)^T P(k)^{-1}\mathbf{e}(k)$.

5.4.2 Separation principle

If the LG system is detectable, and if the control law $\mathbf{u}(k) = C(k)\mathbf{x}(k)$ stabilizes the closed-loop system *assuming perfect state information*, then the control law $\mathbf{u}(k) = C(k)\hat{\mathbf{x}}(k)$ stabilizes the Kalman-filtered closed-loop system.

To see this, define the augmented state

$$\begin{bmatrix}\mathbf{x}(k)\\ \mathbf{e}(k)\end{bmatrix} = \begin{bmatrix}I & 0\\ I & -I\end{bmatrix}\begin{bmatrix}\mathbf{x}(k)\\ \hat{\mathbf{x}}(k)\end{bmatrix}$$

then take $\mathbf{v}(k) \equiv \mathbf{0}$ for all $k \in \mathcal{T}_0$ and $\mathbf{w}(k) \equiv \mathbf{0}$ for all $k \in \mathcal{T}_1$ and show that the augmented dynamics are stable.

This is an important result, because it allows us to tackle the problems of estimation and controller design independently.

5.4.3 Matrix Riccati equation and steady-state solution

Assuming everything but $\bar{P}(k+1)$ is known, the covariance update equation

$$\begin{aligned}\bar{P}(k+1) = F(k)\Big[\bar{P}(k) - \bar{P}(k)H(k)^T\big(H(k)\bar{P}(k)H(k)^T + R(k)\big)^{-1}H(k)\bar{P}(k)\Big]F(k)^T\\ + \Gamma(k)Q(k)\Gamma(k)^T\end{aligned}$$

is a dynamic difference equation for the evolution of $\bar{P}(k+1)$ in terms of $\bar{P}(k)$. It's quadratic in $\bar{P}(k)$. Any such difference or differential equation that contains both constant and quadratic terms in the variable is called a **Riccati equation**, named after the Italian mathematician Jacopo Riccati (1676 - 1754).

In the special case of

- time-invariance, $F(k) \equiv F$, $\Gamma(k) \equiv \Gamma$, $H(k) \equiv H$, $Q(k) \equiv Q$, $R(k) \equiv R$ for all k
- (F, H) completely observable and (F, Γ) completely controllable

the estimator covariance $\bar{P}(k)$ tends to a steady-state value $\bar{P}_{\text{ss}}$ as $k \to \infty$. $\bar{P}_{\text{ss}}$ satisfies the discrete-time algebraic Riccati equation

$$\bar{P}_{\text{ss}} = F\big[\bar{P}_{\text{ss}} - \bar{P}_{\text{ss}}H^T\big(H\bar{P}_{\text{ss}}H^T + R\big)^{-1}H\bar{P}_{\text{ss}}\big]F^T + \Gamma Q\Gamma^T$$

which can be solved accurately and efficiently using Matlab's digital linear quadratic estimator function,

$$[W_{\text{ss}}, \bar{P}_{\text{ss}}, P_{\text{ss}}] = \text{dlqe}(F, \Gamma, H, Q, R)$$

For completely controllable and observable, time-invariant systems, the estimator and error dynamics are

$$\begin{aligned}\hat{\mathbf{x}}(k+1) &= \bar{\mathbf{x}}(k+1) + W_{\text{ss}}[\mathbf{z}(k+1) - H\bar{\mathbf{x}}(k+1)]\\ \mathbf{e}(k+1) &= (I - W_{\text{ss}}H)F\mathbf{e}(k)\end{aligned}$$

where $|\lambda_i| < 1$ for all eigenvalues λ_i of $(I - W_{\text{ss}}H)F$, i.e. the error dynamics are stable.

5.5 KF derivation 1: MMSE

This derivation uses the fundamental equations of linear estimation (1) to calculate the MMSE state estimator. The basic steps are:

- use expectation properties and LG statistical assumptions to calculate $\bar{\mathbf{x}}(k+1)$ and $\bar{P}(k+1)$
- by the same methods, write $\bar{\mathbf{z}}(k+1)$, $\bar{P}_{zz}(k+1)$, and $\bar{P}_{xz}(k+1)$ in terms of $\bar{\mathbf{x}}(k+1)$ and $\bar{P}(k+1)$
- using the fundamental equations of linear estimation, write

$$\hat{\mathbf{x}}(k+1) = \bar{\mathbf{x}}(k+1) + \bar{P}_{xz}(k+1)\bar{P}_{zz}(k+1)^{-1}\big[\mathbf{z}(k+1) - \bar{\mathbf{z}}(k+1)\big]$$
$$P(k+1) = \bar{P}(k+1) - \bar{P}_{xz}(k+1)\bar{P}_{zz}(k+1)^{-1}\bar{P}_{xz}(k+1)^T$$

- define intermediate matrices $W(k+1)$, $S(k+1)$, $\boldsymbol{\nu}(k+1)$ and simplify the results to reproduce the KF equations

5.6 KF derivation 2: MAP

Although the MMSE and MAP estimators are equivalent for LG systems, this is not the case for a nonlinear system or one with non-Gaussian randomness. MAP estimation is useful in such cases. This section sketches the steps in the MAP derivation.

- find the posterior distribution using Bayes' rule:

$$p(\mathbf{x}(k+1), \mathbf{v}(k)|\mathbf{z}(k+1)) = \frac{p(\mathbf{z}(k+1)|\mathbf{x}(k+1)), \mathbf{v}(k))p(\mathbf{x}(k+1), \mathbf{v}(k))}{p(\mathbf{z}(k+1))}$$

  - write down $p(\mathbf{z}(k+1)|\mathbf{x}(k+1), \mathbf{v}(k))$ (easy since $\mathbf{z}(k+1)$ is a linear function of Gaussian RVs $\mathbf{x}(k+1)$ and $\mathbf{w}(k+1)$)
  - ignore $p(\mathbf{z}(k+1))$; it will drop out when we maximize

- define the cost

$$J_a(\mathbf{x}(k), \mathbf{v}(k), \mathbf{x}(k+1), k) = -\ln p(\mathbf{x}(k+1), \mathbf{v}(k)|\mathbf{z}(k+1))$$

- solve the unconstrained convex optimization problem

$$\hat{\mathbf{x}}_{\text{MAP}}(k+1) = \arg\max p(\mathbf{x}(k+1), \mathbf{v}(k)|\mathbf{z}(k+1)) = \arg\min J_a(\mathbf{x}(k), \mathbf{v}(k), \mathbf{x}(k+1), k)$$

– use the inverse dynamics

$$\mathbf{x}(k) = F(k)^{-1}[\mathbf{x}(k+1) - G(k)\mathbf{u}(k) - \Gamma(k)\mathbf{v}(k)]$$

to eliminate $\mathbf{x}(k)$ from J_a. Define the cost function

$$J_b(\mathbf{v}(k), \mathbf{x}(k+1), k) = J_a\Big(F(k)^{-1}[\mathbf{x}(k+1) - G(k)\mathbf{u}(k) - \Gamma(k)\mathbf{v}(k)], \mathbf{v}(k), \mathbf{x}(k+1), k\Big)$$

– solve the optimization problem

$$\hat{\mathbf{v}}(k) = \arg\min_{\mathbf{v}(k)} J_b(\mathbf{v}(k), \mathbf{x}(k+1), k)$$

and plug $\hat{\mathbf{v}}(k)$ into J_b to define the cost function

$$J_c(\mathbf{x}(k+1), k+1) = J_b(\hat{\mathbf{v}}(k), \hat{\mathbf{x}}(k+1), k)$$

– solve the optimization problem

$$\hat{\mathbf{x}}_{\text{MAP}}(k+1) = \arg\min_{\mathbf{x}(k+1)} J_c(\mathbf{x}(k+1), k+1)$$

- simplify $\hat{\mathbf{x}}_{\text{MAP}}(k+1)$ to reproduce the KF equations

5.7 Tests of KF correctness

In static estimation, we defined an estimator as consistent if it converged to the true value in the limit of large k:

$$\lim_{k\to\infty} \mathbf{E}[\|\tilde{\mathbf{x}}(k)\|^2] = 0$$

For the Kalman filter, however, the estimation error covariance $P(k)$ settles into a steady state $P_{\text{ss}} \neq 0$ as $k \to \infty$, so consistency in this sense is unattainable.

There are two other metrics we'd like the KF to satisfy, however:

1. unbiasedness:

$$\mathbf{E}[\tilde{\mathbf{x}}(k)] = \mathbf{0} \text{ for all } k$$

2. predicted and measured MSE match:

$$\mathbf{E}[\tilde{\mathbf{x}}(k)\tilde{\mathbf{x}}(k)^T] = P(k) \text{ for all } k$$

5.7.1 Sources of error

If our state estimator is biased, or if the measured MSE fails to match the predicted estimation error covariance, then something is wrong. Some possible sources of error:

1. programming error (fix: adequate testing)
2. numerical error (fix: square root techniques for matrix inversion)
3. modeling error, such as
   - $F(k)$ or $H(k)$ may have parameter errors
   - the system model may be a reduced-order approximate the true system, with some states neglected in the interest of fast computation
   - the assumptions of linear dynamics and Gaussian random variables may not be correct
   - the disturbance covariance matrix $Q(k)$ may be wrong

The final point is a persistent difficulty. While $R(k)$, the measurement error covariance, can usually be found in sensor spec sheets, empirical data on $Q(k)$ are rarely available. $Q(k)$ is often viewed as a tunable parameter that encodes our *a priori* beliefs about the dynamic model: if the dynamic model is very accurate, then a small $Q(k)$ is probably justifiable. If the dynamics are highly simplified, on the other hand, or if the system is subject to lots of disturbances, then $Q(k)$ may be large. There's a whole literature on tuning $Q(k)$.

5.7.2 Likelihood of the filter

One method for comparing two KF models is to compute the likelihood (the probability density of the observations given the model parameters) of each model.

Let K_i be the set of filter parameters (system matrices, measurement matrices, and covariance matrices) in KF implementation i. Then the likelihood of the i^{th} filter is

$$p(\mathbf{z}^k|K_i) = \prod_{j=1}^{k} p(\boldsymbol{\nu}^i(j))$$

where $\boldsymbol{\nu}^i(j)$ is the j^{th} innovation in the i^{th} filter implementation. Conveniently, the information contained in the observations and in the innovations is equivalent. Since $\boldsymbol{\nu}^i(j) \sim \mathcal{N}(\mathbf{0}, S^i(j))$, where $S^i(j)$ is the j^{th} measurement prediction error covariance in the i^{th} filter, the likelihood can be written as

$$p(\mathbf{z}^k|K_i) = (2\pi)^{-k/2}\Big(\prod_{j=1}^{k}\sqrt{\det S^i(j)}\Big)\exp\Big\{-\frac{1}{2}\sum_{j=1}^{k}\boldsymbol{\nu}^i(j)^T(S^i(j))^{-1}\boldsymbol{\nu}^i(j)\Big\}$$

$$\Rightarrow \quad -\ln p(\mathbf{z}^k|K_i) = \frac{k}{2}\ln(2\pi) + \frac{1}{2}\sum_{j=1}^{k}\Big(\boldsymbol{\nu}^i(j)^T(S^i(j))^{-1}\boldsymbol{\nu}^i(j) - \ln\det S^i(j)\Big)$$

The filter likelihood gives a coarse measure of the filter's performance. All else being equal, a filter with a higher likelihood (or, equivalently, a smaller negative log-likelihood) is typically better. Problems associated with over-fitting can arise, however, so it's best to use a formal hypothesis test to evaluate the filter's performance. Five common tests are presented next.

5.7.3 Simulation: normalized estimation error squared

If the true state $\mathbf{x}(k)$ is known, then we can compute the estimation error $\tilde{\mathbf{x}}(k) = \mathbf{x}(k) - \hat{\mathbf{x}}(k)$ and conduct statistical tests. This is rarely the case in experiments, unless we have a second, extremely accurate sensor array for use in the lab. In Monte Carlo simulation, however, the true state is accessible.

Consider the **normalized estimation error squared** (NEES) of the state,

$$\epsilon(k) = \tilde{\mathbf{x}}(k)^T P(k)^{-1} \tilde{\mathbf{x}}(k)$$

For an unbiased estimator, $\tilde{\mathbf{x}}(k) \sim \mathcal{N}(\mathbf{0}, P(k))$, which implies that $\epsilon(k) \sim \chi^2_{n_x}$.

Suppose we generate N histories of $\mathbf{v}(k)$ and $\mathbf{w}(k)$ (and by extension, $\mathbf{x}(k)$ and $\mathbf{z}(k)$), and run our KF on each history, computing N different histories of $\hat{\mathbf{x}}(k)$ and $\epsilon(k)$. Call the i^{th} $\epsilon(k)$ history $\epsilon^i(k)$, and define the N-run average NEES

$$\bar{\epsilon}(k) = \frac{1}{N} \sum_{i=1}^{N} \epsilon^i(k)$$

then $N\bar{\epsilon}(k) \sim \chi^2_{Nn_x}$, and we can test the hypothesis

$$H_0 : \text{the state estimation errors are consistent with the KF covariances}$$

The optimal Neyman-Pearson test of size α is

$$\text{reject } H_0 \text{ if } \bar{\epsilon}(k) \notin [r_1, r_2]$$

where

$$\mathbf{P}\big(\bar{\epsilon}(k) \notin [r_1, r_2] \mid H_0\big) = \alpha$$

The critical region is given by

$$r_1 = \frac{1}{N} F^{-1}\Big(\frac{\alpha}{2}\Big), \quad r_2 = \frac{1}{N} F^{-1}\Big(1 - \frac{\alpha}{2}\Big)$$

where $F^{-1}(\cdot)$ is the inverse $\chi^2_{Nn_x}$ cdf, chi2inv($\cdot$, Nn_x) in Matlab.

NB. "Truth-model simulations are doomed to succeed." Why? Because the "true" state and observations are generated from the same dynamic, measurement, and statistical models that the KF uses. If these models don't accurately reflect reality, then the KF may pass truth-model tests but perform terribly in real experiments.

5.7.4 Multiple experiments: normalized innovation squared

Unlike truth-model simulations, experiments don't give access to the true state $\mathbf{x}(k)$. The innovations $\boldsymbol{\nu}(k)$ and their covariance matrices $S(k)$, however, are always available.

Define the **normalized innovation squared** (NIS),

$$\epsilon_{\boldsymbol{\nu}}(k) = \boldsymbol{\nu}(k)^T S(k)^{-1} \boldsymbol{\nu}(k)$$

Since $\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, S(k))$, the NIS is chi-square distributed: $\epsilon_{\boldsymbol{\nu}}(k) \sim \chi^2_{n_z}$.

Suppose we do N experiments and record the NIS histories $\epsilon^i_{\boldsymbol{\nu}}(k)$, $i \in \{1, \dots, N\}$. Define the N-run average NIS

$$\bar{\epsilon}_{\boldsymbol{\nu}}(k) = \frac{1}{N} \sum_{i=1}^{N} \epsilon^i_{\boldsymbol{\nu}}(k)$$

Then $N\bar{\epsilon}_{\boldsymbol{\nu}}(k) \sim \chi^2_{Nn_z}$, and we can test the hypothesis

$$H_0 : \text{the innovations are consistent with the KF covariances}$$

The optimal Neyman-Pearson test of size α is

$$\text{reject } H_0 \text{ if } \bar{\epsilon}_{\boldsymbol{\nu}}(k) \notin [r_1, r_2]$$

where

$$\mathbf{P}\big(\bar{\epsilon}_{\boldsymbol{\nu}}(k) \notin [r_1, r_2] \mid H_0\big) = \alpha$$

The critical region is given by

$$r_1 = \frac{1}{N} F^{-1}\Big(\frac{\alpha}{2}\Big), \quad r_2 = \frac{1}{N} F^{-1}\Big(1 - \frac{\alpha}{2}\Big)$$

where $F^{-1}(\cdot)$ is the inverse $\chi^2_{Nn_z}$ cdf, chi2inv($\cdot$, Nn_z) in Matlab.

5.7.5 Multiple experiments: whiteness of the innovations

It can be shown that for all j, k

$$\mathbf{E}[\boldsymbol{\nu}(k)] = \mathbf{0}$$
$$\mathbf{E}[\boldsymbol{\nu}(k)\boldsymbol{\nu}(j)^T] = \delta_{jk} S(k)$$

i.e. the innovations are zero mean and white. This gives us another correctness test that can be conducted with either simulations or experiments.

Suppose we do N experiments and record the innovations $\boldsymbol{\nu}^i(k)$ for each experiment $i \in \{1, \dots, N\}$. Define the sample statistic

$$\bar{\rho}_{lm}(j,k) = \frac{\sum_{i=1}^{N} \nu^i_l(j) \nu^i_m(k)}{\sqrt{\big(\sum_{i=1}^{N} \nu^i_l(j)^2\big)\big(\sum_{i=1}^{N} \nu^i_m(k)^2\big)}}$$

where $\nu_l^i(j)$ is the l^{th} element of the j^{th} innovation in the i^{th} experiment. Thus, $\bar{\rho}_{lm}(j,k)$ is the sample correlation coefficient between the l^{th} and m^{th} elements of the innovation at times j and k.

(The above is general but confusing. A common simplification is to look only at the correlation between a particular element of the innovation at one time and the next, i.e. to take $l = m$ and $k = j + 1$.)

If the innovation sequence is white then for $j \neq k$, $\bar{\rho}_{lm}(j,k)$ should obey

$$\mathbf{E}[\bar{\rho}_{lm}(j,k)] = 0$$
$$\mathbf{E}[\bar{\rho}_{lm}(j,k)^2] = \frac{1}{N}$$

so $\mathbf{var}(\bar{\rho}_{lm}(j,k)) = 1/N$. The distribution of $\bar{\rho}_{lm}(j,k)$ is non-trivial to compute, but in light of the CLT it's plausible to treat it as Gaussian. Under this approximation, we can test the hypothesis

$$H_0 : \text{the innovation sequence is white}$$

with optimal Neyman-Pearson test of size α,

$$\text{reject } H_0 \text{ if } \bar{\rho}_{lm}(j,k) \notin [-r, r]$$

where

$$\mathbf{P}\big(\bar{\rho}_{lm}(j,k) \notin [-r,r] \mid H_0\big) = \alpha$$

The critical region is given by

$$r = F^{-1}\Big(1 - \frac{\alpha}{2}\Big)$$

where $F^{-1}(\cdot)$ is the inverse normal cdf with mean 0 and variance $1/N$, norminv($\cdot$, 0, $1/\sqrt{N}$) in Matlab.

NB. If the system is TI and has settled into a steady state, we can visually test the whiteness of the innovations by looking at the power spectral density of the innovation sequence using the fast Fourier transform, fft.m in Matlab.

5.7.6 Single experiment: normalized innovation squared

Sometimes we'd like to do an experimental test, but only have the resources to run a single experiment. In these situations, the innovation tests are inaccessible because we don't have N experiments to average over. The best we can do is use an ergodicity argument to exchange averages over *experiments* for averages over *time*. This ergodicity argument is not justified for a general system, since the innovation sequence need not be stationary, but in practice the method often works.

Suppose we do a single experiment, letting our filter run for $k \in \{0, 1, \ldots, K\}$. The time-averaged NIS is

$$\bar{\epsilon}_{\boldsymbol{\nu}} = \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{\nu}(k)^T S(k)^{-1} \boldsymbol{\nu}(k)$$

Then $K\bar{\epsilon}_{\boldsymbol{\nu}} \sim \chi^2_{Kn_z}$, and we can test the hypothesis

$$H_0 : \text{the innovations are consistent with the KF covariances}$$

The optimal Neyman-Pearson test of size α is

$$\text{reject } H_0 \text{ if } \bar{\epsilon}_{\boldsymbol{\nu}} \notin [r_1, r_2]$$

where

$$\mathbf{P}\big(\bar{\epsilon}_{\boldsymbol{\nu}} \notin [r_1, r_2] \mid H_0\big) = \alpha$$

The critical region is given by

$$r_1 = \frac{1}{K} F^{-1}\Big(\frac{\alpha}{2}\Big), \quad r_2 = \frac{1}{K} F^{-1}\Big(1 - \frac{\alpha}{2}\Big)$$

where $F^{-1}(\cdot)$ is the inverse $\chi^2_{Kn_z}$ cdf, chi2inv($\cdot$, Kn_z) in Matlab.

5.7.7 Single experiment: whiteness of the innovations

We can also exchange averages over experiments with averages over time for the innovation whiteness test.

Suppose we do a single experiment, letting our filter run for $k \in \{0, 1, \ldots, K\}$. The time-averaged correlation coefficient between the l^{th} and m^{th} elements of the innovation at times k and $k + j$ is

$$\bar{\rho}_{lm}(j) = \frac{\sum_{k=1}^{K} \nu_l(k)\nu_m(k+j)}{\sqrt{\big(\sum_{k=1}^{K} \nu_l(k)^2\big)\big(\sum_{k=1}^{K} \nu_m(k+j)^2\big)}}$$

If K is large, then in light of the CLT it's plausible to approximate $\bar{\rho}_{lm}(j)$ as Gaussian. If the innovations are white, then the sample mean of $\bar{\rho}_{lm}(j)$ should be zero for all $j \neq 0$, and its sample variance should be $1/K$. We can therefore test the hypothesis

$$H_0 : \text{the innovations are white}$$

with optimal Neyman-Pearson test of size α,

$$\text{reject } H_0 \text{ if } \bar{\rho}_{lm}(j) \notin [-r, r]$$

where

$$\mathbf{P}\big(\bar{\rho}_{lm}(j) \notin [-r, r] \mid H_0\big) = \alpha$$

The critical region is given by

$$r = F^{-1}\Big(1 - \frac{\alpha}{2}\Big)$$

where $F^{-1}(\cdot)$ is the inverse normal cdf with mean 0 and variance $1/K$, norminv($\cdot$, 0, $1/\sqrt{K}$) in Matlab.

5.8 KF initialization

The KF takes as inputs the initial estimate $\hat{\mathbf{x}}(0)$ and estimation error covariance $P(0)$. In homework problems these are given, but in a real estimation application we need some practical way to come up with them. This is particularly important in nonlinear filtering, where the estimation error may diverge for a bad initial guess.

A few initialization scenarios:

1. All elements of $\hat{\mathbf{x}}(0)$ may be measured by sensors with known error covariances, giving $P(0)$.
   - this is wonderful but very rare in practice
2. You may have some *a priori* knowledge of the system's initial state.
   - more common, but hardly ubiquitous
3. You can solve a batch estimation problem using the first few measurements.
   - similar to calculating an initial guess for a nonlinear least squares problem

This section explores scenario 3, which is a bit laborious but often necessary for good filter performance. Note that information filtering, a modification of Kalman filtering, is a simpler alternative to the following.

Consider the first k measurements,

$$\begin{aligned}
\mathbf{z}(1) &= H(1)\mathbf{x}(1) + \mathbf{w}(1) \\
&= H(1)\Big[F(0)\mathbf{x}(0) + G(0)\mathbf{u}(0) + \Gamma(0)\mathbf{v}(0)\Big] + \mathbf{w}(1) \\
\mathbf{z}(2) &= H(2)\mathbf{x}(2) + \mathbf{w}(2) \\
&= H(2)\Big[F(1)\big(F(0)\mathbf{x}(0) + G(0)\mathbf{u}(0) + \Gamma(0)\mathbf{v}(0)\big) + G(1)\mathbf{u}(1) + \Gamma(1)\mathbf{v}(1)\Big] + \mathbf{w}(2) \\
&\vdots \\
\mathbf{z}(k) &= H(k)\mathbf{x}(k) + \mathbf{w}(k) \\
&= H(k)\Big[\text{(a big ugly expression)}\Big] + \mathbf{w}(k)
\end{aligned}$$

By a process similar to that in section 2.2.1, the lifted measurement vector $\mathbf{z}^k$ can be written in terms of $\mathbf{x}(0)$, the known controls $\mathbf{u}^{k-1}$, and the state and measurement disturbances:

$$\mathbf{z}^k = H^k\mathbf{x}(0) + G^{k-1}\mathbf{u}^{k-1} + C^{k-1}\mathbf{v}^{k-1} + \mathbf{w}^k$$

where

$$\mathbf{z}^k = \begin{bmatrix}\mathbf{z}(1)\\ \vdots \\ \mathbf{z}(k)\end{bmatrix} \in \mathbf{R}^{kn_z}, \quad H^k = \begin{bmatrix} H(1)F(0) \\ \vdots \\ H(k)F(k-1)\cdots F(0)\end{bmatrix} \in \mathbf{R}^{kn_z \times n_x}$$

$$\mathbf{u}^{k-1} = \begin{bmatrix}\mathbf{u}(0)\\ \vdots \\ \mathbf{u}(k-1)\end{bmatrix} \in \mathbf{R}^{kn_u}, \quad \mathbf{v}^{k-1} = \begin{bmatrix}\mathbf{v}(0)\\ \vdots \\ \mathbf{v}(k-1)\end{bmatrix} \in \mathbf{R}^{kn_v}, \quad \mathbf{w}^{k-1} = \begin{bmatrix}\mathbf{w}(1)\\ \vdots \\ \mathbf{w}(k)\end{bmatrix} \in \mathbf{R}^{kn_z}$$

$$G^{k-1} = \begin{bmatrix} H(1)G(0) & & & \\ H(2)F(1)G(0) & H(2)G(1) & & \\ \vdots & & \ddots & \\ H(k)F(k-1)\cdots F(1)G(0) & H(k)F(k-1)\cdots F(2)G(1) & \dots & H(k)G(k-1)\end{bmatrix}$$

$$C^{k-1} = \begin{bmatrix} H(1)\Gamma(0) & & & \\ H(2)F(1)\Gamma(0) & H(2)\Gamma(1) & & \\ \vdots & & \ddots & \\ H(k)F(k-1)\cdots F(1)\Gamma(0) & H(k)F(k-1)\cdots F(2)\Gamma(1) & \dots & H(k)\Gamma(k-1)\end{bmatrix}$$

so $G^{k-1} \in \mathbf{R}^{kn_z \times kn_u}$ and $C^{k-1} \in \mathbf{R}^{kn_z \times kn_v}$.

The lifted covariance matrices are

$$R^k = \begin{bmatrix} R(1) & & \\ & \ddots & \\ & & R(k)\end{bmatrix} \in \mathbf{R}^{kn_z \times kn_z}, \quad Q^{k-1} = \begin{bmatrix} Q(0) & & \\ & \ddots & \\ & & Q(k-1)\end{bmatrix} \in \mathbf{R}^{kn_v \times kn_v}$$

Given these definitions, we can define the least-squares cost function

$$\begin{aligned} C_{\text{LS}}(\mathbf{x}(0), \mathbf{v}^{k-1}) = \frac{1}{2}&\left(\mathbf{z}^k - H^k\mathbf{x}(0) - G^{k-1}\mathbf{u}^{k-1} - C^{k-1}\mathbf{v}^{k-1}\right)^T (R^k)^{-1} \\ &* \left(\mathbf{z}^k - H^k\mathbf{x}(0) - G^{k-1}\mathbf{u}^{k-1} - C^{k-1}\mathbf{v}^{k-1}\right) \\ &+ \frac{1}{2}(\mathbf{v}^{k-1})^T (Q^{k-1})^{-1}\mathbf{v}^{k-1} \end{aligned}$$

and solve the first-order conditions

$$\left(\frac{\partial}{\partial \mathbf{x}(0)} C_{\text{LS}}(\mathbf{x}(0), \mathbf{v}^{k-1})\right)^T = \mathbf{0}, \quad \left(\frac{\partial}{\partial \mathbf{v}^{k-1}} C_{\text{LS}}(\mathbf{x}(0), \mathbf{v}^{k-1})\right)^T = \mathbf{0}$$

for the smoothed initial estimate $\hat{\mathbf{x}}(0|k)$ and the estimated process noise $\hat{\mathbf{v}}^{k-1|k}$.

The smoothed initial error covariance $P(0|k)$ can be produced from

$$\begin{bmatrix} P_{\mathbf{x}(0)\mathbf{x}(0)} & P_{\mathbf{x}(0)\mathbf{v}^{k-1}} \\ P^T_{\mathbf{x}(0)\mathbf{v}^{k-1}} & P_{\mathbf{v}^{k-1}\mathbf{v}^{k-1}} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 C_{\text{LS}}}{\partial \mathbf{x}(0)^2} & \frac{\partial^2 C_{\text{LS}}}{\partial \mathbf{x}(0)\partial \mathbf{v}^{k-1}} \\ \big(\frac{\partial^2 C_{\text{LS}}}{\partial \mathbf{x}(0)\partial \mathbf{v}^{k-1}}\big)^T & \frac{\partial^2 C_{\text{LS}}}{\partial (\mathbf{v}^{k-1})^2} \end{bmatrix}^{-1}$$

(The above comes from the fact that $C_{\text{LS}}(\mathbf{x}(0), \mathbf{v}^{k-1})$ is essentially the negative log-likelihood of a Gaussian distribution.)

It may be tempting at this point to initialize the KF at $k = 0$ with $\hat{\mathbf{x}}(0|k)$ and $P(0|k) = P_{\mathbf{x}(0)\mathbf{x}(0)}$, but doing so would mean double counting the first k samples. This causes a bias in the future estimates – a Bad Thing.

The right way to use the results of the batch estimation is to propagate $\hat{\mathbf{x}}(0|k)$ and $P(0|k)$ forward through the dynamics equation, and start the KF with at stage k. The state propagation equation is

$$\begin{aligned} \hat{\mathbf{x}}(k) &= F(k-1)\cdots F(1)\hat{\mathbf{x}}(0|k) + \sum_{j=0}^{k-1} F(k-1)\cdots F(j+1)\big[G(j)\mathbf{u}(j) + \Gamma(j)\mathbf{v}(j)\big] \\ &= D\hat{\mathbf{x}}(0|k) + E\mathbf{v}^{k-1|k} + \mathbf{g} \\ &= \begin{bmatrix} D & E \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}(0|k) \\ \mathbf{v}^{k-1|k} \end{bmatrix} \end{aligned}$$

where

$$\begin{aligned} D &= F(k-1)\cdots F(1) \in \mathbf{R}^{n_x \times n_x} \\ E &= \begin{bmatrix} F(k-1)\cdots F(1)\Gamma(0), & F(k-1)\cdots F(2)\Gamma(1), & \ldots, & \Gamma(k-1) \end{bmatrix} \in \mathbf{R}^{n_x \times k n_v} \\ \mathbf{g} &= \sum_{j=0}^{k-1} F(k-1)\cdots F(j+1)G(j)\mathbf{u}(j) \in \mathbf{R}^{n_x} \end{aligned}$$

By the transformation rule $\mathbf{cov}(A\mathbf{x} + b) = A\mathbf{cov}(\mathbf{x})A^T$, the estimation error covariance propagation is

$$\begin{aligned} P(k) &= \begin{bmatrix} D & E \end{bmatrix} \begin{bmatrix} P_{\mathbf{x}(0)\mathbf{x}(0)} & P_{\mathbf{x}(0)\mathbf{v}^{k-1}} \\ P^T_{\mathbf{x}(0)\mathbf{v}^{k-1}} & P_{\mathbf{v}^{k-1}\mathbf{v}^{k-1}} \end{bmatrix} \begin{bmatrix} D^T \\ E^T \end{bmatrix} \\ &= \begin{bmatrix} D & E \end{bmatrix} \begin{bmatrix} \frac{\partial^2 C_{\text{LS}}}{\partial \mathbf{x}(0)^2} & \frac{\partial^2 C_{\text{LS}}}{\partial \mathbf{x}(0)\partial \mathbf{v}^{k-1}} \\ \big(\frac{\partial^2 C_{\text{LS}}}{\partial \mathbf{x}(0)\partial \mathbf{v}^{k-1}}\big)^T & \frac{\partial^2 C_{\text{LS}}}{\partial (\mathbf{v}^{k-1})^2} \end{bmatrix}^{-1} \begin{bmatrix} D^T \\ E^T \end{bmatrix} \end{aligned}$$

Summarizing, the initialization steps are

- solve a batch estimation problem with the first k samples for $\hat{\mathbf{x}}(0|k)$ and $\hat{\mathbf{v}}^{k-1|k}$
- compute the estimation error covariance by inverting the Hessian of $C_{\text{LS}}(\hat{\mathbf{x}}(0|k), \hat{\mathbf{v}}^{k-1|k})$
- propagate $\hat{\mathbf{x}}(0|k)$ through the dynamics equation k times to calculate $\hat{\mathbf{x}}(k)$
- use the covariance transformation formula to calculate $P(k)$
- initialize the KF at sample k with $\hat{\mathbf{x}}(k)$ and $P(k)$

5.9 Model mismatch

So far we've assumed that our dynamic and measurement models exactly reflect the underlying physics of the system we're studying. This is never the case in practice – our models are always inaccurate to some degree. This section explores the robustness of the KF to modeling errors.

Suppose we implement a KF based on the model

$$\begin{aligned}\mathbf{x}(k+1) &= F_f(k)\mathbf{x}(k) + G_f(k)\mathbf{u}(k) + \Gamma_f(k)\mathbf{v}_f(k)\\ \mathbf{z}(k) &= H_f(k)\mathbf{x}(k) + \mathbf{w}_f(k)\end{aligned}$$

with the typical statistical assumptions on $\mathbf{v}_f(k)$ and $\mathbf{w}_f(k)$:

$$\begin{aligned}\mathbf{E}[\mathbf{v}_f(k)] &= \mathbf{0}\\ \mathbf{E}[\mathbf{v}_f(k)\mathbf{v}_f(j)^T] &= \delta_{kj}Q_f(k)\\ \mathbf{E}[\mathbf{w}_f(k)] &= \mathbf{0}\\ \mathbf{E}[\mathbf{w}_f(k)\mathbf{w}_f(j)^T] &= \delta_{kj}R_f(k)\end{aligned}$$

along with the lack of correlation between the disturbance, noise and initial state.

What happens if we implement this filter on a LG system that's actually governed by $F(k), G(k), \Gamma(k), H(k), Q(k)$ and $R(k)$?

Let $\mathbf{e}^-(k)$ and $\mathbf{e}^+(k)$ be the *a priori* and *a posteriori* errors, respectively:

$$\begin{aligned}\mathbf{e}^-(k) &= \mathbf{x}(k) - \bar{\mathbf{x}}(k)\\ \mathbf{e}^+(k) &= \mathbf{x}(k) - \hat{\mathbf{x}}(k)\end{aligned}$$

It can be shown that for the mismatched KF, these errors evolve according to

$$\begin{aligned}\mathbf{e}^-(k+1) &= F_f(k)\mathbf{e}^+(k) + [F(k) - F_f(k)]\mathbf{x}(k) + [G(k) - G_f(k)]\mathbf{u}(k) + \Gamma(k)\mathbf{v}(k)\\ \mathbf{e}^+(k+1) &= [I - W_f(k+1)H_f(k+1)]\mathbf{e}^-(k+1)\\ &\quad + W_f(k+1)[H_f(k+1) - H(k+1)]\mathbf{x}(k+1) + W_f(k+1)\mathbf{w}(k+1)\end{aligned}$$

The *a posteriori* error equation can be written as

$$\mathbf{e}^+(k+1) = [I - W_f(k+1)H_f(k+1)]F_f(k)\mathbf{e}^+(k) + \mathbf{f}(\mathbf{v}(k), \mathbf{w}(k+1)) + \mathbf{g}(\mathbf{x}(k) + \mathbf{u}(k))$$

where $\mathbf{f}(\mathbf{v}(k), \mathbf{w}(k+1))$ includes the usual disturbance and noise terms, but $\mathbf{g}(\mathbf{x}(k), \mathbf{u}(k))$ consists of non-homogenous forcing terms. These terms can drive $\mathbf{e}^+(k)$ into instability.

It's complicated to fully analyze $\lim_{k\to\infty} \mathbf{e}^+(k)$, because the time histories of $\mathbf{x}(k)$ and $\mathbf{u}(k)$ are needed as inputs (see Bar-Shalom for details). The punchline is this:

$$P^{\text{true}}(k+1) \succeq P_f(k+1)$$

i.e. the state estimation MSE given by the mismatched filter is upper bounded by the true state estimation error covariance (meaning you think your estimates are more accurate than they really are).

If $P^{\text{true}}(k+1)$ is acceptably small, then the model mismatch may still give good results despite the systematic modeling error. Sometimes, however, the results are unacceptable and you need to derive a better physical model, for instance by

- improving the model calibration (e.g. tuning $Q(k)$)
- including some previously neglected states

NB. The model mismatch analysis assumes that the true physics ($F(k), G(k), \Gamma(k), H(k)$) and the true statistics ($Q(k)$ and $R(k)$) are known. In reality, however, we don't know them exactly – if we did, we'd use them in the KF! Still, analyses of this kind are useful in testing, because we can test a candidate filter against a variety of potentially "true" physics and statistics. A good filter will be robust to a broad spectrum of mismatch scenarios.

6 The square-root information filter

The Kalman filter algorithm in section 5.3 can run into numerical issues. Each cycle involves squaring $F(k), \Gamma(k), H(k+1)$, and $W(k+1)$. If any of these matrices are ill-conditioned, then iteratively squaring them can quickly lead to roundoff errors. The innovation calculation and the state estimation error covariance update equation also involve subtractions. If the quantities being differenced are of similar magnitude, then iteratively subtracting them can quickly accumulate roundoff errors. A solution to the numerical instability problem is to use square-root techniques (QR and Cholesky factorization).

Another difficulty with implementing the KF in practice is initialization. The batch initialization procedure outlined in section 5.8 can be time-consuming, and may not give particularly good results. Ideally, we'd like to be able to initialize the KF with the diffuse prior – that is, with an infinite state estimation error covariance matrix $P(0)$ – but this is not achievable in practice. A solution to the initialization problem is to use an information filter, which reformulates the KF algorithm in terms of inverse covariance matrices.

This section develops the **square-root information filter** (SRIF), which combines square-root computational techniques with the information filter formulation. We begin with the (non-square-root) information filter, then add in square-root techniques.

6.1 The IF algorithm

Consider again the LG system

$$\begin{aligned} \mathbf{x}(k+1) &= F(k)\mathbf{x}(k) + G(k)\mathbf{u}(k) + \Gamma(k)\mathbf{v}(k) \\ \mathbf{z}(k) &= H(k)\mathbf{x}(k) + \mathbf{w}(k) \end{aligned}$$

with the usual statistical assumptions on $\mathbf{v}(k)$, $\mathbf{w}(k)$ and $\mathbf{x}(0)$.

Instead of processing $\hat{\mathbf{x}}(k), P(k), \bar{\mathbf{x}}(k+1), \bar{P}(k+1)$, in the information filter we process the following:

$$\begin{aligned} \hat{\mathbf{y}}(k) &= P(k)^{-1}\hat{\mathbf{x}}(k) \\ \bar{\mathbf{y}}(k) &= \bar{P}(k)^{-1}\bar{\mathbf{x}}(k) \\ \mathcal{I}(k) &= P(k)^{-1} \\ \bar{\mathcal{I}}(k) &= \bar{P}(k)^{-1} \end{aligned}$$

where the inverse covariance matrices $\mathcal{I}(k)$ and $\bar{\mathcal{I}}(k)$ are called **information matrices**. The intuition is that there's an inverse relationship between the "size" of a covariance matrix, and the amount of information contained in the corresponding RV. The nomenclature is meant to evoke the Fisher information matrix.

By following the steps of the MMSE KF derivation and making use of the matrix inversion lemma, we can derive dynamic propagation and measurement update formulas in terms of the information matrices and transformed vectors. Given $\hat{\mathbf{y}}(0)$ and $\mathcal{I}(0)$, the KF algorithm is:

1. set $k = 0$
2. **dynamic propagation**
   (a) for convenience, define the intermediate matrix
   $$A(k) = F(k)^{-T}\mathcal{I}(k)F(k)^{-1}$$
   (b) predict the next (transformed) state
   $$\bar{\mathbf{y}}(k{+}1) = \left\{I - A(k)\Gamma(k)\left[\Gamma(k)^T A(k)\Gamma(k) + Q(k)^{-1}\right]^{-1}\Gamma(k)^T\right\}\left[F(k)^{-T}\hat{\mathbf{y}}(k) + A(k)G(k)\mathbf{u}(k)\right]$$
   (c) calculate the corresponding information matrix
   $$\bar{\mathcal{I}}(k+1) = A(k) - A(k)\Gamma(k)\left[\Gamma(k)^T A(k)\Gamma(k) + Q(k)^{-1}\right]^{-1}\Gamma(k)^T A(k)$$
3. **measurement update**
   (a) update the (transformed) state estimate
   $$\hat{\mathbf{y}}(k+1) = \bar{\mathbf{y}}(k+1) + H(k+1)^T R(k+1)^{-1}\mathbf{z}(k+1)$$
   (b) calculate the corresponding information matrix
   $$\mathcal{I}(k+1) = \bar{\mathcal{I}}(k+1) + H(k+1)^T R(k+1)^{-1}H(k+1)$$
4. **inverse transformation**
   (a) recover the state estimate
   $$\hat{\mathbf{x}}(k+1) = \mathcal{I}(k+1)^{-1}\hat{\mathbf{y}}(k+1)$$
   (b) recover the state estimation error covariance
   $$P(k+1) = \mathcal{I}(k+1)^{-1}$$
5. increment k and return to step 2

6.1.1 Remarks

1. If all of the following hold:
   - $n_x < n_z$ (more measurements than states)
   - $n_v < n_z$ (more measurements than process noise elements)
   - $R(k+1)$ is diagonal (no correlation between elements of $\mathbf{w}(k+1)$)

then the IF is computationally faster than the KF.

2. If we have no prior information about $\mathbf{x}(0)$, then in the IF algorithm we only need to set $\mathcal{I}(0) = 0$ (which implies $\hat{\mathbf{y}}(0) = \mathbf{0}$). This is a major advantage of the IF: it avoids the batch initialization problem in the KF. The downside is that $\mathcal{I}(k)$ won't be invertible for the first few stages, so we won't be able to recover $P(k)$ or $\hat{\mathbf{x}}(k)$ for a little while.

   NB. This is less of an advantage in nonlinear estimation, because we need an initial state estimate to linearize the dynamics and measurement model about.

3. **KF $\leftrightarrow$ IF duality.**

   In the dynamic propagation step of the KF algorithm, we typically find that $P(k) \to \bar{P}(k+1) \succ P(k)$, because the state disturbance tends to diffuse our knowledge of $\mathbf{x}(k+1)$. Similarly, in the measurement update step of the IF algorithm, $\bar{\mathcal{I}}(k+1) \to \mathcal{I}(k+1) \succ \bar{\mathcal{I}}(k+1)$, because we gain information when we measure $\mathbf{z}(k+1)$.

   Similarly, in the measurement update step of the KF algorithm, $\bar{P}(k+1) \to P(k+1) \prec \bar{P}(k+1)$, because we gain information when we measure $\mathbf{z}(k+1)$. In the dynamic propagation step of the IF algorithm, we typically find that $\mathcal{I}(k) \to \bar{\mathcal{I}}(k+1) \prec \mathcal{I}(k)$ because we lose information due to the state disturbance.

   The big idea: the dynamic propagation step of the KF is informationally equivalent to the measurement update step of the IF, and vice versa.

6.2 SRIF background

The IF algorithm avoids the initialization problem, but it still involves squaring matrices and subtraction. The SRIF solves these numerical problems by using QR and Cholesky factorizations. The two guiding principles of the SRIF:

1. only admit RVs with zero mean and identity covariance
2. only manipulate matrices through orthonormal transformations[2]

6.2.1 SRIEs

In the SRIF, we require that $\boldsymbol{\mathfrak{z}}_x(k), \mathbf{w}_x(k)$ and $\mathbf{w}_v(k)$ satisfy the **square-root information equations** (SRIEs)

$$\begin{aligned} \mathbf{0} &= \mathcal{R}_{vv}(k)\mathbf{v}(k) + \mathbf{w}_v(k) \\ \boldsymbol{\mathfrak{z}}_x(k) &= \mathcal{R}_{xx}(k)\mathbf{x}(k) + \mathbf{w}_x(k) \end{aligned}$$

[2]With one exception: we allow inversion of the dynamics matrices $F(k)$.

where $\mathbf{w}_x(k)$ and $\mathbf{w}_v(k)$ are zero mean with identity covariances,

$$\mathbf{E}\left[\begin{bmatrix}\mathbf{w}_x(k)\\ \mathbf{w}_v(k)\end{bmatrix}\right] = \mathbf{0}$$
$$\mathbf{E}\left[\begin{bmatrix}\mathbf{w}_x(k)\\ \mathbf{w}_v(k)\end{bmatrix}\begin{bmatrix}\mathbf{w}_x(j)^T & \mathbf{w}_v(j)^T\end{bmatrix}\right] = \delta_{kj}\begin{bmatrix}I & 0\\ 0 & I\end{bmatrix}$$

and where $\mathcal{R}_{xx}(k) \in \mathbf{R}^{n_x \times n_x}$ and $\bar{\mathcal{R}}_{xx}(k) \in \mathbf{R}^{n_x \times n_x}$ are some as yet unknown matrices, assumed to be invertible.

From the second SRIE, $\mathbf{x}(k) = \mathcal{R}_{xx}(k)^{-1}(\mathfrak{z}_x(k) - \mathbf{w}_x(k))$, so the state estimate is

$$\begin{aligned}\hat{\mathbf{x}}(k) &= \mathbf{E}[\mathbf{x}(k)|k] = \mathbf{E}[\mathcal{R}_{xx}(k)^{-1}(\mathfrak{z}_x(k) - \mathbf{w}_x(k))|k]\\ &= \mathcal{R}_{xx}(k)^{-1}\mathfrak{z}_x(k)\end{aligned}$$

and the estimation error covariance is

$$\begin{aligned}P(k) &= \mathbf{E}[(\mathbf{x}(k) - \hat{\mathbf{x}}(k))(\mathbf{x}(k) - \hat{\mathbf{x}}(k))^T|k] = \mathcal{R}_{xx}(k)^{-1}\,\mathbf{E}[\mathbf{w}_x(k)\mathbf{w}_x(k)^T|k]\mathcal{R}_{xx}(k)^{-T}\\ &= \mathcal{R}_{xx}(k)^{-1}\mathcal{R}_{xx}(k)^{-T}\end{aligned}$$

which gives the relationship

$$\mathcal{R}_{xx}(k)^T\mathcal{R}_{xx}(k) = P(k)^{-1} = \mathcal{I}(k)$$

so $\mathcal{R}_{xx}(k)$ must be a Cholesky factor of the information matrix $\mathcal{I}(k)$.

Similarly, the first SRIE implies that $\mathbf{v}(k) = -\mathcal{R}_{vv}^{-1}\mathbf{w}_v(k)$, so

$$\begin{aligned}\mathbf{E}[\mathbf{v}(k)|k] &= \mathbf{0}\\ Q(k) = \mathbf{E}[\mathbf{v}(k)\mathbf{v}(k)^T|k] &= \mathcal{R}_{vv}(k)^{-1}\,\mathbf{E}[\mathbf{w}_v(k)\mathbf{w}_v(k)^T|k]\mathcal{R}_{vv}(k)^{-T}\\ &= \mathcal{R}_{vv}(k)^{-1}\mathcal{R}_{vv}(k)^{-T}\end{aligned}$$

which gives the relationship

$$\mathcal{R}_{vv}(k)^T\mathcal{R}_{vv}(k) = Q(k)^{-1}$$

so $\mathcal{R}_{vv}(k)$ must be a Cholesky factor of the inverse of the disturbance covariance matrix $Q(k)$. Note that $\mathcal{R}_{vv}(k)$ is best calculated using $\text{inv}(\text{chol}(Q(k)))^T$ in Matlab.

6.2.2 Dynamic propagation

How to conduct the dynamic propagation step such that the SRIEs are satisfied?

If $F(k)$ is non-singular, then we can solve the dynamics equation for $\mathbf{x}(k)$:

$$\mathbf{x}(k) = F(k)^{-1}[\mathbf{x}(k+1) - G(k)\mathbf{u}(k) - \Gamma(k)\mathbf{v}(k)]$$

Eliminating $\mathbf{x}(k)$ from the SRIEs gives the system

$$\begin{bmatrix} \mathbf{0} \\ \boldsymbol{\mathfrak{z}}_x(k) + \mathcal{R}_{xx}(k)F(k)^{-1}G(k)\mathbf{u}(k) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_{vv}(k) & 0 \\ -\mathcal{R}_{xx}(k)F(k)^{-1}\Gamma(k) & \mathcal{R}_{xx}(k)F(k)^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{v}(k) \\ \mathbf{x}(k+1) \end{bmatrix} + \begin{bmatrix} \mathbf{w}_v(k) \\ \mathbf{w}_x(k) \end{bmatrix}$$

from which we'd like to produce an expression for the predicted, transformed state $\bar{\boldsymbol{\mathfrak{z}}}_x(k+1)$.

To avoid inverting the coefficient matrix, we can QR factorize it by finding an orthonormal $T_a(k) \in \mathbf{R}^{(n_x+n_v)\times(n_x+n_v)}$ and upper triangular, non-singular $\bar{\mathcal{R}}_{vv}(k) \in \mathbf{R}^{n_v \times n_v}$ and $\bar{\mathcal{R}}_{xx}(k+1) \in \mathbf{R}^{n_x \times n_x}$ such that

$$\begin{bmatrix} \mathcal{R}_{vv}(k) & 0 \\ -\mathcal{R}_{xx}(k)F(k)^{-1}\Gamma(k) & \mathcal{R}_{xx}(k)F(k)^{-1} \end{bmatrix} = T_a(k)^T \begin{bmatrix} \bar{\mathcal{R}}_{vv}(k) & \bar{\mathcal{R}}_{vx}(k+1) \\ 0 & \bar{\mathcal{R}}_{xx}(k+1) \end{bmatrix}$$

Plugging this into the SRIE system gives

$$\begin{bmatrix} \bar{\boldsymbol{\mathfrak{z}}}_v(k) \\ \bar{\boldsymbol{\mathfrak{z}}}_x(k+1) \end{bmatrix} = \begin{bmatrix} \bar{\mathcal{R}}_{vv}(k) & \bar{\mathcal{R}}_{vx}(k+1) \\ 0 & \bar{\mathcal{R}}_{xx}(k+1) \end{bmatrix} \begin{bmatrix} \mathbf{v}(k) \\ \mathbf{x}(k+1) \end{bmatrix} + \begin{bmatrix} \bar{\mathbf{w}}_v(k) \\ \bar{\mathbf{w}}_x(k+1) \end{bmatrix}$$

where

$$\begin{bmatrix} \bar{\boldsymbol{\mathfrak{z}}}_v(k) \\ \bar{\boldsymbol{\mathfrak{z}}}_x(k+1) \end{bmatrix} = T_a(k) \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\mathfrak{z}}_x(k) + \mathcal{R}_{xx}(k)F(k)^{-1}G(k)\mathbf{u}(k) \end{bmatrix}$$
$$\begin{bmatrix} \bar{\mathbf{w}}_v(k) \\ \bar{\mathbf{w}}_x(k+1) \end{bmatrix} = T_a(k) \begin{bmatrix} \mathbf{w}_v(k) \\ \mathbf{w}_x(k) \end{bmatrix}$$

Note that overbars here indicate transformation, not expectation.

As required, the transformed RVs have zero mean and identity covariance:

$$\mathbf{E}\left[\begin{bmatrix} \bar{\mathbf{w}}_v(k) \\ \bar{\mathbf{w}}_x(k+1) \end{bmatrix} \right] = \mathbf{0}$$
$$\mathbf{E}\left[\begin{bmatrix} \bar{\mathbf{w}}_v(k) \\ \bar{\mathbf{w}}_x(k+1) \end{bmatrix} \begin{bmatrix} \bar{\mathbf{w}}_v(k)^T & \bar{\mathbf{w}}_x(k+1)^T \end{bmatrix} \right] = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$$

Solving the transformed SRIEs for $\bar{\boldsymbol{\mathfrak{z}}}_x(k+1)$ gives

$$\bar{\boldsymbol{\mathfrak{z}}}_x(k+1) = \bar{\mathcal{R}}_{xx}(k+1)\mathbf{x}(k+1) + \bar{\mathbf{w}}_x(k+1)$$

The second SRIE equation,

$$\bar{\boldsymbol{\mathfrak{z}}}_v(k) = \bar{\mathcal{R}}_{vv}(k)\mathbf{v}(k) + \bar{\mathcal{R}}_{vx}(k+1)\mathbf{x}(k+1) + \bar{\mathbf{w}}_v(k)$$

can be set aside until we study smoothing.

6.2.3 Measurement update

How to conduct the measurement update such that the SRIEs are satisfied?

We'll use the Cholesky factorizations of the noise covariance matrices,

$$R_a(k+1)^T R_a(k+1) = R(k+1)$$

where $R_a(k+1) \in \mathbf{R}^{n_z \times n_z}$ is upper triangular and non-singular.

Under the transformations

$$\begin{aligned}
\mathbf{z}_a(k+1) &= R_a(k+1)^{-T}\mathbf{z}(k+1) \in \mathbf{R}^{n_z} \\
\mathbf{w}_a(k+1) &= R_a(k+1)^{-T}\mathbf{w}(k+1) \in \mathbf{R}^{n_z} \\
H_a(k+1) &= R_a(k+1)^{-T}H(k+1) \in \mathbf{R}^{n_z \times n_x}
\end{aligned}$$

the measurement model becomes

$$\mathbf{z}_a(k+1) = H_a(k+1)\mathbf{x}(k+1) + \mathbf{w}_a(k+1)$$

where

$$\mathbf{E}[\mathbf{w}_a(k+1)] = \mathbf{0}, \quad \mathbf{E}[\mathbf{w}_a(k+1)\mathbf{w}_a(k+1)^T] = I$$

Combining the state SRIE and the measurement equation gives the system

$$\begin{bmatrix} \bar{\boldsymbol{\mathfrak{z}}}_x(k+1) \\ \mathbf{z}_a(k+1) \end{bmatrix} = \begin{bmatrix} \bar{\mathcal{R}}_{xx}(k+1) \\ H_a(k+1) \end{bmatrix} \mathbf{x}(k+1) + \begin{bmatrix} \bar{\mathbf{w}}_x(k+1) \\ \mathbf{w}_a(k+1) \end{bmatrix}$$

To avoid inverting the coefficient matrix, we can QR factorize it by finding an orthonormal $T_b(k+1) \in \mathbf{R}^{(n_x+n_v)\times(n_x+n_v)}$ and upper triangular, non-singular $\mathcal{R}_{xx}(k+1) \in \mathbf{R}^{n_x \times n_x}$ such that

$$\begin{bmatrix} \bar{\mathcal{R}}_{xx}(k+1) \\ H_a(k+1) \end{bmatrix} = T_b(k+1)^T \begin{bmatrix} \mathcal{R}_{xx}(k+1) \\ 0 \end{bmatrix}$$

The SRIE-measurement system then becomes

$$\begin{bmatrix} \boldsymbol{\mathfrak{z}}_x(k+1) \\ \boldsymbol{\mathfrak{z}}_r(k+1) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_{xx}(k+1) \\ 0 \end{bmatrix} \mathbf{x}(k+1) + \begin{bmatrix} \mathbf{w}_x(k+1) \\ \mathbf{w}_r(k+1) \end{bmatrix}$$

where

$$\begin{aligned}
\begin{bmatrix} \boldsymbol{\mathfrak{z}}_x(k+1) \\ \boldsymbol{\mathfrak{z}}_r(k+1) \end{bmatrix} &= T_b(k+1) \begin{bmatrix} \bar{\boldsymbol{\mathfrak{z}}}_x(k+1) \\ \mathbf{z}_a(k+1) \end{bmatrix} \\
\begin{bmatrix} \mathbf{w}_x(k+1) \\ \mathbf{w}_r(k+1) \end{bmatrix} &= T_b(k+1) \begin{bmatrix} \bar{\mathbf{w}}_x(k+1) \\ \mathbf{w}_a(k+1) \end{bmatrix}
\end{aligned}$$

As required, the transformed RVs have zero mean and identity covariance:

$$\mathbf{E}\left[\begin{bmatrix}\mathbf{w}_x(k+1)\\ \mathbf{w}_r(k+1)\end{bmatrix}\right] = \mathbf{0}$$

$$\mathbf{E}\left[\begin{bmatrix}\mathbf{w}_x(k+1)\\ \mathbf{w}_r(k+1)\end{bmatrix}\begin{bmatrix}\mathbf{w}_x(k+1)^T & \mathbf{w}_r(k+1)^T\end{bmatrix}\right] = \begin{bmatrix}I & 0\\ 0 & I\end{bmatrix}$$

Solving the transformed SRIE-measurement system for $\boldsymbol{\mathfrak{z}}_x(k+1)$ gives

$$\boldsymbol{\mathfrak{z}}_x(k+1) = \mathcal{R}_{xx}(k+1)\mathbf{x}(k+1) + \mathbf{w}_x(k+1)$$

The second equation,

$$\boldsymbol{\mathfrak{z}}_r(k+1) = \mathbf{w}_r(k+1)$$

gives the measurement residual (hence the subscript r), which is closely related to the innovation in the standard KF algorithm. It can be shown that

$$\frac{1}{2}\boldsymbol{\mathfrak{z}}_r(k+1)^T\boldsymbol{\mathfrak{z}}_r(k+1) = \frac{1}{2}\boldsymbol{\nu}(k+1)^T S(k+1)^{-1}\boldsymbol{\nu}(k+1)$$

The residuals can be used in the tests of filter correctness described in section 5.7.

6.3 The SRIF algorithm

Instead of keeping track of $\hat{\mathbf{x}}(k)$, $P(k)$ and $\boldsymbol{\nu}(k)$, in the SRIF algorithm we keep track of $\boldsymbol{\mathfrak{z}}_x(k)$, $\mathcal{R}_{xx}(k)$, and $\boldsymbol{\mathfrak{z}}_r(k)$.

Given $\mathcal{R}_{xx}(0)^T\mathcal{R}_{xx}(0) = \mathcal{I}(0)$ and $\boldsymbol{\mathfrak{z}}_x(0) = \mathcal{R}_{xx}(0)\hat{\mathbf{x}}(0)$, the SRIF algorithm is:

1. **offline calculations**
   (a) for each k, compute $F(k)^{-1}$
   (b) disturbance Cholesky factorization: for each k, find an upper triangular $\mathcal{R}_{vv}(k) \in \mathbf{R}^{n_v \times n_v}$ such that
   $$\mathcal{R}_{vv}(k)^T\mathcal{R}_{vv}(k) = Q(k)^{-1}$$
   (c) noise Cholesky factorization: for each k, find an upper triangular $R_a(k) \in \mathbf{R}^{n_z \times n_z}$ such that
   $$R_a(k)^T R_a(k) = R(k)$$
   then transform the measurement matrices,
   $$H_a(k) = R_a(k)^{-T}H(k)$$
2. set $k = 0$
3. **dynamic propagation**

(a) QR factorization: find an orthonormal $T_a(k) \in \mathbf{R}^{(n_x+n_v)\times(n_x+n_v)}$ and upper triangular, non-singular $\bar{\mathcal{R}}_{vv}(k) \in \mathbf{R}^{n_v \times n_v}$ and $\bar{\mathcal{R}}_{xx}(k+1) \in \mathbf{R}^{n_x \times n_x}$ such that

$$\begin{bmatrix} \mathcal{R}_{vv}(k) & 0 \\ -\mathcal{R}_{xx}(k)F(k)^{-1}\Gamma(k) & \mathcal{R}_{xx}(k)F(k)^{-1} \end{bmatrix} = T_a(k)^T \begin{bmatrix} \bar{\mathcal{R}}_{vv}(k) & \bar{\mathcal{R}}_{vx}(k+1) \\ 0 & \bar{\mathcal{R}}_{xx}(k+1) \end{bmatrix}$$

(b) predict the (transformed) state

$$\begin{bmatrix} \bar{\boldsymbol{\mathfrak{z}}}_v(k) \\ \bar{\boldsymbol{\mathfrak{z}}}_x(k+1) \end{bmatrix} = T_a(k) \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\mathfrak{z}}_x(k) + \mathcal{R}_{xx}(k)F(k)^{-1}G(k)\mathbf{u}(k) \end{bmatrix}$$

4. **measurement update**

(a) QR factorization: find an orthonormal $T_b(k+1) \in \mathbf{R}^{(n_x+n_z)\times(n_x+n_z)}$ and an upper triangular $\mathcal{R}_{xx}(k+1) \in \mathbf{R}^{n_x \times n_x}$ such that

$$\begin{bmatrix} \bar{\mathcal{R}}_{xx}(k+1) \\ H_a(k+1) \end{bmatrix} = T_b(k+1)^T \begin{bmatrix} \mathcal{R}_{xx}(k+1) \\ 0 \end{bmatrix}$$

(b) compute the (transformed) state estimate

$$\begin{bmatrix} \boldsymbol{\mathfrak{z}}_x(k+1) \\ \boldsymbol{\mathfrak{z}}_r(k+1) \end{bmatrix} = T_b(k+1) \begin{bmatrix} \bar{\boldsymbol{\mathfrak{z}}}_x(k+1) \\ \mathbf{z}_a(k+1) \end{bmatrix}$$

5. **inverse transformation**

(a) recover the state estimate

$$\hat{\mathbf{x}}(k+1) = \mathcal{R}_{xx}(k+1)^{-1}\boldsymbol{\mathfrak{z}}_x(k+1)$$

(b) recover the state estimation error covariance

$$P(k+1) = \mathcal{R}_{xx}(k+1)^{-1}\mathcal{R}_{xx}(k+1)^{-T}$$

6. increment k and return to step 3

6.3.1 Remarks

1. The SRIF algorithm never squares or inverts a matrix unless it's upper triangular (and hence numerically stable to square or invert). The exception to this rule is the dynamics matrix $F(k)$, which we assume to be non-singular and well-conditioned. The SRIF can be modified to accommodate singular dynamics matrices, but the algorithm is much more complicated.

2. The SRIF also never takes the difference of two quantities that may be of similar magnitude.

3. It can be shown that the SRIF and KF are equivalent.

4. Punchline: the SRIF is a numerically stable implementation of the optimal filter for the LG system.

5. Other numerically stable KF implementations exist, such as the square-root covariance filter (SRCF) presented in Bar-Shalom. The SRCF is generally better suited to high-dimensional problems than the SRIF.

7 Smoothing

In filtering, we estimate the current state of a dynamical system based on information up to and including the current measurement. In smoothing, we use current information to estimate a past state. A common application of smoothing is processing data from an experiment after the fact.

The general smoothing problem is to compute the estimate

$$\hat{\mathbf{x}}(k|j) = \mathbf{E}[\mathbf{x}(k)|\mathbf{z}^j]$$

for some $j > k$.

Smoothing problems come in three flavors.

1. In **fixed point smoothing**, k is a fixed time of interest and we continually refine our estimate of $\mathbf{x}(k)$ as more and more data become available. We compute

$$\hat{\mathbf{x}}(k|j),\ \hat{\mathbf{x}}(k|j+1),\ \hat{\mathbf{x}}(k|j+1),\ \ldots$$

2. In **fixed lag smoothing**, we estimate the state based on data that run a fixed interval $i > 0$ into the future. We compute

$$\hat{\mathbf{x}}(k|k+i),\ \hat{\mathbf{x}}(k+1|k+i+1),\ \hat{\mathbf{x}}(k+2|k+i+2),\ \ldots$$

3. In **fixed interval smoothing**, we have a batch of data $\mathbf{z}(1), \ldots, \mathbf{z}(N)$ and we use all the data to estimate the state at each stage. We compute

$$\hat{\mathbf{x}}(0|N),\ \hat{\mathbf{x}}(1|N),\ \hat{\mathbf{x}}(2|N),\ \ldots,\ \hat{\mathbf{x}}(N|N)$$

In this section, we'll solve the fixed interval smoothing problem. Incidentally, the fixed interval smoothing solution will provide a brute-force approach for the fixed point and fixed lag smoothing problems, although better algorithms for these problems exist.

7.1 Notation

Just like filtering, there are a variety of notational conventions for smoothing in the literature. A few are listed here; these notes will use the last notation presented.

- smoothed state estimate

$$\begin{aligned}\mathbf{E}[\mathbf{x}(k)|\mathbf{z}^N] &\equiv \hat{\mathbf{x}}(k|\mathbf{z}^N) \equiv \hat{\mathbf{x}}(k|N) \\ &\equiv \mathbf{x}^*(k)\end{aligned}$$

- smoothed state estimate covariance

$$\begin{aligned}\mathbf{E}[(\mathbf{x}(k) - \hat{\mathbf{x}}(k|\mathbf{z}^N))(\mathbf{x}(k) - \hat{\mathbf{x}}(k|\mathbf{z}^N))^T \mid \mathbf{z}^N] &\equiv P(k|\mathbf{z}^N) \equiv P(k|N) \\ &\equiv P^*(k)\end{aligned}$$

- smoothed disturbance

$$\begin{aligned}\mathbf{E}[\mathbf{v}(k)|\mathbf{z}^N] &\equiv \hat{\mathbf{v}}(k|\mathbf{z}^N) \equiv \hat{\mathbf{v}}(k|N) \\ &\equiv \mathbf{v}^*(k)\end{aligned}$$

- smoothed disturbance covariance

$$\begin{aligned}\mathbf{E}[(\mathbf{v}(k) - \hat{\mathbf{v}}(k|\mathbf{z}^N))(\mathbf{v}(k) - \hat{\mathbf{v}}(k|\mathbf{z}^N))^T \mid \mathbf{z}^N] &\equiv P_{vv}(k|\mathbf{z}^N) \equiv P_{vv}(k|N) \\ &\equiv P^*_{vv}(k)\end{aligned}$$

- smoothed disturbance-state covariance

$$\begin{aligned}\mathbf{E}[(\mathbf{v}(k) - \hat{\mathbf{v}}(k|\mathbf{z}^N))(\mathbf{x}(k) - \hat{\mathbf{x}}(k|\mathbf{z}^N))^T \mid \mathbf{z}^N] &\equiv P_{vx}(k|\mathbf{z}^N) \equiv P_{vx}(k|N) \\ &\equiv P^*_{vx}(k)\end{aligned}$$

7.2 Covariance-based fixed interval smoothing

The covariance-based fixed interval smoother presented here consists of a forward and backward pass. Filters of this type are known as Rauch-Tung-Striebel (RTS) smoothers. There are other approaches to smoothing. For example, some smoothing algorithms run two filters simultaneously, one starting from $k = 0$ and the other from $k = N$ and converging in the middle. In practice, however, most smoothing is done using RTS methods.

The covariance-based RTS smoother consists of the following steps.

1. **forward (filtering) pass**: given $\hat{\mathbf{x}}(0)$ and $P(0)$, solve the filtering problem and store the resulting $\hat{\mathbf{x}}(k), P(k), \bar{\mathbf{x}}(k)$, and $\bar{P}(k)$ for all $k \in \{1, \ldots, N\}$.
2. **backward (smoothing) pass**
   (a) initialize $\mathbf{x}^*(N) = \hat{\mathbf{x}}(N)$ and $P^*(N) = P(N)$
   (b) set $k = N - 1$
   (c) determine $\mathbf{v}^*(k)$, $P^*_{vv}(k)$, and $P^*_{vx}(k+1)$ from $\hat{\mathbf{x}}^*(k+1)$ and $P^*(k+1)$.
   (d) determine $\mathbf{x}^*(k)$ and $P^*(k)$ from $\mathbf{v}^*(k)$, $P^*_{vv}(k)$, and $P^*_{vx}(k+1)$
   (e) if $k = 0$, stop; otherwise, decrement k and go to step (c)

Chapters 5 and 6 presented several filtering methods, so we only need to figure out steps (c) and (d) in the smoothing pass. The next two subsections show how to accomplish these.

7.2.1 Smoothed disturbance and associated covariances

This section gives a sketch of the MAP derivation of the formulas for step 2(c).

- **smoothed disturbance**

– using Bayes' rule and the definition of the conditional *pdf*, express the posterior *pdf* as

$$p(\mathbf{v}(k), \mathbf{x}(k+1)|\mathbf{z}^N) = \frac{p(\mathbf{v}(k), \mathbf{x}(k+1)|\mathbf{z}^{k+1}) p(\mathbf{z}_{k+1}^N|\mathbf{x}(k+1))}{p(\mathbf{z}_{k+1}^N|\mathbf{z}^{k+1})}$$

– define the cost

$$\begin{aligned} J_s^k(\mathbf{v}(k), \mathbf{x}(k+1)) &= -\ln p(\mathbf{v}(k), \mathbf{x}(k+1)|\mathbf{z}^N) \\ &= -\ln p(\mathbf{v}(k), \mathbf{x}(k+1)|\mathbf{z}^{k+1}) - \ln p(\mathbf{z}_{k+1}^N|\mathbf{x}(k+1)) + \ln p(\mathbf{z}_{k+1}^N|\mathbf{z}^{k+1}) \end{aligned}$$

so that (from the definition of the MAPE)

$$\begin{aligned} \mathbf{v}^*(k) &= \arg\max_{\mathbf{v}(k)} p(\mathbf{v}(k), \mathbf{x}^*(k+1)|\mathbf{z}^N) \\ &= \arg\min_{\mathbf{v}(k)} J_s^k(\mathbf{v}(k), \mathbf{x}^*(k+1)) \\ &= \arg\min_{\mathbf{v}(k)} \big\{ -\ln p(\mathbf{v}(k), \mathbf{x}^*(k+1)|\mathbf{z}^{k+1}) - \ln p(\mathbf{z}_{k+1}^N|\mathbf{x}^*(k+1)) \big\} \end{aligned}$$

– note that we've already solved this problem in the KF MAP derivation; the result is

$$\mathbf{v}^*(k) = Q(k)\Gamma(k)^T P(k+1)^{-1} \big[\mathbf{x}^*(k+1) - \bar{\mathbf{x}}(k+1)\big]$$

- **covariances**

– note that

$$\begin{bmatrix} P_{vv}^*(k) & P_{vx}^*(k+1) \\ P_{vx}^*(k+1)^T & P^*(k+1) \end{bmatrix} = \left(\frac{\partial^2 J_s^k}{\partial \mathbf{a}^2} \right)^{-1}$$

where

$$\mathbf{a} = \begin{bmatrix} \mathbf{v}(k) \\ \mathbf{x}^*(k+1) \end{bmatrix}$$

– write the Hessian of J_s^k as

$$\frac{\partial^2 J_s^k}{\partial \mathbf{a}^2} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

where

$$C_{11} = \frac{\partial^2 J_s^k}{\partial \mathbf{v}(k)^2} = Q(k)^{-1} + \Gamma(k)^T F(k)^{-1} P(k)^{-1} F(k)^{-1} \Gamma(k)$$

and

$$C_{12} = \frac{\partial^2 J_s^k}{\partial \mathbf{v}(k) \partial \mathbf{x}(k+1)} = -\Gamma(k)^T F(k)^{-1} P(k)^{-1} F(k)^{-1}$$

– using the block inversion formula, solve for

$$\begin{aligned} P^*_{vv}(k) &= C_{11}^{-1} + C_{11}^{-1} C_{12} P^*(k+1) C_{12}^T C_{11}^{-1} \\ P^*_{vx}(k+1) &= -C_{11}^{-1} C_{12} P^*(k+1) \end{aligned}$$

and simplify to produce

$$\begin{aligned} P^*_{vv}(k) &= Q(k) - Q(k)\Gamma(k)^T \bar{P}(k+1)^{-1} \big[\bar{P}(k+1) - P^*(k+1)\big] \\ &\quad * \bar{P}(k+1)^{-1} \Gamma(k) Q(k) \\ P^*_{vx}(k+1) &= Q(k)\Gamma(k)^T \bar{P}(k+1)^{-1} P^*(k+1) \end{aligned}$$

7.2.2 Smoothed state estimate and covariance

This section gives a sketch of the MAP derivation of the formulas for step (d) in the smoothing pass.

- **smoothed state estimate**: take the expectation, conditioned on $\mathbf{z}^N$, of the inverse dynamics to find that

$$\mathbf{x}^*(k) = F(k)^{-1} \big[\mathbf{x}^*(k+1) - G(k)\mathbf{u}(k) - \Gamma(k)\mathbf{v}^*(k)\big]$$

- **covariance**
  - define the error

$$\tilde{\mathbf{x}}^*(k) = \mathbf{x}(k) - \mathbf{x}^*(k) = F(k)^{-1} \big[\tilde{\mathbf{x}}^*(k+1) - \Gamma(k)\tilde{\mathbf{v}}^*(k)\big]$$

    where

$$\tilde{\mathbf{v}}^*(k) = \mathbf{v}(k) - \mathbf{v}^*(k)$$

  - compute

$$\begin{aligned} P^*(k) &= \mathbf{E}[\tilde{\mathbf{x}}^*(k)\tilde{\mathbf{x}}^*(k)^T | \mathbf{z}^N] \\ &= F(k)^{-1} \big[P^*_{k+1} - P^*_{vx}(k+1)^T \Gamma(k)^T \\ &\quad - \Gamma(k) P^*_{vx}(k+1) + \Gamma(k) P^*_{vv}(k) \Gamma(k)^T \big] F(k)^{-T} \end{aligned}$$

    and simplify

7.2.3 Summary

The covariance-based smoothing algorithm is

1. **forward (filtering) pass**: given $\hat{\mathbf{x}}(0)$ and $P(0)$, solve the filtering problem and store the resulting $\hat{\mathbf{x}}(k), P(k), \bar{\mathbf{x}}(k)$, and $\bar{P}(k)$ for all $k \in \{1, \ldots, N\}$.

2. **backward (smoothing) pass**

   (a) initialize $\mathbf{x}^*(N) = \hat{\mathbf{x}}(N)$ and $P^*(N) = P(N)$

   (b) set $k = N - 1$

   (c) compute the smoother gain

$$C(k) = P(k)F(k)^T \bar{P}(k+1)^{-1}$$

   (d) compute the smoothed state estimate

$$\mathbf{x}^*(k) = \hat{\mathbf{x}}(k) + C(k)\big[\mathbf{x}^*(k+1) - \bar{\mathbf{x}}(k+1)\big]$$

   (e) compute the smoothed state covariance

$$P^*(k) = P(k) - C(k)\big[\bar{P}(k+1) - P^*(k+1)\big]C(k)^T$$

   (f) if $k = 0$, stop; otherwise, decrement k and go to step (c)

7.3 Information-based fixed interval smoothing

The covariance-based fixed interval smoother algorithm includes matrix squares and subtractions, which can be numerically unstable. We'll address this once again using square-root techniques.

The square-root information smoother (SRIS) algorithm consists of the following steps:

1. **forward (filtering) pass**: given $\hat{\mathbf{x}}(0)$ and $P(0)$, solve the SRIF problem and store $\mathcal{R}_{xx}(k)$, $\mathcal{R}_{vv}(k)$ and $\bar{\boldsymbol{\mathfrak{z}}}_v(k)$ for all $k \in \{0, \ldots, N-1\}$ as well as $\boldsymbol{\mathfrak{z}}_x(N)$

2. **backward (smoothing) pass**

   (a) Initialize $\boldsymbol{\mathfrak{z}}_x^*(N) = \boldsymbol{\mathfrak{z}}_x(N)$ and $\mathcal{R}_{xx}^*(N) = \mathcal{R}_{xx}(N)$ and compute

$$\mathbf{x}^*(N) = \mathcal{R}_{xx}^*(N)^{-1}\boldsymbol{\mathfrak{z}}_x^*(N)$$

   and

$$P^*(N) = \mathcal{R}_{xx}^*(N)^{-1}\mathcal{R}_{xx}^*(N)^{-T}$$

   (b) set $k = N - 1$

   (c) use dynamics to eliminate $\mathbf{x}(k+1)$ from the smoothed SRIEs for $\mathbf{v}(k)$ and $\mathbf{x}(k+1)$ to yield SRIEs for $\mathbf{v}(k)$ and $\mathbf{x}(k)$

   (d) recover the smoothed state and its covariance:

$$\mathbf{x}^*(k) = \mathcal{R}_{xx}^*(k)^{-1}\boldsymbol{\mathfrak{z}}_x^*(k)$$
$$P^*(k) = \mathcal{R}_{xx}^*(k)^{-1}\mathcal{R}_{xx}^*(k)^{-T}$$

   (e) if $k = 0$, stop; otherwise, decrement k and go to (c)

The next subsection shows how to accomplish step 2(c).

7.3.1 Eliminating $\mathbf{x}(k+1)$

At stage k in the smoothing pass, we have the SRIE system

$$\begin{bmatrix} \bar{\boldsymbol{\mathfrak{z}}}_v(k) \\ \boldsymbol{\mathfrak{z}}_x^*(k+1) \end{bmatrix} = \begin{bmatrix} \bar{\mathcal{R}}_{vv}(k) & \bar{\mathcal{R}}_{vx}(k) \\ 0 & \mathcal{R}_{xx}^*(k+1) \end{bmatrix} \begin{bmatrix} \mathbf{v}(k) \\ \mathbf{x}(k+1) \end{bmatrix} + \begin{bmatrix} \bar{\mathbf{w}}_v(k) \\ \mathbf{w}_x(k+1) \end{bmatrix}$$

Replacing $\mathbf{x}(k+1)$ with $F(k)\mathbf{x}(k) + B(k)\mathbf{u}(k) + \Gamma(k)\mathbf{v}(k)$ gives

$$\begin{bmatrix} \bar{\boldsymbol{\mathfrak{z}}}_v(k) - \bar{\mathcal{R}}_{vx}(k+1)G(k)\mathbf{u}(k) \\ \boldsymbol{\mathfrak{z}}_x(k+1) - \mathcal{R}_{xx}^*(k+1)G(k)\mathbf{u}(k) \end{bmatrix} = \begin{bmatrix} \bar{\mathcal{R}}_{vv}(k) + \bar{\mathcal{R}}_{xx}(k+1)\Gamma(k) & \bar{\mathcal{R}}_{vx}(k+1)F(k) \\ \mathcal{R}_{xx}^*(k+1)\Gamma(k) & \mathcal{R}_{xx}^*(k+1)F(k) \end{bmatrix} \begin{bmatrix} \mathbf{v}(k) \\ \mathbf{x}(k) \end{bmatrix} + \begin{bmatrix} \bar{\mathbf{w}}_v(k) \\ \mathbf{w}_x^*(k+1) \end{bmatrix}$$

where

$$\begin{bmatrix} \bar{\mathbf{w}}_v(k) \\ \mathbf{w}_x^*(k+1) \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, I)$$

To avoid inverting the coefficient matrix, we can QR factorize it by finding an orthonormal $T_c(k) \in \mathbf{R}^{(n_x+n_v)\times(n_x+n_v)}$ and upper triangular, non-singular $\mathcal{R}_{vv}^*(k) \in \mathbf{R}^{n_v \times n_v}$ and $\mathcal{R}_{xx}^*(k) \in \mathbf{R}^{n_x \times n_x}$ such that

$$\begin{bmatrix} \bar{\mathcal{R}}_{vv}(k) + \bar{\mathcal{R}}_{xx}(k+1)\Gamma(k) & \bar{\mathcal{R}}_{vx}(k+1)F(k) \\ \mathcal{R}_{xx}^*(k+1)\Gamma(k) & \mathcal{R}_{xx}^*(k+1)F(k) \end{bmatrix} = T_c(k)^T \begin{bmatrix} \mathcal{R}_{vv}^*(k) & \mathcal{R}_{vx}^*(k) \\ 0 & \mathcal{R}_{xx}^*(k) \end{bmatrix}$$

The transformed, smoothed noise and state are given by

$$\begin{bmatrix} \boldsymbol{\mathfrak{z}}_v^*(k) \\ \boldsymbol{\mathfrak{z}}_x^*(k) \end{bmatrix} = T_c(k) \begin{bmatrix} \bar{\boldsymbol{\mathfrak{z}}}_v(k) - \bar{\mathcal{R}}_{vx}(k+1)G(k)\mathbf{u}(k) \\ \boldsymbol{\mathfrak{z}}_x^*(k+1) - \mathcal{R}_{xx}^*(k+1)G(k)\mathbf{u}(k) \end{bmatrix}$$

With $\mathcal{R}_{vv}^*(k)$, $\mathcal{R}_{xx}^*(k)$, $\boldsymbol{\mathfrak{z}}_v^*(k)$ and $\boldsymbol{\mathfrak{z}}_x^*(k)$, we can recover the smoothed disturbance and state and their covariances.

7.3.2 SRIS summary

The SRIS algorithm is:

1. **forward (filtering) pass**: given $\hat{\mathbf{x}}(0)$ and $P(0)$, solve the SRIF problem and store $\bar{\mathcal{R}}_{vv}(k)$, $\bar{\mathcal{R}}_{vx}(k)$, $\bar{\mathcal{R}}_{xx}(k)$, $\bar{\boldsymbol{\mathfrak{z}}}_v(k)$ for each k. Also store $\mathcal{R}_{xx}(N)$ and $\boldsymbol{\mathfrak{z}}_x(N)$.

2. **backward (smoothing) pass**

   (a) Initialize $\boldsymbol{\mathfrak{z}}_x^*(N) = \boldsymbol{\mathfrak{z}}_x(N)$ and $\mathcal{R}_{xx}^*(N) = \mathcal{R}_{xx}(N)$ and compute

$$\mathbf{x}^*(N) = \mathcal{R}_{xx}^*(N)^{-1}\boldsymbol{\mathfrak{z}}_x^*(N)$$
$$P^*(N) = \mathcal{R}_{xx}^*(N)^{-1}\mathcal{R}_{xx}^*(N)^{-T}$$

(b) set $k = N - 1$

(c) eliminate $\mathbf{x}(k+1)$ from the SRIE system

i. QR factorization: find an orthonormal $T_c(k) \in \mathbf{R}^{(n_x+n_v)\times(n_x+n_v)}$ and upper triangular, non-singular $\mathcal{R}^*_{vv}(k) \in \mathbf{R}^{n_v \times n_v}$ and $\mathcal{R}^*_{xx}(k) \in \mathbf{R}^{n_x \times n_x}$ such that

$$T_c(k)^T \begin{bmatrix} \mathcal{R}^*_{vv}(k) & \mathcal{R}^*_{vx}(k) \\ 0 & \mathcal{R}^*_{xx}(k) \end{bmatrix} = \begin{bmatrix} \bar{\mathcal{R}}_{vv}(k) + \bar{\mathcal{R}}_{xx}(k+1)\Gamma(k) & \bar{\mathcal{R}}_{vx}(k+1)F(k) \\ \mathcal{R}^*_{xx}(k+1)\Gamma(k) & \mathcal{R}^*_{xx}(k+1)F(k) \end{bmatrix}$$

ii. Compute the transformed, smoothed disturbance and state:

$$\begin{bmatrix} \boldsymbol{\mathfrak{z}}^*_v(k) \\ \boldsymbol{\mathfrak{z}}^*_x(k) \end{bmatrix} = T_c(k) \begin{bmatrix} \bar{\boldsymbol{\mathfrak{z}}}_v(k) - \bar{\mathcal{R}}_{vx}(k+1)G(k)\mathbf{u}(k) \\ \boldsymbol{\mathfrak{z}}^*_x(k+1) - \mathcal{R}^*_{xx}(k+1)G(k)\mathbf{u}(k) \end{bmatrix}$$

(d) recover the smoothed state and its covariance:

$$\begin{aligned} \mathbf{x}^*(k) &= \mathcal{R}^*_{xx}(k)^{-1}\boldsymbol{\mathfrak{z}}^*_x(k) \\ P^*(k) &= \mathcal{R}^*_{xx}(k)^{-1}\mathcal{R}^*_{xx}(k)^{-T} \end{aligned}$$

and (if desired) the smoothed disturbance and associated covariances:

$$\begin{aligned} \mathbf{v}^*(k) &= \mathcal{R}^*_{vv}(k)^{-1}\big[\boldsymbol{\mathfrak{z}}^*_v(k) - \mathcal{R}^*_{vx}(k)\mathbf{x}^*(k)\big] \\ P^*_{vv}(k) &= \mathcal{R}^*_{vv}(k)^{-1}\big[I + \mathcal{R}^*_{vx}(k)\mathcal{R}^*_{xx}(k)^{-1}\mathcal{R}^*_{xx}(k)^{-T}\mathcal{R}^*_{vx}(k)^T\big]\mathcal{R}^*_{vv}(k)^{-T} \\ P^*_{vx}(k) &= -\mathcal{R}^*_{vv}(k)^{-1}\mathcal{R}^*_{vx}(k)\mathcal{R}^*_{xx}(k)^{-1}\mathcal{R}^*_{xx}(k)^{-T} \end{aligned}$$

(e) if $k = 0$, stop; otherwise, decrement k and go to (c)

NB. The smoothed disturbances may be useful for refining the probabilistic model of the state disturbance, which in most applications is not particularly well modeled.

8 State estimation in stochastic nonlinear dynamical systems

The filtering and smoothing algorithms developed in sections 5, 6 and 7 apply only to discrete-time systems that

- have linear dynamics and linear measurement maps
- contain only Gaussian random variables

In practice, however, one often encounters systems where either the dynamics or the measurement map is nonlinear. In such cases, even if all the variables injected into the system – the initial state and the disturbance and noise sequences – are Gaussian, the state and measurements will not, in general, be Gaussian.

This section will develop state estimation techniques for such systems. We'll begin by showing how to discretize a continuous-time nonlinear stochastic system, then we'll develop the following nonlinear filters:

- the extended Kalman filter
- the iterated extended Kalman filter
- the unscented Kalman filter
- the particle filter

8.1 Discretization of nonlinear differential equations

Consider the nonlinear differential equation

$$\dot{\mathbf{x}}(t) = \boldsymbol{f}(t, \mathbf{x}(t), \mathbf{u}(t), \tilde{\mathbf{v}}(t)) \tag{7}$$

where the disturbance sequence is zero mean and white:

$$\mathbf{E}[\tilde{\mathbf{v}}(t)] = \mathbf{0}, \quad \mathbf{E}[\tilde{\mathbf{v}}(t)\tilde{\mathbf{v}}(\tau)^T] = \delta(t-\tau)\tilde{Q}(t)$$

We restrict our attention to systems where the disturbance enters only as an *additive*[3] linear combination, i.e. to systems where

$$\boldsymbol{f}(t, \mathbf{x}(t), \mathbf{u}(t), \tilde{\mathbf{v}}(t)) = \boldsymbol{\psi}(t, \mathbf{x}(t), \mathbf{u}(t)) + D(t)\tilde{\mathbf{v}}(t)$$

for some nonlinear function $\boldsymbol{\psi} : \mathbf{R} \times \mathbf{R}^{n_x} \times \mathbf{R}^{n_u}$ and some matrix $D(t) \in \mathbf{R}^{n_x} \times \mathbf{R}^{n_v}$.

[3]Since continuous white noise has infinite (delta-function) covariance, theoretical problems may arise if it enters nonlinearly.

Note that the scripted symbol $\boldsymbol{f}$ is not the same as the printed symbol $\mathbf{f}$. The scripted version denotes the continuous-time dynamics map. The printed version will represent the dynamics map of the discretized system.

In the sequel we will show how to convert the ODE (7) into a difference equation. Note that a nonlinear continuous-time measurement map can be discretized using the methods in Section 4.2.2.

Suppose we divide up the time axis into uniform intervals of length Δt, and we apply to system (7) the zero-order hold controls

$$\mathbf{u}(t) = \begin{cases} \vdots & \\ \mathbf{u}(k-1), & t \in [(k-1)\Delta t, k\Delta t) \\ \mathbf{u}(k), & t \in [k\Delta t, (k+1)\Delta t) \\ \mathbf{u}(k+1), & t \in [(k+1)\Delta t, (k+2)\Delta t) \\ \vdots & \end{cases}$$

Suppose as well that at $t = k\Delta t$, we know the true state $\mathbf{x}(k\Delta t) \equiv \mathbf{x}(k)$ exactly. If we can find an expression for $\mathbf{x}((k+1)\Delta t) \equiv \mathbf{x}(k+1)$ in terms of $\mathbf{x}(k)$, then we will have discretized the continuous-time dynamics.

If Δt is small, then we can plausibly model the true, time-varying disturbance $\tilde{\mathbf{v}}(t)$ as approximately constant over each time step,

$$\tilde{\mathbf{v}}(t) \approx \begin{cases} \vdots & \\ \mathbf{v}(k-1), & t \in [(k-1)\Delta t, k\Delta t) \\ \mathbf{v}(k), & t \in [k\Delta t, (k+1)\Delta t) \\ \mathbf{v}(k+1), & t \in [(k+1)\Delta t, (k+2)\Delta t) \\ \vdots & \end{cases}$$

where the $\mathbf{v}(k)$ are zero mean and white,

$$\mathbf{E}[\mathbf{v}(k)] = \mathbf{0}, \quad \mathbf{E}[\mathbf{v}(k)\mathbf{v}(j)^T] = \delta_{kj} Q(k)$$

NB. By comparing the random components of the forward Euler approximation of the dynamics in 7 and the random component of the approximate dynamics of $\mathbf{x}_k(t)$ derived below, one can show that the correct covariance relationship (for small Δt) is

$$Q(k) = \frac{1}{\Delta t} \tilde{Q}(k\Delta t)$$

Let $\mathbf{x}_k(t)$ be the approximate state of the system driven by the approximate noise $\mathbf{v}(k)$ during the k^{th} time step, $t \in [k\Delta t, (k+1)\Delta t)$. Note that $\mathbf{x}_k(t) \neq \mathbf{x}(t)$ in general. Although both $\mathbf{x}(t)$ and $\mathbf{x}_k(t)$ are driven by the zero-order hold control $\mathbf{u}(k)$, the true state $\mathbf{x}(t)$ is driven by the true disturbance $\tilde{\mathbf{v}}(t)$ which may vary between $t = k\Delta t$ and $t = (k+1)\Delta t$. On

the other hand, the approximate state $\mathbf{x}_k(t)$ is driven by the approximate noise $\mathbf{v}(k)$, which is constant throughout the k^{th} time step.

Given these definitions, we can propagate $\mathbf{x}_k(t)$ forward in time by solving the IVP

$$\begin{aligned} \dot{\mathbf{x}}_k(t) &= \boldsymbol{f}(t, \mathbf{x}_k(t), \mathbf{u}(k), \mathbf{v}(k)) \\ \mathbf{x}_k(k\Delta t) &= \mathbf{x}(k) \end{aligned} \tag{8}$$

There are fast, accurate techniques for solving such nonlinear IVPs. The most popular is the Runge-Kutta fourth-order method.

If the dynamics function $\boldsymbol{f}$ is known for all $t \in [k\Delta t, (k+1)\Delta t)$, then we can consider this procedure of numerical integration as a function that takes as inputs

- the initial condition $\mathbf{x}(k)$
- the control $\mathbf{u}(k)$
- the disturbance $\mathbf{v}(k)$

and that produces the output $\mathbf{x}(k+1)$. Thus, numerical integration (along with the approximation of $\tilde{\mathbf{v}}(t)$) serves exactly the role of $\mathbf{f}$ in the nonlinear difference equation

$$\mathbf{x}(k+1) = \mathbf{f}(k, \mathbf{x}(k), \mathbf{u}(k), \mathbf{v}(k))$$

so our task of discretizing the nonlinear ODE is complete.

8.1.1 Derivatives of the discrete-time dynamics

In nonlinear filtering algorithms for discrete-time systems, we'll also need the partial derivatives of the discrete-time dynamics function $\mathbf{f}$ with respect to $\mathbf{x}(k)$ and $\mathbf{v}(k)$. These will serve roles analogous to those of $F(k)$ and $\Gamma(k)$ in the covariance dynamic propagation formula.

Differentiating the differential equation in (8) with respect to $\mathbf{x}(k)$ gives

$$\begin{aligned} & \frac{\partial}{\partial \mathbf{x}(k)}\left[\dot{\mathbf{x}}_k(t)\right] = \frac{\partial}{\partial \mathbf{x}(k)}\left[\boldsymbol{f}(t, \mathbf{x}_k(t), \mathbf{u}(k), \mathbf{v}(k))\right] \\ \iff \quad & \frac{\mathrm{d}}{\mathrm{d}t}\left[\frac{\partial \mathbf{x}_k(t)}{\partial \mathbf{x}(k)}\right] = \frac{\partial}{\partial \mathbf{x}_k(t)}\left[\boldsymbol{f}(t, \mathbf{x}_k(t), \mathbf{u}(k), \mathbf{v}(k))\right]\left[\frac{\partial \mathbf{x}_k(t)}{\partial \mathbf{x}(k)}\right] \end{aligned}$$

Differentiating the initial condition with respect to $\mathbf{x}(k)$ gives

$$\frac{\partial}{\partial \mathbf{x}(k)}\left[\mathbf{x}_k(k\Delta t)\right] = \frac{\partial}{\partial \mathbf{x}(k)}\left[\mathbf{x}(k)\right] = I$$

For convenience, let

$$A(t) = \frac{\partial}{\partial \mathbf{x}_k(t)}\left[\boldsymbol{f}(t, \mathbf{x}_k(t), \mathbf{u}(k), \mathbf{v}(k))\right] \in \mathbf{R}^{n_x \times n_x}$$

Then we have the following IVP for the desired partial derivative,

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[\frac{\partial \mathbf{x}_k(t)}{\partial \mathbf{x}(k)}\right] = A(t)\frac{\partial \mathbf{x}_k(t)}{\partial \mathbf{x}(k)}$$
$$\left.\frac{\partial \mathbf{x}_k(t)}{\partial \mathbf{x}(k)}\right|_{k\Delta t} = I$$

which is a state transition IVP like the one in section 5. We can numerically integrate forward in time to $t = (k+1)\Delta t$, and then set

$$\left.\frac{\partial \mathbf{x}_k(t)}{\partial \mathbf{x}(k)}\right|_{t=(k+1)\Delta t} \equiv \frac{\partial}{\partial \mathbf{x}(k)}\mathbf{f}(k, \mathbf{x}(k), \mathbf{u}(k), \mathbf{v}(k))$$

which gives the partial of the discrete-time dynamics with respect to $\mathbf{x}(k)$.

We can find the partial of $\mathbf{f}$ with respect to $\mathbf{v}(k)$ similarly. Differentiating the ODE in (8) with respect to $\mathbf{v}(k)$ gives

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{v}(k)}\left[\dot{\mathbf{x}}_k(t)\right] = \frac{\partial}{\partial \mathbf{v}(k)}\left[\boldsymbol{f}(t, \mathbf{x}_k(t), \mathbf{u}(k), \mathbf{v}(k))\right] \\
\iff &\frac{\mathrm{d}}{\mathrm{d}t}\left[\frac{\partial \mathbf{x}_k(t)}{\partial \mathbf{v}(k)}\right] = \frac{\partial}{\partial \mathbf{x}_k(t)}\left[\boldsymbol{f}(t, \mathbf{x}_k(t), \mathbf{u}(k), \mathbf{v}(k))\right]\left[\frac{\partial \mathbf{x}_k(t)}{\partial \mathbf{v}(k)}\right] + \frac{\partial}{\partial \mathbf{v}(k)}\left[\boldsymbol{f}(t, \mathbf{x}_k(t), \mathbf{u}(k), \mathbf{v}(k))\right] \\
\iff &\frac{\mathrm{d}}{\mathrm{d}t}\left[\frac{\partial \mathbf{x}_k(t)}{\partial \mathbf{v}(k)}\right] = A(t)\frac{\partial \mathbf{x}_k(t)}{\partial \mathbf{v}(k)} + D(t)
\end{aligned}$$

Differentiating the initial condition gives

$$\frac{\partial}{\partial \mathbf{v}(k)}\left[\mathbf{x}_k(k\Delta t)\right] = \frac{\partial}{\partial \mathbf{v}(k)}\left[\mathbf{x}(k)\right] = 0$$

Thus, we have another matrix IVP:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[\frac{\partial \mathbf{x}_k(t)}{\partial \mathbf{v}(k)}\right] = A(t)\frac{\partial \mathbf{x}_k(t)}{\partial \mathbf{v}(k)} + D(t)$$
$$\left.\frac{\partial \mathbf{x}_k(t)}{\partial \mathbf{v}(k)}\right|_{k\Delta t} = 0$$

which we can numerically integrate to produce

$$\left.\frac{\partial \mathbf{x}_k(t)}{\partial \mathbf{v}(k)}\right|_{t=(k+1)\Delta t} \equiv \frac{\partial}{\partial \mathbf{v}(k)}\mathbf{f}(k, \mathbf{x}(k), \mathbf{u}(k), \mathbf{v}(k))$$

8.1.2 One-shot discretization algorithm

This subsection presents an algorithm for simultaneously computing the discretized dynamics map and its partials with respect to $\mathbf{x}(k)$ and $\mathbf{v}(k)$.

For convenience, define the following column decompositions:

$$\frac{\partial \mathbf{x}_k(t)}{\partial \mathbf{x}(k)} = \begin{bmatrix} \boldsymbol{\phi}_1(t) & \boldsymbol{\phi}_2(t) & \dots & \boldsymbol{\phi}_{n_x}(t) \end{bmatrix} \in \mathbf{R}^{n_x \times n_x}$$

where $\boldsymbol{\phi}_j(t) \in \mathbf{R}^{n_x}$, and

$$\frac{\partial \mathbf{x}_k(t)}{\partial \mathbf{v}(k)} = \begin{bmatrix} \boldsymbol{\gamma}_1(t) & \boldsymbol{\gamma}_2(t) & \dots & \boldsymbol{\gamma}_{n_v}(t) \end{bmatrix} \in \mathbf{R}^{n_x \times n_v}$$

where $\boldsymbol{\gamma}_j(t) \in \mathbf{R}^{n_x}$.

Define the stacked vector

$$\mathbf{x}_{\text{big}}(t) = \begin{bmatrix} \mathbf{x}_k(t) \\ \boldsymbol{\phi}_1(t) \\ \vdots \\ \boldsymbol{\phi}_{n_x}(t) \\ \boldsymbol{\gamma}_1(t) \\ \vdots \\ \boldsymbol{\gamma}_{n_v}(t) \end{bmatrix} \in \mathbf{R}^{n_x(1+n_x+n_v)}$$

We will numerically integrate a single IVP for $\mathbf{x}_{\text{big}}(t)$ to produce $\mathbf{x}_{\text{big}}((k+1)\Delta t)$. Define the standard Cartesian basis vectors $\mathbf{e}_j \in \mathbf{R}^{n_x}$ such that

$$\begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \dots & \mathbf{e}_{n_x} \end{bmatrix} = I$$

Then the initial condition for the IVP is

$$\mathbf{x}_{\text{big}}(k\Delta t) = \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_{n_x} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}$$

We also need the lifted continuous-time dynamics map $\boldsymbol{f}_{\text{big}}$ such that

$$\dot{\mathbf{x}}_{\text{big}} = \boldsymbol{f}_{\text{big}}(t, \mathbf{x}_{\text{big}}(t), \mathbf{u}(k), \mathbf{v}(k))$$

With the above definition of $A(t)$,

$$A(t) = \frac{\partial}{\partial \mathbf{x}_k(t)} \Big[\boldsymbol{f}(t, \mathbf{x}_k(t), \mathbf{u}(k), \mathbf{v}(k)) \Big] \in \mathbf{R}^{n_x \times n_x}$$

and with the decomposition of the disturbance matrix $D(t)$,

$$D(t) = \begin{bmatrix} \mathbf{d}_1(t) & \mathbf{d}_2(t) & \dots & \mathbf{d}_{n_v}(t) \end{bmatrix} \in \mathbf{R}^{n_x \times n_v}$$

where $\mathbf{d}_j(t) \in \mathbf{R}^{n_x}$, the lifted dynamics map is

$$\boldsymbol{f}_{\text{big}}(t, \mathbf{x}_{\text{big}}(t), \mathbf{u}(k), \mathbf{v}(k)) = \begin{bmatrix} \boldsymbol{f}(t, \mathbf{x}_k(t), \mathbf{u}(k), \mathbf{v}(k)) \\ A(t)\boldsymbol{\phi}_1(t) \\ \vdots \\ A(t)\boldsymbol{\phi}_{n_x}(t) \\ A(t)\boldsymbol{\gamma}_1(t) + \mathbf{d}_1(t) \\ \vdots \\ A(t)\boldsymbol{\gamma}_{n_v}(t) + \mathbf{d}_{n_v}(t) \end{bmatrix}$$

This gives us an IVP in $\mathbf{x}_{\text{big}}(t)$ that we can numerically integrate to produce $\mathbf{x}_{\text{big}}((k+1)\Delta t)$, which contains all useful information for the discretized nonlinear dynamical system.

8.2 The extended Kalman filter

Consider the nonlinear system defined for $k \in \{0, \dots, N-1\}$,

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{f}(k, \mathbf{x}(k), \mathbf{u}(k), \mathbf{v}(k)) \\ \mathbf{z}(k+1) &= \mathbf{h}(k+1, \mathbf{x}(k+1)) + \mathbf{w}(k+1) \end{aligned} \tag{9}$$

where the disturbance and noise sequences are zero mean and white,

$$\begin{aligned} \mathbf{E}[\mathbf{v}(k)] = 0, \quad \mathbf{E}[\mathbf{v}(k)\mathbf{v}(j)^T] = \delta_{kj} Q(k) \\ \mathbf{E}[\mathbf{w}(k)] = 0, \quad \mathbf{E}[\mathbf{w}(k)\mathbf{w}(j)^T] = \delta_{kj} R(k) \end{aligned}$$

Note that such a system may be defined directly, or may be the result of discretizing a continuous-time nonlinear system.

The **extended Kalman filter** (EKF) is the simplest nonlinear filter for this problem. It relies on a Taylor series expansion of the dynamics and measurement map about the current state estimate. This section presents the first-order EKF algorithm, although a second-order EKF algorithm also exists. The first-order EKF keeps only first-order terms in the Taylor expansion. The second-order EKF keeps terms up to second order.

8.2.1 EKF derivation

While the optimal estimator for the nonlinear dynamical system (9) is likely a nonlinear function of the measurements, we can calculate the LMMSE estimator using the fundamental equations of linear estimation, cf. Section 2.1.

The LMMSE equations are

$$\begin{aligned}\hat{\mathbf{x}}(k+1) &= \bar{\mathbf{x}}(k+1) + P_{xz}(k+1)P_{zz}(k+1)^{-1}[\mathbf{z}(k+1) - \bar{\mathbf{z}}(k+1)] \\ P(k+1) &= \bar{P}(k+1) - \bar{P}_{xz}(k+1)\bar{P}_{zz}(k+1)^{-1}\bar{P}_{xz}(k+1)^T\end{aligned}$$

For a general nonlinear system, the above expectations and MSE matrices may be difficult to compute. In the EKF, we simplify the calculations by Taylor expanding $\mathbf{f}$ and $\mathbf{h}$ and keeping only higher order terms.

If the true state $\mathbf{x}(k)$ and the true disturbance $\mathbf{v}(k)$ are "close" to $\hat{\mathbf{x}}(k)$ and $\mathbf{E}[\mathbf{v}(k)] = \mathbf{0}$ – meaning the MSE matrix $P(k)$ and the disturbance covariance $Q(k)$ are "small" in some matrix sense – then the first-order Taylor expansion of $\mathbf{f}$ about $\hat{\mathbf{x}}(k)$ and $\mathbf{v}(k) = \mathbf{0}$ should be a reasonable approximation of $\mathbf{f}$. Given this approximation, the state prediction is

$$\begin{aligned}\bar{\mathbf{x}}(k+1) &= \mathbf{E}[\mathbf{f}(k, \mathbf{x}(k), \mathbf{u}(k), \mathbf{v}(k))|\mathbf{z}^k] \\ &\approx \mathbf{E}[\mathbf{f}(k, \hat{\mathbf{x}}(k), \mathbf{u}(k), \mathbf{0}) + \left.\frac{\partial f}{\partial \mathbf{x}}\right|_k [\mathbf{x}(k) - \hat{\mathbf{x}}(k)] \\ &\quad + \left.\frac{\partial f}{\partial \mathbf{v}}\right|_k [\mathbf{v}(k) - \mathbf{0}) \ |\mathbf{z}^k] \\ &= \mathbf{f}(k, \hat{\mathbf{x}}(k), \mathbf{u}(k), \mathbf{0})\end{aligned}$$

where the last line follows from the fact that $\mathbf{E}[\mathbf{x}(k)|\mathbf{z}^k] = \hat{\mathbf{x}}(k)$ and $\mathbf{E}[\mathbf{v}(k)|\mathbf{z}^k] = \mathbf{0}$.

A similar calculation gives the prediction MSE

$$\bar{P}(k+1) = F(k)P(k)F(k)^T + \Gamma(k)Q(k)\Gamma(k)^T$$

where we define

$$\begin{aligned}F(k) &= \left.\frac{\partial}{\partial \mathbf{x}(k)}\mathbf{f}(k, \mathbf{x}(k), \mathbf{u}(k), \mathbf{v}(k))\right|_{k, \hat{\mathbf{x}}(k), \mathbf{u}(k), \mathbf{v}(k)=\mathbf{0}} \\ \Gamma(k) &= \left.\frac{\partial}{\partial \mathbf{v}(k)}\mathbf{f}(k, \mathbf{x}(k), \mathbf{u}(k), \mathbf{v}(k))\right|_{k, \hat{\mathbf{x}}(k), \mathbf{u}(k), \mathbf{v}(k)=\mathbf{0}}\end{aligned}$$

The above are the desired formulas for $\bar{\mathbf{x}}(k+1)$ and $\bar{P}(k+1)$. In the LMMSE equations, we also need to compute $\bar{\mathbf{z}}(k+1)$ and the associated MSE matrices.

If $\bar{P}(k+1)$ is "small", then the first-order Taylor expansion of the measurement map about $\bar{\mathbf{x}}(k+1)$ should be a reasonable approximation of $\mathbf{h}$. Given this approximation, the

predicted measurement is

$$\begin{aligned}\bar{\mathbf{z}}(k+1) &= \mathbf{E}[\mathbf{h}(k+1,\mathbf{x}(k+1)) + \mathbf{w}(k+1)|\mathbf{z}^k] \\ &= \mathbf{E}[\mathbf{h}(k+1,\mathbf{x}(k+1))|\mathbf{z}^k] \\ &\approx \mathbf{E}[\mathbf{h}(k+1,\bar{\mathbf{x}}(k+1)) + \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}(k+1)}\right|_{k+1}[\mathbf{x}(k+1) - \bar{\mathbf{x}}(k+1)]|\mathbf{z}^k] \\ &= \mathbf{h}(k+1,\bar{\mathbf{x}}(k+1))\end{aligned}$$

where the last follows from the fact that $\mathbf{E}[\mathbf{x}(k+1)|\mathbf{z}^k] = \bar{\mathbf{x}}(k+1)$.

Similar calculations give the MSE matrices

$$\begin{aligned}\bar{P}_{xz}(k+1) &= \bar{P}(k+1)H(k+1)^T \\ \bar{P}_{zz}(k+1) &= H(k+1)\bar{P}(k+1)H(k+1)^T + R(k+1)\end{aligned}$$

where we define

$$H(k+1) = \left.\frac{\partial}{\partial \mathbf{x}(k+1)}\mathbf{h}(k+1,\mathbf{x}(k+1))\right|_{k+1,\bar{\mathbf{x}}(k+1)}$$

This concludes the derivation of the EKF, interpreted as the LMMSE estimator using first-order Taylor approximations of the dynamics and measurement maps.

8.2.2 EKF algorithm

The EKF can be put into a form that closely resembles the KF algorithm in Section 5. Given the mean $\hat{\mathbf{x}}(0)$ and covariance $P(0)$ of the initial state, the EKF algorithm is

1. set $k = 0$
2. **dynamic propagation**
   (a) predict the next state

$$\bar{\mathbf{x}}(k+1) = \mathbf{f}(k,\hat{\mathbf{x}}(k),\mathbf{u}(k),\mathbf{0})$$

   (b) calculate the MSE of this prediction

$$\bar{P}(k+1) = F(k)P(k)F(k)^T + \Gamma(k)Q(k)\Gamma(k)^T$$

   where

$$\begin{aligned}F(k) &= \left.\frac{\partial}{\partial \mathbf{x}(k)}\mathbf{f}(k,\mathbf{x}(k),\mathbf{u}(k),\mathbf{v}(k))\right|_{k,\hat{\mathbf{x}}(k),\mathbf{u}(k),\mathbf{v}(k)=\mathbf{0}} \\ \Gamma(k) &= \left.\frac{\partial}{\partial \mathbf{v}(k)}\mathbf{f}(k,\mathbf{x}(k),\mathbf{u}(k),\mathbf{v}(k))\right|_{k,\hat{\mathbf{x}}(k),\mathbf{u}(k),\mathbf{v}(k)=\mathbf{0}}\end{aligned}$$

3. **measurement update**

   (a) predict the measurement

   $$\bar{\mathbf{z}}(k+1) = \mathbf{h}(k+1, \bar{\mathbf{x}}(k+1))$$

   (b) calculate the MSE of this prediction

   $$\bar{P}_{zz}(k+1) \equiv S(k+1) = H(k+1)\bar{P}(k+1)H(k+1)^T + R(k+1)$$

   where

   $$H(k+1) = \frac{\partial}{\partial \mathbf{x}(k+1)}\mathbf{h}(k+1, \mathbf{x}(k+1))\bigg|_{k+1, \bar{\mathbf{x}}(k+1)}$$

   (c) measure $\mathbf{z}(k+1)$ and calculate the innovation

   $$\boldsymbol{\nu}(k+1) = \mathbf{z}(k+1) - \bar{\mathbf{z}}(k+1)$$

   (d) calculate the filter gain

   $$W(k+1) = \bar{P}(k+1)H(k+1)^T S(k+1)^{-1}$$

   (e) update the state estimate

   $$\hat{\mathbf{x}}(k+1) = \bar{\mathbf{x}}(k+1) + W(k+1)\boldsymbol{\nu}(k+1)$$

   (f) update the state estimate MSE

   $$P(k+1) = \bar{P}(k+1) - W(k+1)S(k+1)W(k+1)^T$$

4. increment k and return to step 2

8.2.3 Remarks

- Since the estimates $\hat{\mathbf{x}}(k)$ and predictions $\bar{\mathbf{x}}(k+1)$ and $\bar{\mathbf{z}}(k+1)$ depend on linearizations of the dynamics and measurement maps, they will probably be biased. The associated error matrices are therefore MSEs, not covariances.

- In the EKF, the matrices $F(k)$ and $\Gamma(k)$ are calculated at each stage by evaluating the Jacobians of $\mathbf{f}$ with respect to $\mathbf{x}(k)$ and $\mathbf{v}(k)$, respectively, at the state estimate $\hat{\mathbf{x}}(k)$ and the mean disturbance $\mathbf{v}(k) = \mathbf{0}$.

  Similarly, the matrix $H(k+1)$ is calculated by evaluating the Jacobian of $\mathbf{h}$ with respect to $\mathbf{x}(k+1)$ at the predicted state $\bar{\mathbf{x}}(k+1)$.

  Although the Jacobians of $\mathbf{f}$ and $\mathbf{h}$ can be computed symbolically in advance, the state estimates $\hat{\mathbf{x}}(k)$ and predictions $\bar{\mathbf{x}}(k+1)$ are not known in advance. Thus, unlike the covariance calculations in the KF, the MSE calculations in the EKF cannot be computed offline.

- Much like the KF, the EKF can run into problems with numerical stability and with initialization. The **extended square root information filter** (ESRIF) fixes these problems.

8.2.4 ESRIF derivation

Just as the EKF looks a lot like the KF, the ESRIF looks a lot like the SRIF in Chapter 6. It follows the same principles: only admit random variables with zero mean and identity covariance, and only manipulate matrices by orthonormal transformations.

As with the SRIF, the starting point of the ESRIF are the SRIEs

$$\begin{aligned} \mathbf{0} &= \mathcal{R}_{vv}(k)\mathbf{v}(k) + \mathbf{w}_v(k) \\ \boldsymbol{\mathfrak{z}}_x(k) &= \mathcal{R}_{xx}(k)\mathbf{x}(k) + \mathbf{w}_x(k) \end{aligned}$$

where $\mathcal{R}_{vv}(k)^T\mathcal{R}_{vv}(k) = Q(k)^{-1}$ and $\mathcal{R}_{xx}(k)^T\mathcal{R}_{xx}(k) = P(k)^{-1}$.

In the dynamic propagation step, we seek an expression for $\bar{\boldsymbol{\mathfrak{z}}}_x(k+1)$ in terms of $\boldsymbol{\mathfrak{z}}_x(k)$ and $\mathcal{R}_{xx}(k)$.

Suppose there exists an inverse dynamics map $\mathbf{f}^{-1}$ such that

$$\mathbf{f}^{-1}(k, \mathbf{f}(k, \mathbf{x}(k), \mathbf{u}(k), \mathbf{v}(k)), \mathbf{u}(k), \mathbf{v}(k)) = \mathbf{x}(k)$$

Then we can eliminate $\mathbf{x}(k)$ from the SRIEs:

$$\begin{aligned} \mathbf{0} &= \mathcal{R}_{vv}(k)\mathbf{v}(k) + \mathbf{w}_v(k) \\ \boldsymbol{\mathfrak{z}}_x(k) &= \mathcal{R}_{xx}(k)\mathbf{f}^{-1}(k, \mathbf{x}(k+1), \mathbf{u}(k), \mathbf{v}(k)) + \mathbf{w}_x(k) \end{aligned}$$

The linear approximation of $\mathbf{f}^{-1}(k, \mathbf{x}(k+1), \mathbf{u}(k), \mathbf{v}(k))$, for $\mathbf{x}(k+1)$ near $\bar{\mathbf{x}}(k+1)$ and for $\mathbf{v}(k)$ near $\mathbf{0}$, is

$$\begin{aligned} \mathbf{f}^{-1}(k, \mathbf{x}(k+1), \mathbf{u}(k), \mathbf{v}(k)) \approx{}& \mathbf{f}^{-1}(k, \bar{\mathbf{x}}(k+1), \mathbf{u}(k), \mathbf{v}(k)) \\ &+ \left.\frac{\partial \mathbf{f}^{-1}}{\partial \mathbf{x}(k+1)}\right|_k \Big[\mathbf{x}(k+1) - \bar{\mathbf{x}}(k+1)\Big] + \left.\frac{\partial \mathbf{f}^{-1}}{\partial \mathbf{v}(k)}\right|_k \mathbf{v}(k) \end{aligned}$$

where $\mathbf{f}^{-1}(k, \bar{\mathbf{x}}(k+1), \mathbf{u}(k), \mathbf{v}(k)) = \hat{\mathbf{x}}(k)$, by the definition of $\mathbf{f}^{-1}$ and the fact that in the EKF, $\bar{\mathbf{x}}(k+1) = \mathbf{f}(k, \hat{\mathbf{x}}(k), \mathbf{u}(k), \mathbf{v}(k))$.

By applying the chain rule, one can show that

$$\begin{aligned} \frac{\partial \mathbf{f}^{-1}}{\partial \mathbf{x}(k+1)} &= \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}(k)}\right)^{-1} \\ \frac{\partial \mathbf{f}^{-1}}{\partial \mathbf{v}(k)} &= -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}(k)}\right)^{-1} \frac{\partial \mathbf{f}}{\partial \mathbf{v}(k)} \end{aligned}$$

so, using the definitions of $F(k)$ and $\Gamma(k)$ from the EKF derivation, the linearized inverse dynamics are

$$\mathbf{f}^{-1}(k, \mathbf{x}(k+1), \mathbf{u}(k), \mathbf{v}(k)) \approx \hat{\mathbf{x}}(k) + F(k)^{-1}\Big[\mathbf{x}(k+1) - \bar{\mathbf{x}}(k+1)\Big] - F(k)^{-1}\Gamma(k)\mathbf{v}(k)$$

The linearized SRIEs are therefore

$$\begin{aligned}\mathbf{0} &= \mathcal{R}_{vv}(k)\mathbf{v}(k) + \mathbf{w}_v(k)\\ \mathfrak{z}_x(k) &= \mathcal{R}_{xx}(k)\Big\{\hat{\mathbf{x}}(k) + F(k)^{-1}\big[\mathbf{x}(k+1) - \bar{\mathbf{x}}(k+1)\big] - F(k)^{-1}\Gamma(k)\mathbf{v}(k)\Big\} + \mathbf{w}_x(k)\end{aligned}$$

Noting that $\mathcal{R}_{xx}(k)\hat{\mathbf{x}}(k) = \mathfrak{z}_x(k+1)$, the linearized SRIEs in matrix form are

$$\begin{bmatrix}\mathbf{0}\\ \mathcal{R}_{xx}(k)F(k)^{-1}\mathbf{f}(k, \mathcal{R}_{xx}(k)^{-1}\mathfrak{z}_x(k), \mathbf{u}(k), \mathbf{0})\end{bmatrix} = \begin{bmatrix}\mathcal{R}_{vv}(k) & 0\\ -\mathcal{R}_{xx}(k)F(k)^{-1}\Gamma(k) & \mathcal{R}_{xx}(k)F(k)^{-1}\end{bmatrix} * \begin{bmatrix}\mathbf{v}(k)\\ \mathbf{x}(k+1)\end{bmatrix} + \begin{bmatrix}\mathbf{w}_v(k)\\ \mathbf{w}_x(k)\end{bmatrix}$$

To avoid inverting the coefficient matrix, we QR-factorize it, i.e. we find an orthonormal matrix $T_a(k)^T \in \mathbf{R}^{(n_x+n_v)\times(n_x+n_v)}$ and upper triangular matrices $\bar{\mathcal{R}}_{vv}(k) \in \mathbf{R}^{n_v\times n_v}$ and $\bar{\mathcal{R}}_{xx}(k+1) \in \mathbf{R}^{n_x\times n_x}$ such that

$$\begin{bmatrix}\mathcal{R}_{vv}(k) & 0\\ -\mathcal{R}_{xx}(k)F(k)^{-1}\Gamma(k) & \mathcal{R}_{xx}(k)F(k)^{-1}\end{bmatrix} = T_a(k)^T\begin{bmatrix}\bar{\mathcal{R}}_{vv}(k) & \bar{\mathcal{R}}_{vx}(k+1)\\ 0 & \bar{\mathcal{R}}_{xx}(k+1)\end{bmatrix}$$

Multiplying both sides of the linearized SRIE system by $T_a(k)$ gives

$$\begin{bmatrix}\bar{\mathfrak{z}}_v(k)\\ \bar{\mathfrak{z}}_x(k+1)\end{bmatrix} = \begin{bmatrix}\bar{\mathcal{R}}_{vv}(k) & \bar{\mathcal{R}}_{vx}(k+1)\\ 0 & \bar{\mathcal{R}}_{xx}(k+1)\end{bmatrix}\begin{bmatrix}\mathbf{v}(k)\\ \mathbf{x}(k+1)\end{bmatrix} + \begin{bmatrix}\bar{\mathbf{w}}_v(k)\\ \bar{\mathbf{w}}_x(k+1)\end{bmatrix}$$

where

$$\begin{bmatrix}\bar{\mathfrak{z}}_v(k)\\ \bar{\mathfrak{z}}_x(k+1)\end{bmatrix} = T_a(k)\begin{bmatrix}\mathbf{0}\\ \mathcal{R}_{xx}(k)F(k)^{-1}\mathbf{f}(k, \mathcal{R}_{xx}(k)^{-1}\mathfrak{z}_x(k), \mathbf{u}(k), \mathbf{0})\end{bmatrix}$$
$$\begin{bmatrix}\bar{\mathbf{w}}_v(k)\\ \bar{\mathbf{w}}_x(k+1)\end{bmatrix} = T_a(k)\begin{bmatrix}\mathbf{w}_v(k)\\ \mathbf{w}_x(k)\end{bmatrix}$$

This completes the dynamic propagation step: we have the prediction $\bar{\mathfrak{z}}_x(k+1)$ in terms of $\mathfrak{z}_x(k)$ and $\mathcal{R}_{xx}(k)$. We also have $\bar{\mathfrak{z}}_v(k)$ for later use in smoothing.

In the measurement update step, we seek expressions for $\mathfrak{z}_x(k+1)$ and $\mathcal{R}_{xx}(k+1)$ in terms of $\mathfrak{z}_x(k)$ and intermediate matrices from the dynamic propagation step.

We'll use the Cholesky factorizations of the noise covariance matrices,

$$R_a(k+1)^T R_a(k+1) = R(k+1)$$

where $R_a(k+1) \in \mathbf{R}^{n_z\times n_z}$ is upper triangular and non-singular.

Under the transformations

$$\begin{aligned}\mathbf{z}_a(k+1) &= R_a(k+1)^{-T}\mathbf{z}(k+1) \in \mathbf{R}^{n_z}\\ \mathbf{w}_a(k+1) &= R_a(k+1)^{-T}\mathbf{w}(k+1) \in \mathbf{R}^{n_z}\\ \mathbf{h}_a(k+1, \mathbf{x}(k+1)) &= R_a(k+1)^{-T}\mathbf{h}(k+1, \mathbf{x}(k+1)) \in \mathbf{R}^{n_z\times n_x}\end{aligned}$$

the measurement model becomes

$$\mathbf{z}_a(k+1) = \mathbf{h}_a(k+1, \mathbf{x}(k+1)) + \mathbf{w}_a(k+1)$$

In the neighborhood of $\bar{\mathbf{x}}(k+1)$, the linearized measurement map is

$$\begin{aligned}\mathbf{h}_a(k+1, \mathbf{x}(k+1)) &\approx \mathbf{h}_a(k+1, \bar{\mathbf{x}}(k+1)) + \left.\frac{\partial \mathbf{h}_a}{\partial \mathbf{x}(k+1)}\right|_{k+1, \bar{\mathbf{x}}(k+1)} \big[\mathbf{x}(k+1) - \bar{\mathbf{x}}(k+1)\big] \\ &= \mathbf{h}_a(k+1, \bar{\mathbf{x}}(k+1)) + R_a(k+1)^{-T} H(k+1) \big[\mathbf{x}(k+1) - \bar{\mathbf{x}}(k+1)\big]\end{aligned}$$

where we used the EKF definition of $H(k+1)$.

Combining the state SRIE and the linearized measurement equation gives the system

$$\begin{bmatrix} \bar{\boldsymbol{\mathfrak{z}}}_x(k+1) \\ \mathbf{z}_a(k+1) - \mathbf{h}_a(k+1, \bar{\mathbf{x}}(k+1)) + H_a(k+1)\bar{\mathbf{x}}(k+1) \end{bmatrix} = \begin{bmatrix} \bar{\mathcal{R}}_{xx}(k+1) \\ H_a(k+1) \end{bmatrix} \mathbf{x}(k+1) + \begin{bmatrix} \bar{\mathbf{w}}_x(k+1) \\ \mathbf{w}_a(k+1) \end{bmatrix}$$

where we can calculate $\bar{\mathbf{x}}(k+1)$ by propagating the transformed estimate through the dynamics map, assuming no noise:

$$\bar{\mathbf{x}}(k+1) = \mathbf{f}(k, \mathcal{R}_{xx}(k+1)^{-1} \boldsymbol{\mathfrak{z}}_x(k), \mathbf{u}(k), \mathbf{0})$$

To avoid inverting the coefficient matrix, we can QR factorize it by finding an orthonormal $T_b(k+1) \in \mathbf{R}^{(n_x+n_v)\times(n_x+n_v)}$ and upper triangular, non-singular $\mathcal{R}_{xx}(k+1) \in \mathbf{R}^{n_x \times n_x}$ such that

$$\begin{bmatrix} \bar{\mathcal{R}}_{xx}(k+1) \\ H_a(k+1) \end{bmatrix} = T_b(k+1)^T \begin{bmatrix} \mathcal{R}_{xx}(k+1) \\ 0 \end{bmatrix}$$

The SRIE-measurement system then becomes

$$\begin{bmatrix} \boldsymbol{\mathfrak{z}}_x(k+1) \\ \boldsymbol{\mathfrak{z}}_r(k+1) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_{xx}(k+1) \\ 0 \end{bmatrix} \mathbf{x}(k+1) + \begin{bmatrix} \mathbf{w}_x(k+1) \\ \mathbf{w}_r(k+1) \end{bmatrix}$$

where

$$\begin{aligned}\begin{bmatrix} \boldsymbol{\mathfrak{z}}_x(k+1) \\ \boldsymbol{\mathfrak{z}}_r(k+1) \end{bmatrix} &= T_b(k+1) \begin{bmatrix} \bar{\boldsymbol{\mathfrak{z}}}_x(k+1) \\ \mathbf{z}_a(k+1) - \mathbf{h}_a(k+1, \bar{\mathbf{x}}(k+1)) + H_a(k+1)\bar{\mathbf{x}}(k+1) \end{bmatrix} \\ \begin{bmatrix} \mathbf{w}_x(k+1) \\ \mathbf{w}_r(k+1) \end{bmatrix} &= T_b(k+1) \begin{bmatrix} \bar{\mathbf{w}}_x(k+1) \\ \mathbf{w}_a(k+1) \end{bmatrix}\end{aligned}$$

8.2.5 ESRIF algorithm

Given $\mathcal{R}_{xx}(0)^T \mathcal{R}_{xx}(0) = \mathcal{I}(0)$ and $\boldsymbol{\mathfrak{z}}_x(0) = \mathcal{R}_{xx}(0)\hat{\mathbf{x}}(0)$, the ESRIF algorithm is:

1. **offline calculations**

   (a) derive expressions for the Jacobians $\frac{\partial \mathbf{f}}{\partial \mathbf{x}(k)}$, $\frac{\partial \mathbf{f}}{\partial \mathbf{v}(k)}$, and $\frac{\partial \mathbf{h}}{\partial \mathbf{x}(k+1)}$

(b) disturbance Cholesky factorization: for each k, find an upper triangular $\mathcal{R}_{vv}(k) \in \mathbf{R}^{n_v \times n_v}$ such that

$$\mathcal{R}_{vv}(k)^T \mathcal{R}_{vv}(k) = Q(k)^{-1}$$

(c) noise Cholesky factorization: for each k, find an upper triangular $R_a(k) \in \mathbf{R}^{n_z \times n_z}$ such that

$$R_a(k)^T R_a(k) = R(k)$$

2. set $k = 0$

3. **dynamic propagation**

(a) evaluate the Jacobians

$$F(k) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}(k)}\right|_{k,\hat{\mathbf{x}}(k),\mathbf{u}(k),\mathbf{v}(k)=\mathbf{0})}$$
$$\Gamma(k) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{v}(k)}\right|_{k,\hat{\mathbf{x}}(k),\mathbf{u}(k),\mathbf{v}(k)=\mathbf{0})}$$

where $\hat{\mathbf{x}}(k) = \mathcal{R}_{xx}(k)^{-1}\boldsymbol{\mathfrak{z}}_x(k)$

(b) QR factorization: find an orthonormal matrix $T_a(k)^T \in \mathbf{R}^{(n_x+n_v)\times(n_x+n_v)}$ and upper triangular matrices $\bar{\mathcal{R}}_{vv}(k) \in \mathbf{R}^{n_v \times n_v}$ and $\bar{\mathcal{R}}_{xx}(k+1) \in \mathbf{R}^{n_x \times n_x}$ such that

$$\begin{bmatrix} \mathcal{R}_{vv}(k) & 0 \\ -\mathcal{R}_{xx}(k)F(k)^{-1}\Gamma(k) & \mathcal{R}_{xx}(k)F(k)^{-1} \end{bmatrix} = T_a(k)^T \begin{bmatrix} \bar{\mathcal{R}}_{vv}(k) & \bar{\mathcal{R}}_{vx}(k+1) \\ 0 & \bar{\mathcal{R}}_{xx}(k+1) \end{bmatrix}$$

(c) predict the (transformed) state

$$\begin{bmatrix} \bar{\boldsymbol{\mathfrak{z}}}_v(k) \\ \bar{\boldsymbol{\mathfrak{z}}}_x(k+1) \end{bmatrix} = T_a(k) \begin{bmatrix} \mathbf{0} \\ \mathcal{R}_{xx}(k)F(k)^{-1}\mathbf{f}(k,\hat{\mathbf{x}}(k),\mathbf{u}(k),\mathbf{0}) \end{bmatrix}$$

4. **measurement update**

(a) evaluate the (transformed) Jacobian

$$H_a(k+1) = R_a(k+1)^{-T} \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}(k+1)}\right|_{k+1,\bar{\mathbf{x}}(k+1)}$$

where $\bar{\mathbf{x}}(k+1) = \mathbf{f}(k, \mathcal{R}_{xx}(k)^{-1}\boldsymbol{\mathfrak{z}}_x(k), \mathbf{u}(k), \mathbf{0})$

(b) QR factorization: find an orthonormal $T_b(k+1) \in \mathbf{R}^{(n_x+n_v)\times(n_x+n_v)}$ and upper triangular, non-singular $\mathcal{R}_{xx}(k+1) \in \mathbf{R}^{n_x \times n_x}$ such that

$$\begin{bmatrix} \bar{\mathcal{R}}_{xx}(k+1) \\ H_a(k+1) \end{bmatrix} = T_b(k+1)^T \begin{bmatrix} \mathcal{R}_{xx}(k+1) \\ 0 \end{bmatrix}$$

(c) compute the (transformed) state estimate

$$\begin{bmatrix} \boldsymbol{\mathfrak{z}}_x(k+1) \\ \boldsymbol{\mathfrak{z}}_r(k+1) \end{bmatrix} = T_b(k+1) \begin{bmatrix} \bar{\boldsymbol{\mathfrak{z}}}_x(k+1) \\ \mathbf{z}_a(k+1) - \mathbf{h}_a(k+1, \bar{\mathbf{x}}(k+1)) + H_a(k+1)\bar{\mathbf{x}}(k+1) \end{bmatrix}$$

5. **inverse transformation**

   (a) recover the state estimate

$$\hat{\mathbf{x}}(k+1) = \mathcal{R}_{xx}(k+1)^{-1} \boldsymbol{\mathfrak{z}}_x(k+1)$$

   (b) recover the state estimation error covariance

$$P(k+1) = \mathcal{R}_{xx}(k+1)^{-1} \mathcal{R}_{xx}(k+1)^{-T}$$

6. increment k and return to step 3

8.2.6 Alternate interpretation of the EKF

In Section 8.2.1, we interpreted the EKF as the LMMSE state estimator using first-order Taylor approximations of the dynamics and measurement maps. An alternate interpretation of the EKF is to consider each iteration as a single Gauss-Newton step in the direction of the MAP state estimate.

Suppose that we can reasonably model the state of system (9) as Gaussian. This is a big supposition. Even if all random variables injected into the system ($\mathbf{x}(0)$, $\mathbf{v}(0)$, $\mathbf{w}(0)$, $\mathbf{v}(1)$, $\mathbf{w}(1)$, ...) are Gaussian, the state of the system for $k \geq 1$ will almost certainly *not* be Gaussian. This is because the nonlinear transformation $\mathbf{f}$ does not, in general, preserve normality.

Still, many non-Gaussian distributions look approximately Gaussian. If the Gaussian assumption on $\mathbf{x}(k+1)$ is reasonable, then the cost J_a in Section 5.6 (the MAP derivation of the KF) approximates the negative natural logarithm of the posterior *pdf* of $\mathbf{x}(k+1)$. This suggests that we can compute a suboptimal *a priori* estimate of $\mathbf{x}(k+1)$ by minimizing J_a.

In the MAP KF derivation, we eliminated $\mathbf{x}(k)$ from J_a by using the inverse dynamics. Suppose that in the nonlinear system (9), the dynamics map $\mathbf{f}$ has an inverse, in the sense that

$$\begin{aligned} & \mathbf{x}(k) = \mathbf{f}^{-1}(k, \mathbf{x}(k+1), \mathbf{u}(k), \mathbf{v}(k)) \\ \iff \quad & \mathbf{x}(k+1) = \mathbf{f}(k, \mathbf{f}^{-1}(k, \mathbf{x}(k+1), \mathbf{u}(k), \mathbf{v}(k)), \mathbf{u}(k), \mathbf{v}(k)) \end{aligned}$$

If such an inverse exists, then we can use it to eliminate $\mathbf{x}(k)$ from J_a. This lets us define the cost

$$J_b(\mathbf{v}(k), \mathbf{x}(k+1), k) = J_a(\mathbf{f}^{-1}(k, \mathbf{x}(k+1), \mathbf{u}(k), \mathbf{v}(k)), \mathbf{v}(k), \mathbf{x}(k+1), k)$$

If we examine the structure of J_b, we see that it's a weighted least squares cost function of the nonlinear algebraic equations

$$\begin{aligned} \mathbf{f}^{-1}(k, \mathbf{x}(k+1), \mathbf{u}(k), \mathbf{v}(k)) - \hat{\mathbf{x}}(k) &= \mathbf{0} \qquad \text{weighted by } P(k)^{-1} \\ \mathbf{v}(k) &= \mathbf{0} \qquad \text{weighted by } Q(k)^{-1} \\ \mathbf{z}(k+1) - \mathbf{h}(k+1, \mathbf{x}(k+1)) &= \mathbf{0} \qquad \text{weighted by } R(k+1)^{-1} \end{aligned}$$

In section 3.1.2, we developed the Gauss-Newton method for solving such nonlinear least squares problems. The Gauss-Newton method begins by linearizing the system about an initial guess, in this case $\mathbf{x}(k+1) = \bar{\mathbf{x}}(k+1)$ and $\mathbf{v}(k) = \mathbf{0}$. The linearized system is

$$\begin{aligned} \mathbf{f}^{-1}(k, \bar{\mathbf{x}}(k+1), \mathbf{u}(k), \mathbf{0}) + \left.\frac{\partial \mathbf{f}^{-1}}{\partial \mathbf{x}(k+1)}\right|_{k, \bar{\mathbf{x}}(k+1), \mathbf{u}(k), \mathbf{v}(k)=\mathbf{0}} \big[\mathbf{x}(k+1) - \bar{\mathbf{x}}(k+1)\big] \\ + \left.\frac{\partial \mathbf{f}^{-1}}{\partial \mathbf{v}(k)}\right|_{k, \bar{\mathbf{x}}(k+1), \mathbf{u}(k), \mathbf{v}(k)=\mathbf{0}} \big[\mathbf{v}(k) - \mathbf{0}\big] - \hat{\mathbf{x}}(k) = \mathbf{0} \\ \mathbf{v}(k) = \mathbf{0} \\ \mathbf{z}(k+1) - \mathbf{h}(k+1, \mathbf{x}(k+1)) - \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}(k+1)}\right|_{k+1, \bar{\mathbf{x}}(k+1)} \big[\mathbf{x}(k+1) - \bar{\mathbf{x}}(k+1)\big] = \mathbf{0} \end{aligned}$$

Noting that

$$\mathbf{f}^{-1}(k, \bar{\mathbf{x}}(k+1), \mathbf{u}(k), \mathbf{0}) = \hat{\mathbf{x}}(k)$$

and defining

$$\begin{aligned} F(k)^{-1} &\equiv \left.\frac{\partial \mathbf{f}^{-1}}{\partial \mathbf{x}(k+1)}\right|_{k, \bar{\mathbf{x}}(k+1), \mathbf{u}(k), \mathbf{v}(k)=\mathbf{0}} \\ -F(k)^{-1}\Gamma(k) &\equiv \left.\frac{\partial \mathbf{f}^{-1}}{\partial \mathbf{v}(k)}\right|_{k, \bar{\mathbf{x}}(k+1), \mathbf{u}(k), \mathbf{v}(k)=\mathbf{0}} \\ H(k+1) &\equiv \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}(k+1)}\right|_{k+1, \bar{\mathbf{x}}(k+1)} \\ \bar{\mathbf{z}}(k+1) &\equiv \mathbf{h}(k+1, \bar{\mathbf{x}}(k+1)) \end{aligned}$$

the linearized system becomes

$$\begin{aligned} F(k)^{-1}\big[\mathbf{x}(k+1) - \bar{\mathbf{x}}(k+1)\big] - F(k)^{-1}\Gamma(k)\mathbf{v}(k) &= \mathbf{0} \qquad \text{weighted by } P(k)^{-1} \\ \mathbf{v}(k) &= \mathbf{0} \qquad \text{weighted by } Q(k)^{-1} \\ \mathbf{z}(k+1) - \bar{\mathbf{z}}(k+1) - H(k+1)\big[\mathbf{x}(k+1) - \bar{\mathbf{x}}(k+1)\big] &= \mathbf{0} \qquad \text{weighted by } R(k+1)^{-1} \end{aligned}$$

With these definitions, the weighted nonlinear least squares cost J_b takes on the exact same form as in the MAP KF derivation, so minimizing it produces the EKF analogs of the KF equations.

The punchline: if the state of (9) is approximately Gaussian distributed, then an iteration of the EKF can be interpreted as a single Gauss-Newton step in the direction of the MAP state estimate, starting from the initial guess

$$\begin{bmatrix} \mathbf{x}(k+1) \\ \mathbf{v}(k) \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{x}}(k+1) \\ \mathbf{0} \end{bmatrix}$$

This motivates our next nonlinear filtering algorithm, the iterated EKF, which takes multiple Gauss-Newton steps during each measurement update iteration.

8.3 The iterated EKF

The **iterated, extended Kalman filter** (IEKF) improves upon the accuracy of the EKF by taking multiple Gauss-Newton steps during each filter iteration. This comes at a computational cost, of course. There are two common versions of the IEKF: the classical IEKF and the backward smoothing IEKF. The former is only modestly slower than the EKF, but the latter can be hundreds of times slower.

In the MAP KF derivation, we took the negative logarithm of the *a priori pdf*, used the inverse dynamics to eliminate $\mathbf{x}(k)$, and then minimized over $\mathbf{v}(k)$. The resulting cost function was

$$\begin{aligned} J_c(\mathbf{x}(k+1), k+1) = &\frac{1}{2}\big[\mathbf{x}(k+1) - \bar{\mathbf{x}}(k+1)\big]^T \bar{P}(k+1)^{-1}\big[\mathbf{x}(k+1) - \bar{\mathbf{x}}(k+1)\big] \\ &+\frac{1}{2}\big[\mathbf{z}(k+1) - \mathbf{h}(k+1, \mathbf{x}(k+1))\big]^T R(k+1)^{-1}\big[\mathbf{z}(k+1) - \mathbf{h}(k+1, \mathbf{x}(k+1))\big] \end{aligned}$$

In both versions of the IEKF, we perform the dynamic propagation step as in the EKF, but in the measurement update step we take additional Gauss-Newton steps until we converge to a local minimum of J_c.

8.3.1 The classical IEKF

Let $\hat{\mathbf{x}}^i(k+1)$ be the estimate of $\mathbf{x}(k+1)$ during the i^{th} Gauss-Newton iteration in the measurement update step, initialized by $\hat{\mathbf{x}}^0(k+1) = \bar{\mathbf{x}}(k+1)$. Letting

$$H^i(k+1) = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}(k+1)}\right|_{k+1, \hat{\mathbf{x}}^i(k+1)}$$

the measurement map, linearized about the i^{th} estimate, is

$$\mathbf{h}(k+1, \mathbf{x}(k+1)) \approx \mathbf{h}(k+1, \hat{\mathbf{x}}^i(k+1)) + H^i(k+1)\big[\mathbf{x}(k+1) - \hat{\mathbf{x}}^i(k+1)\big] = \mathbf{0}$$

Plugging this into J_c gives the linearized cost at the i^{th} Gauss-Newton iteration,

$$\begin{aligned} J_c(\hat{\mathbf{x}}^i(k+1), k+1) = &\frac{1}{2}\big[\hat{\mathbf{x}}^i(k+1) - \bar{\mathbf{x}}(k+1)\big]^T \bar{P}(k+1)^{-1}\big[\hat{\mathbf{x}}^i(k+1) - \bar{\mathbf{x}}(k+1)\big] \\ &+\frac{1}{2}\Big\{\mathbf{z}(k+1) - \mathbf{h}(k+1, \hat{\mathbf{x}}^i(k+1)) - H^i(k+1)\big[\mathbf{x}(k+1) - \hat{\mathbf{x}}^i(k+1)\big]\Big\}^T \\ &*R(k+1)^{-1}\Big\{\mathbf{z}(k+1) - \mathbf{h}(k+1, \hat{\mathbf{x}}^i(k+1)) - H^i(k+1)\big[\mathbf{x}(k+1) - \hat{\mathbf{x}}^i(k+1)\big]\Big\} \end{aligned}$$

Applying the FOC to J_c gives the update equation for the Gauss-Newton guess,

$$\begin{aligned}\hat{\mathbf{x}}^{i+1}(k+1) = \hat{\mathbf{x}}^i(k+1) \\ &+ \alpha P^i(k+1) H^i(k+1)^T R(k+1)^{-1} \big[\mathbf{z}(k+1) - \mathbf{h}(k+1, \hat{\mathbf{x}}^i(k+1))\big] \\ &+ \alpha P^i(k+1) \bar{P}(k+1)^{-1} \big[\bar{\mathbf{x}}(k+1) - \hat{\mathbf{x}}^i(k+1)\big]\end{aligned}$$

where

$$P^i(k+1) = \big[\bar{P}(k+1)^{-1} + H^i(k+1)^T R(k+1)^{-1} H^i(k+1)\big]^{-1}$$

We can iterate these equations until the Gauss-Newton algorithm converges, i.e. until $J_c(\hat{\mathbf{x}}^i(k+1), k+1) = J_c(\hat{\mathbf{x}}^{i-1}(k+1), k+1)$ to within some tolerance.

Given the mean $\hat{\mathbf{x}}(0)$ and covariance $P(0)$ of the initial state, and for Gauss-Newton parameters $\alpha_{\min}$ and $i_{\max}$, the classical IEKF algorithm is

1. set $k = 0$
2. **dynamic propagation**
   (a) predict the next state
   $$\bar{\mathbf{x}}(k+1) = \mathbf{f}(k, \hat{\mathbf{x}}(k), \mathbf{u}(k), \mathbf{0})$$
   (b) calculate the MSE of this prediction
   $$\bar{P}(k+1) = F(k) P(k) F(k)^T + \Gamma(k) Q(k) \Gamma(k)^T$$
   where
   $$F(k) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}(k)}\right|_{k, \hat{\mathbf{x}}(k), \mathbf{u}(k), \mathbf{v}(k) = \mathbf{0}}$$
   $$\Gamma(k) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{v}(k)}\right|_{k, \hat{\mathbf{x}}(k), \mathbf{u}(k), \mathbf{v}(k) = \mathbf{0}}$$
3. **measurement update**
   (a) measure $\mathbf{z}(k+1)$
   (b) apply the Gauss-Newton method:
      i. set $i = 0$ and initialize $\hat{\mathbf{x}}^i(k+1) = \bar{\mathbf{x}}(k+1)$
      ii. evaluate the Jacobian iterate
      $$H^i(k+1) = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}(k+1)}\right|_{k+1, \hat{\mathbf{x}}^i(k+1)}$$
      iii. compute the MSE iterate
      $$P^i(k+1) = \big[\bar{P}(k+1)^{-1} + H^i(k+1)^T R(k+1)^{-1} H^i(k+1)\big]^{-1}$$

iv. set $\alpha = 1$

v. if $\alpha \leq \alpha_{\min}$, go to 3c

vi. compute the state estimation iterate

$$\begin{aligned}\hat{\mathbf{x}}^{i+1}(k+1) = \hat{\mathbf{x}}^{i}(k+1) \\ + \alpha P^{i}(k+1) H^{i}(k+1)^{T} R(k+1)^{-1} \left[\mathbf{z}(k+1) - \mathbf{h}(k+1, \hat{\mathbf{x}}^{i}(k+1))\right] \\ + \alpha P^{i}(k+1) \bar{P}(k+1)^{-1} \left[\bar{\mathbf{x}}(k+1) - \hat{\mathbf{x}}^{i}(k+1)\right]\end{aligned}$$

vii. if $J_c(\hat{\mathbf{x}}^{i+1}(k+1), k+1) > J_c(\hat{\mathbf{x}}^{i}(k+1), k+1)$, halve α and go to step 3(b)vi

viii. if $i = i_{\max}$, go to 3c; otherwise, increment i and go to step 3(b)ii

(c) update the state estimate

$$\hat{\mathbf{x}}(k+1) = \hat{\mathbf{x}}^{i}(k+1)$$

(d) update the state estimation MSE

$$P(k+1) = P^{i}(k+1)$$

4. increment k and return to step 2

8.3.2 The IESRIF

The IEKF can run into the initialization and numerical problems common to the Kalman filter. As usual, our solution to these problems is to work with information matrices and to use square-root methods. The version of the IEKF that this produces is called the **iterated, extended square-root information filter** (IESRIF).

The IESRIF uses the same dynamic propagation step as the ESRIF in Section 8.2.4, but modifies the measurement update to incorporate multiple Gauss-Newton iterations.

With the definitions

$$\begin{aligned}\bar{\mathcal{R}}_{xx}(k+1)^{T} \bar{\mathcal{R}}_{xx}(k+1) &= \bar{P}(k+1) \\ \bar{\boldsymbol{\jmath}}_{x}(k+1) &= \bar{\mathcal{R}}_{xx}(k+1) \bar{\mathbf{x}}(k+1) \\ R_a(k+1)^{-T} R_a(k+1)^{-1} &= R(k+1)^{-1} \\ \mathbf{z}_a(k+1) &= R_a(k+1)^{-T} \mathbf{z}(k+1) \\ \mathbf{h}_a(k+1, \mathbf{x}(k+1)) &= R_a(k+1)^{-T} \mathbf{h}(k+1, \mathbf{x}(k+1))\end{aligned}$$

the nonlinear least squares cost J_c becomes

$$\begin{aligned}J_c(\mathbf{x}(k+1), k+1) = \frac{1}{2} \left[\bar{\mathcal{R}}_{xx}(k+1)\mathbf{x}(k+1) - \bar{\boldsymbol{\jmath}}_{x}(k+1)\right]^{T} \left[\bar{\mathcal{R}}_{xx}(k+1)\mathbf{x}(k+1) - \bar{\boldsymbol{\jmath}}_{x}(k+1)\right] \\ + \frac{1}{2} \left[\mathbf{z}_a(k+1) - \mathbf{h}_a(k+1, \mathbf{x}(k+1))\right]^{T} \left[\mathbf{z}_a(k+1) - \mathbf{h}_a(k+1, \mathbf{x}(k+1))\right]\end{aligned}$$

Given $\mathcal{R}_{xx}(0)^{T} \mathcal{R}_{xx}(0) = \mathcal{I}(0)$, $\boldsymbol{\jmath}_{x}(0) = \mathcal{R}_{xx}(0)\hat{\mathbf{x}}(0)$, and the Gauss-Newton parameters $i_{\max}$ and $\alpha_{\min}$, the IESRIF algorithm is:

1. **offline calculations**
   (a) derive expressions for the Jacobians $\frac{\partial \mathbf{f}}{\partial \mathbf{x}(k)}$, $\frac{\partial \mathbf{f}}{\partial \mathbf{v}(k)}$, and $\frac{\partial \mathbf{h}}{\partial \mathbf{x}(k+1)}$
   (b) disturbance Cholesky factorization: for each k, find an upper triangular $\mathcal{R}_{vv}(k) \in \mathbf{R}^{n_v \times n_v}$ such that
   $$\mathcal{R}_{vv}(k)^T \mathcal{R}_{vv}(k) = Q(k)^{-1}$$
   (c) noise Cholesky factorization: for each k, find an upper triangular $R_a(k) \in \mathbf{R}^{n_z \times n_z}$ such that
   $$R_a(k)^T R_a(k) = R(k)$$
2. set $k = 0$
3. **dynamic propagation**
   (a) evaluate the Jacobians
   $$F(k) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}(k)}\right|_{k, \hat{\mathbf{x}}(k), \mathbf{u}(k), \mathbf{v}(k)=\mathbf{0})}$$
   $$\Gamma(k) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{v}(k)}\right|_{k, \hat{\mathbf{x}}(k), \mathbf{u}(k), \mathbf{v}(k)=\mathbf{0})}$$
   where $\hat{\mathbf{x}}(k) = \mathcal{R}_{xx}(k)^{-1} \boldsymbol{\mathfrak{z}}_x(k)$
   (b) QR factorization: find an orthonormal matrix $T_a(k)^T \in \mathbf{R}^{(n_x+n_v)\times(n_x+n_v)}$ and upper triangular matrices $\bar{\mathcal{R}}_{vv}(k) \in \mathbf{R}^{n_v \times n_v}$ and $\bar{\mathcal{R}}_{xx}(k+1) \in \mathbf{R}^{n_x \times n_x}$ such that
   $$\begin{bmatrix} \mathcal{R}_{vv}(k) & 0 \\ -\mathcal{R}_{xx}(k) F(k)^{-1} \Gamma(k) & \mathcal{R}_{xx}(k) F(k)^{-1} \end{bmatrix} = T_a(k)^T \begin{bmatrix} \bar{\mathcal{R}}_{vv}(k) & \bar{\mathcal{R}}_{vx}(k+1) \\ 0 & \bar{\mathcal{R}}_{xx}(k+1) \end{bmatrix}$$
   (c) predict the (transformed) state
   $$\begin{bmatrix} \bar{\boldsymbol{\mathfrak{z}}}_v(k) \\ \bar{\boldsymbol{\mathfrak{z}}}_x(k+1) \end{bmatrix} = T_a(k) \begin{bmatrix} \mathbf{0} \\ \mathcal{R}_{xx}(k) F(k)^{-1} \mathbf{f}(k, \mathcal{R}_{xx}(k)^{-1} \boldsymbol{\mathfrak{z}}_x(k), \mathbf{u}(k), \mathbf{0}) \end{bmatrix}$$
4. **measurement update**
   (a) measure $\mathbf{z}(k+1)$ and compute $\mathbf{z}_a(k+1) = R_a(k+1)^{-T} \mathbf{z}(k+1)$
   (b) apply the Gauss-Newton method:
      i. set $i = 1$ and initialize $\hat{\mathbf{x}}^0(k+1) = \bar{\mathbf{x}}(k+1) = \bar{\mathcal{R}}_{xx}(k+1)^{-1} \bar{\boldsymbol{\mathfrak{z}}}_x(k+1)$
      ii. evaluate the (transformed) Jacobian iterate,
      $$H_a^i(k+1) = R_a(k+1)^{-T} \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}(k+1)}\right|_{k+1, \hat{\mathbf{x}}^{i-1}(k+1)}$$

iii. QR factorization: find an orthonormal $T_b^i(k+1) \in \mathbf{R}^{(n_x+n_v)\times(n_x+n_v)}$ and upper triangular, non-singular $\mathcal{R}_{xx}^i(k+1) \in \mathbf{R}^{n_x \times n_x}$ such that

$$\begin{bmatrix} \bar{\mathcal{R}}_{xx}(k+1) \\ H_a^i(k+1) \end{bmatrix} = T_b^i(k+1)^T \begin{bmatrix} \mathcal{R}_{xx}^i(k+1) \\ 0 \end{bmatrix}$$

iv. compute the (transformed) state estimate

$$\begin{bmatrix} \boldsymbol{\mathfrak{z}}_x^i(k+1) \\ \boldsymbol{\mathfrak{z}}_r^i(k+1) \end{bmatrix} = T_b^i(k+1) \begin{bmatrix} \bar{\boldsymbol{\mathfrak{z}}}_x(k+1) \\ \mathbf{z}_a(k+1) - \mathbf{h}_a(k+1, \hat{\mathbf{x}}^i(k+1)) + H_a^i(k+1)\hat{\mathbf{x}}^i(k+1) \end{bmatrix}$$

v. set $\alpha = 1$

vi. if $\alpha \leq \alpha_{\min}$, go to step 4c

vii. compute the candidate state estimation iterate

$$\hat{\mathbf{x}}^i(k+1) = \hat{\mathbf{x}}^{i-1}(k+1) + \alpha\big[\mathcal{R}_{xx}^i(k+1)^{-1}\boldsymbol{\mathfrak{z}}_x^i(k+1) - \hat{\mathbf{x}}^{i-1}(k+1)\big]$$

viii. if $J_c(\hat{\mathbf{x}}^i(k+1), k+1) > J_c(\hat{\mathbf{x}}^{i-1}(k+1), k+1)$, halve α and go to step 4(b)vii

ix. if $i = i_{\max}$, go to step 4c; otherwise, increment i and go to step 4(b)ii

(c) update the state estimate

$$\hat{\mathbf{x}}(k+1) = \mathcal{R}_{xx}^i(k+1)^{-1}\boldsymbol{\mathfrak{z}}_x^i(k+1)$$

(d) update the state estimation MSE

$$P(k+1) = \mathcal{R}_{xx}^i(k+1)^{-1}\mathcal{R}_{xx}^i(k+1)^{-T}$$

5. increment k and return to step 3

8.3.3 The backward smoothing IEKF

In the Gauss-Newton iterations of the classical IEKF, we continually re-linearize the measurement map about the current estimate $\hat{\mathbf{x}}^i(k+1)$. However, in each iteration we use the same value for $\bar{\mathbf{x}}(k+1)$. In the **backward smoothing IEKF**, also known as the **moving horizon estimator**, we refine the classical IEKF by computing the smoothed estimate $\mathbf{x}^{i*}(k)$ of $\mathbf{x}(k)$. We do this at every Gauss-Newton iteration, starting the smoothing algorithm from the current state estimate $\hat{\mathbf{x}}^i(k+1)$. We then propagate $\mathbf{x}^{i*}(k)$ through the dynamics map to improve upon the prediction $\bar{\mathbf{x}}(k+1)$.

In the backward smoothing IEKF, it is convenient to find the least-squares solution to the system

$$\begin{aligned} \mathbf{0} &= \mathbf{x}(k) - \hat{\mathbf{x}}(k) \\ \mathbf{0} &= \mathbf{v}(k) \\ \mathbf{0} &= \mathbf{z}(k+1) - \mathbf{h}(k+1, \mathbf{f}(k, \mathbf{x}(k), \mathbf{u}(k), \mathbf{v}(k))) \end{aligned}$$

which, when linearized about the smoothed state $\mathbf{x}^{i*}(k)$ and smoothed disturbance $\mathbf{v}^*(k)$ (starting the smoothing algorithm from $\hat{\mathbf{x}}^i(k+1)$), produces the system

$$\begin{aligned}
\mathbf{0} &= \mathbf{x}(k) - \bar{\mathbf{x}}(k) \\
\mathbf{0} &= \mathbf{v}(k) \\
\mathbf{0} &= \mathbf{z}(k+1) - \mathbf{h}(k+1, \hat{\mathbf{x}}^i(k+1)) \\
&\quad - H^i(k+1)\Big\{F^i(k)\big[\mathbf{x}(k) - \mathbf{x}^{i*}(k)\big] + \Gamma^i(k)\big[\mathbf{v}(k) - \mathbf{v}^*(k)\big]\Big\}
\end{aligned}$$

where

$$\begin{aligned}
\hat{\mathbf{x}}^i(k+1) &= \mathbf{f}(k, \mathbf{x}^{i*}(k), \mathbf{u}(k), \mathbf{v}^{i*}(k)) \\
H^i(k+1) &= \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}(k+1)}\right|_{k+1, \hat{\mathbf{x}}^i(k+1)} \\
F^i(k) &= \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}(k)}\right|_{k, \mathbf{x}^{i*}(k), \mathbf{u}(k), \mathbf{v}^{i*}(k)} \\
\Gamma^i(k) &= \left.\frac{\partial \mathbf{f}}{\partial \mathbf{v}(k)}\right|_{k, \mathbf{x}^{i*}(k), \mathbf{u}(k), \mathbf{v}^{i*}(k)}
\end{aligned}$$

The linearized equations can be solved by the Gauss-Newton method, checking that the cost $J_a(\mathbf{x}^i(k), \mathbf{v}(k), \hat{\mathbf{x}}^i(k+1), k)$ decreases with each iteration.

NB. The backward smoothing IEKF can be extended to smooth the previous m stages. Smoothing further into the past increases estimation accuracy at the expense of significant computation time, since the smoothing occurs at each Gauss-Newton iteration of each measurement update step.

8.4 The unscented Kalman filter

The EKF and IEKF have some problems:

1. Neglecting the higher order terms in the Taylor expansions of $\mathbf{f}$ and $\mathbf{h}$ means that the filter's accuracy is suboptimal. While this accuracy hit is usually acceptable, it occasionally leads to truly awful performance. In fact, there's no guarantee that the estimates won't **diverge** from the true states.

2. It can be difficult to derive analytical expressions for the Jacobians of $\mathbf{f}$ and $\mathbf{h}$. It can also be costly to compute them online. If we use the second-order EKF to address issue 1, then this computational issue is compounded by the need to derive and evaluate Hessians.

The **unscented Kalman filter** (UKF), also known as the **sigma points filter**, seeks to address these two issues. Of course, the UKF has issues of its own:

1. While the UKF retains higher-order terms than the EKF, it still doesn't include all the nonlinear terms in $\mathbf{f}$ and $\mathbf{h}$. In some cases, the UKF can also diverge. (As a rule of thumb, however, the UKF usually out-performs the EKF.)

2. The UKF is more computationally intensive than the EKF (roughly double). Both the UKF and EKF are still orders of magnitude faster than the backward smoothing IEKF, however.

3. There are divergence issues unique to the UKF. These may arise when $\mathbf{f}$ or $\mathbf{h}$ have discontinuities or singularities. In these cases, the EKF may out-perform the UKF.

8.4.1 Sigma points estimation

The general sigma points estimation problem is the following. Given $\mathbf{z} = \mathbf{h}(\mathbf{x})$, where $\mathbf{E}[\mathbf{x}] = \bar{\mathbf{x}}$ and $\mathbf{E}[(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T] = P$, find the mean and covariance of $\mathbf{z}$.

The sigma points filter works by generating a cluster of points in a hypervolume in $\mathbf{x}$-space, centered at $\bar{\mathbf{x}}$. Each point is then propagated through $\mathbf{h}$, resulting in a spread of points in $\mathbf{z}$-space. The mean and covariance of $\mathbf{z}$ are approximated by taking a weighted average of the resulting points.

Let $S^T \in \mathbf{R}^{n_x \times n_x}$ be a Cholesky factor of P, i.e. $SS^T = P$, and let $\mathbf{s}_j$ be the j^{th} column of S. Generate the $2n_x + 1$ sigma points $\mathbf{x}_i$, where

$$\mathbf{x}_i = \begin{cases} \bar{\mathbf{x}}, & i = 0 \\ \bar{\mathbf{x}} + \sqrt{n_x + \lambda}\, \mathbf{s}_i, & i \in \{1, \ldots, n_x\} \\ \bar{\mathbf{x}} - \sqrt{n_x + \lambda}\, \mathbf{s}_i, & i \in \{n_x + 1, \ldots, 2n_x\} \end{cases}$$

The parameter λ governs the hypervolume spanned by the sigma points; a bigger λ means the points will span a larger hypervolume. The suggested value of λ is

$$\lambda = \alpha^2(n_x + \kappa) - n_x$$

where $\alpha \in [10^{-4}, 1]$ and $\kappa \in \{0, 3 - n_x\}$ are tunable parameters.

Having generated the sigma points, we propagate them through the measurement map to produce

$$\boldsymbol{\zeta}_i = \mathbf{h}(\mathbf{x}_i)$$

for $i \in \{1, \ldots, 2n_x + 1\}$. We then approximate the mean and covariance of $\mathbf{z}$ by

$$\bar{\mathbf{z}} \approx \sum_{i=0}^{2n_x} w_i^{(m)} \boldsymbol{\zeta}_i$$

$$P_{zz} \approx \sum_{i=0}^{2n_x} w_i^{(c)} (\boldsymbol{\zeta}_i - \bar{\mathbf{z}})(\boldsymbol{\zeta}_i - \bar{\mathbf{z}})^T$$

where the weights in the mean calculation are

$$w_i^{(m)} = \begin{cases} \frac{\lambda}{n_x+\lambda} & i = 0 \\ \frac{1}{2(n_x+\lambda)} & i \in \{1, \ldots, 2n_x\} \end{cases}$$

and the weights in the covariance calculation are

$$w_i^{(c)} = \begin{cases} \frac{\lambda}{n_x+\lambda} + 1 - \alpha^2 + \beta & i = 0 \\ \frac{1}{2(n_x+\lambda)} & i \in \{1, \ldots, 2n_x\} \end{cases}$$

Note that β is another design parameter. In general, the optimal value of β depends on the distribution of $\mathbf{x}$. If $\mathbf{x}$ is Gaussian, then $\beta = 2$ is optimal. Otherwise, β can be determined by analyzing the higher-order terms in the Taylor expansion of $\mathbf{h}$. See pages 221-280 of *Kalman Filtering and Neural Networks* by E.A. Wan and R. van der Merwe for more details.

8.4.2 The UKF algorithm

This section shows how to embed the sigma points estimation technique shown above into an online filtering algorithm (the UKF).

Given $\hat{\mathbf{x}}(0)$ and $P(0)$, the UKF algorithm is:

1. **parameter definitions:** set the tuning parameters $\alpha \in [10^{-4}, 1]$, $\kappa \in \{0, 3 - n_x - n_v\}$ and β (use $\beta = 2$ if the state is modeled as Gaussian) and compute

$$\lambda = \alpha^2(n_x + n_v + \kappa) - (n_x + n_v)$$

2. set $k = 0$

3. compute the matrix square roots $S_x(k) \in \mathbf{R}^{n_x \times n_x}$ and $S_v(k) \in \mathbf{R}^{n_v \times n_v}$, such that $S_x(k)S_x(k)^T = P(k)$ and $S_v(k)S_v(k)^T = Q(k)$. Define the column vectors $\mathbf{s}_x^i(k) \in \mathbf{R}^{n_x}$ and $\mathbf{s}_v^i(k) \in \mathbf{R}^{n_v}$ such that

$$S_x(k) = \begin{bmatrix} \mathbf{s}_x^1(k) & \ldots & \mathbf{s}_x^{n_x}(k) \end{bmatrix}$$
$$S_v(k) = \begin{bmatrix} \mathbf{s}_v^1(k) & \ldots & \mathbf{s}_v^{n_v}(k) \end{bmatrix}$$

4. generate $2(n_x + n_v) + 1$ sigma points in $\mathbf{R}^{n_x+n_v}$ clustered about $(\mathbf{x}(k), \mathbf{v}(k))^T =$

$(\hat{\mathbf{x}}(k), \mathbf{0})^T$,

$$\begin{bmatrix} \mathbf{x}_k^i \\ \mathbf{v}_k^i \end{bmatrix} = \begin{cases} \begin{bmatrix} \hat{\mathbf{x}}(k) \\ \mathbf{0} \end{bmatrix} & i = 0 \\ \begin{bmatrix} \hat{\mathbf{x}}(k) + \sqrt{n_x + n_v + \lambda}\ \mathbf{s}_x^i(k) \\ \mathbf{0} \end{bmatrix} & i \in \{1, \ldots, n_x\} \\ \begin{bmatrix} \hat{\mathbf{x}}(k) - \sqrt{n_x + n_v + \lambda}\ \mathbf{s}_x^{i-n_x}(k) \\ \mathbf{0} \end{bmatrix} & i \in \{n_x + 1, \ldots, 2n_x\} \\ \begin{bmatrix} \hat{\mathbf{x}}(k) \\ \sqrt{n_x + n_v + \lambda}\ \mathbf{s}_v^{i-2n_x} \end{bmatrix} & i \in \{2n_x + 1, \ldots, 2n_x + n_v\} \\ \begin{bmatrix} \hat{\mathbf{x}}(k) \\ -\sqrt{n_x + n_v + \lambda}\ \mathbf{s}_v^{i-2n_x-n_v} \end{bmatrix} & i \in \{2n_x + n_v + 1, \ldots, 2(n_x + n_v)\} \end{cases}$$

5. propagate the sigma points through $\mathbf{f}$ and $\mathbf{h}$ to produce

$$\begin{aligned} \bar{\mathbf{x}}_{k+1}^i &= \mathbf{f}(k, \mathbf{x}_k^i, \mathbf{u}(k), \mathbf{v}_k^i) \\ \bar{\mathbf{z}}_{k+1}^i &= \mathbf{h}(k+1, \bar{\mathbf{x}}_{k+1}^i) \end{aligned}$$

6. predict the state and measurement

$$\begin{aligned} \bar{\mathbf{x}}(k+1) &= \sum_{i=0}^{2(n_x+n_v)} w_i^{(m)} \bar{\mathbf{x}}_{k+1}^i \\ \bar{\mathbf{z}}(k+1) &= \sum_{i=0}^{2(n_x+n_v)} w_i^{(m)} \bar{\mathbf{z}}_{k+1}^i \end{aligned}$$

and the associated covariances

$$\begin{aligned} \bar{P}(k+1) &= \sum_{i=0}^{2(n_x+n_v)} w_i^{(c)} \left[\bar{\mathbf{x}}_{k+1}^i - \bar{\mathbf{x}}(k+1)\right] \left[\bar{\mathbf{x}}_{k+1}^i - \bar{\mathbf{x}}(k+1)\right]^T \\ \bar{P}_{xz}(k+1) &= \sum_{i=0}^{2(n_x+n_v)} w_i^{(c)} \left[\bar{\mathbf{x}}_{k+1}^i - \bar{\mathbf{x}}(k+1)\right] \left[\bar{\mathbf{z}}_{k+1}^i - \bar{\mathbf{z}}(k+1)\right]^T \\ \bar{P}_{zz}(k+1) &= R(k+1) + \sum_{i=0}^{2(n_x+n_v)} w_i^{(c)} \left[\bar{\mathbf{z}}_{k+1}^i - \bar{\mathbf{z}}(k+1)\right] \left[\bar{\mathbf{z}}_{k+1}^i - \bar{\mathbf{z}}(k+1)\right]^T \end{aligned}$$

where the weights in the mean calculations are

$$w_i^{(m)} = \begin{cases} \frac{\lambda}{n_x+n_v+\lambda} & i = 0 \\ \frac{1}{2(n_x+n_v+\lambda)} & i \in \{1, \ldots, 2(n_x+n_v)\} \end{cases}$$

and the weights in the covariance calculation are

$$w_i^{(c)} = \begin{cases} \frac{\lambda}{n_x+n_v+\lambda} + 1 - \alpha^2 + \beta & i = 0 \\ \frac{1}{2(n_x+n_v+\lambda)} & i \in \{1, \ldots, 2(n_x+n_v)\} \end{cases}$$

7. apply the FEoLE to update the estimate and its MSE

$$\hat{\mathbf{x}}(k+1) = \bar{\mathbf{x}}(k+1) + \bar{P}_{xz}(k+1)\bar{P}_{zz}^{-1}(k+1)\big[\mathbf{z}(k+1) - \bar{\mathbf{z}}(k+1)\big]$$
$$P(k+1) = \bar{P}(k+1) - \bar{P}_{xz}(k+1)\bar{P}_{zz}(k+1)^{-1}\bar{P}_{xz}(k+1)^T$$

8. increment k and go to 3

8.4.3 Remarks on the UKF

- The UKF is easier to implement than the EKF, because there's no need to calculate Jacobians.
- The UKF avoids Jacobians by using what amounts to a finite difference approximation, where the step size $\Delta\mathbf{x}$ is chosen to be on the order of the statistical variations in $\mathbf{x}(k)$.
- The UKF is parsimonious, in that it uses just enough sigma points to capture the first few higher-order terms that the EKF neglects. Better performance can be achieved (at the cost of increased computation time) by using more sigma points. This is the basis for the particle filter.

8.5 The particle filter

The **particle filter** extends the ideas of the UKF to incorporate more sigma points, *randomly generated* in a cluster about $\hat{\mathbf{x}}$ and $\mathbf{E}[\mathbf{v}] = \mathbf{0}$. Under some mild assumptions, this results in better probabilistic models of $\mathbf{x}$ and $\mathbf{z}$, at the cost of increased computation time.

We begin the development of the particle filter with some background on Monte Carlo simulation.

8.5.1 Monte Carlo simulation

Monte Carlo techniques use random number generators to simulate random processes and to approximate probability density integrals (means, covariances, and higher moments).

Suppose you have a messy *pdf* $p(\mathbf{x})$, and you want to calculate

$$\mathbf{E}[\mathbf{f}(\mathbf{x})] = \int_{\mathbf{R}^{n_x}} \mathbf{f}(\mathbf{x})p(\mathbf{x})\mathrm{d}\mathbf{x}$$

This integral may be difficult or impossible to compute analytically, and even numerical integration may be costly. Another approach is to use a random number generate to produce

the N_s *independent* samples $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}_s^N$ from $p(\mathbf{x})$. For large N_s, the expectation is well-approximated by

$$\mathbf{E}[\mathbf{f}(\mathbf{x})] \approx \frac{1}{N_s} \sum_{i=1}^{N_s} \mathbf{f}(\mathbf{x}^i)$$

and as $N_s \to \infty$, this approximation becomes exact.

The above relies on the ability to sample $p(\mathbf{x})$ directly and independently, but for non-standard distributions this is rarely possible. A technique called **importance sampling** allows one to use any related distribution $q(\mathbf{x})$ to compute $\mathbf{E}[\mathbf{f}(\mathbf{x})]$ by Monte Carlo methods. One can write the expectation as

$$\begin{aligned}\mathbf{E}[\mathbf{f}(\mathbf{x})] &= \int_{\mathbf{R}^{n_x}} \mathbf{f}(\mathbf{x}) \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} \right) q(\mathbf{x}) \mathrm{d}\mathbf{x} \\ &= \frac{\int_{\mathbf{R}^{n_x}} \mathbf{f}(\mathbf{x}) \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} \right) q(\mathbf{x}) \mathrm{d}\mathbf{x}}{\int_{\mathbf{R}^{n_x}} \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} \right) q(\mathbf{x}) \mathrm{d}\mathbf{x}}\end{aligned}$$

where we include the denominator to handle the possibility that $p(\mathbf{x})$ may not be normalized.

We can approximate the expectation by generating the N_s independent samples $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}_s^N$ from $q(\mathbf{x})$, then computing

$$\mathbf{E}[\mathbf{f}(\mathbf{x})] \approx \sum_{i=1}^{N_s} w^i \mathbf{f}(\mathbf{x}^i)$$

where the weights are given by

$$\begin{aligned}\bar{w}^i &= \frac{p(\mathbf{x}^i)}{q(\mathbf{x}^i)} \\ w^i &= \frac{\bar{w}^i}{\sum_{j=1}^{N_s} \bar{w}^i}\end{aligned}$$

Since $\sum_{i=1}^{N_s} w^i = 1$ and $w^i \geq 0$ for all i, this amounts to approximating the original *pdf* $p(\mathbf{x})$ by

$$p(\mathbf{x}) \approx \sum_{i=1}^{N_s} w^i \delta(\mathbf{x} - \mathbf{x}^i)$$

Note that importance sampling tends to work better – meaning fewer samples are needed for a given approximation accuracy – the more closely $q(\mathbf{x})$ resembles the original *pdf* $p(\mathbf{x})$.

8.5.2 A simple particle filtering algorithm

In this section, we outline a simple algorithm for nonlinear state estimation using Monte Carlo methods. In subsequent sections we'll improve upon this algorithm.

This algorithm assumes $\mathbf{x}(0) \sim \mathcal{N}(\hat{\mathbf{x}}(0), P(0))$, $\mathbf{v}(k) \sim \mathcal{N}(\mathbf{0}, Q(k))$ and $\mathbf{w}(k) \sim \mathcal{N}(\mathbf{0}, R(k))$, where $\hat{\mathbf{x}}(0)$ and $P(0)$ are known. For a given number of Monte Carlo samples N_s, the particle filtering algorithm is the following.

1. set $k = 0$
2. for each $i \in \mathcal{S} = \{1, \ldots, N_s\}$, generate the candidate state history $\mathbf{x}^i(0), \ldots, \mathbf{x}^i(k)$ as follows:
   (a) sample $\mathbf{x}^i(0)$ from $\mathcal{N}(\hat{\mathbf{x}}(0), P(0))$
   (b) set $j = 0$
   (c) sample $\mathbf{v}^i(j)$ from $\mathcal{N}(\mathbf{0}, Q(j))$
   (d) compute $\mathbf{x}^i(j+1) = \mathbf{f}(j, \mathbf{x}^i(j), \mathbf{u}(j), \mathbf{v}^i(j))$
   (e) if $j = k - 1$, stop; otherwise, increment j and go to step (c)
3. for each $i \in \mathcal{S}$, compute the unnormalized sample weight

$$\bar{w}_k^i = \exp\Big\{ -\frac{1}{2} \sum_{j=1}^{k} \big[\mathbf{z}(j) - \mathbf{h}(j, \mathbf{x}^i(j))\big]^T R(j)^{-1} \big[\mathbf{z}(j) - \mathbf{h}(j, \mathbf{x}^i(j))\big] \Big\}$$

   then normalize them:

$$w_k^i = \frac{\bar{w}_k^i}{\sum_{j=1}^{N_s} \bar{w}_k^j}$$

4. compute the estimate

$$\hat{\mathbf{x}}(k) = \sum_{i=1}^{N_s} w_k^i \mathbf{x}^i(k)$$

   and its covariance

$$P(k) = \sum_{i=1}^{N_s} w_k^i \big[\mathbf{x}^i(k) - \hat{\mathbf{x}}(k)\big] \big[\mathbf{x}^i(k) - \hat{\mathbf{x}}(k)\big]^T$$

5. increment k and go to step 2

8.5.3 First improvement: recursive weight calculation

The algorithm in section 8.5.2 is costly to run, because at every stage k it requires computing N_s truth model simulations of $\mathbf{x}(0), \ldots, \mathbf{x}(k)$. We can address this issue by developing a recursive algorithm for w_k^i in terms of w_k^{i-1}.

Recall that the general form for the weights is

$$w_k^i = C_k \frac{p(\mathbf{x}^i(0), \ldots, \mathbf{x}^i(k) \mid \mathbf{z}^k)}{q(\mathbf{x}^i(0), \ldots, \mathbf{x}^i(k) \mid \mathbf{z}^k)}$$

for some normalizing constant C_k.

Using Bayes' rule, it can be shown that

$$q(\mathbf{x}^i(0),\ldots,\mathbf{x}^i(k) \mid \mathbf{z}^k) = q(\mathbf{x}^i(k) \mid \mathbf{x}^i(0),\ldots,\mathbf{x}^i(k-1),\mathbf{z}^k)q(\mathbf{x}^i(0),\ldots,\mathbf{x}^i(k-1) \mid \mathbf{z}^{k-1})$$

and

$$p(\mathbf{x}^i(0),\ldots,\mathbf{x}^i(k) \mid \mathbf{z}^k) = \frac{p(\mathbf{z}(k) \mid \mathbf{x}^i(k))p(\mathbf{x}^i(k) \mid \mathbf{x}^i(k-1))p(\mathbf{x}^i(0),\ldots,\mathbf{x}^i(k-1) \mid \mathbf{z}^{k-1})}{p(\mathbf{z}(k) \mid \mathbf{z}^{k-1})}$$

so, letting

$$\bar{w}_k^i = \frac{w_k^i}{C_k} = \frac{p(\mathbf{x}^i(0),\ldots,\mathbf{x}^i(k) \mid \mathbf{z}^k)}{q(\mathbf{x}^i(0),\ldots,\mathbf{x}^i(k) \mid \mathbf{z}^k)}$$

we have

$$\begin{aligned}
\bar{w}_k^i &= \frac{p(\mathbf{z}(k) \mid \mathbf{x}^i(k))p(\mathbf{x}^i(k) \mid \mathbf{x}^i(k-1))p(\mathbf{x}^i(0),\ldots,\mathbf{x}^i(k-1) \mid \mathbf{z}^{k-1})}{p(\mathbf{z}(k) \mid \mathbf{z}^{k-1})q(\mathbf{x}^i(k) \mid \mathbf{x}^i(0),\ldots,\mathbf{x}^i(k-1),\mathbf{z}^k)q(\mathbf{x}^i(0),\ldots,\mathbf{x}^i(k-1) \mid \mathbf{z}^{k-1})} \\
&= \left[\frac{p(\mathbf{z}(k) \mid \mathbf{x}^i(k))p(\mathbf{x}^i(k) \mid \mathbf{x}^i(k-1))}{p(\mathbf{z}(k) \mid \mathbf{z}^{k-1})q(\mathbf{x}^i(k) \mid \mathbf{x}^i(0),\ldots,\mathbf{x}^i(k-1),\mathbf{z}^k)}\right]\left[\frac{p(\mathbf{x}^i(0),\ldots,\mathbf{x}^i(k-1) \mid \mathbf{z}^{k-1})}{q(\mathbf{x}^i(0),\ldots,\mathbf{x}^i(k-1) \mid \mathbf{z}^{k-1})}\right] \\
&= \left[\frac{p(\mathbf{z}(k) \mid \mathbf{x}^i(k))p(\mathbf{x}^i(k) \mid \mathbf{x}^i(k-1))}{p(\mathbf{z}(k) \mid \mathbf{z}^{k-1})q(\mathbf{x}^i(k) \mid \mathbf{x}^i(0),\ldots,\mathbf{x}^i(k-1),\mathbf{z}^k)}\right]\frac{w_{k-1}^i}{C_{k-1}}
\end{aligned}$$

Recall that any *pdf* q from which we can generate independent samples is a valid distribution to use in importance sampling. If we choose q such that

$$q(\mathbf{x}^i(k) \mid \mathbf{x}^i(0),\ldots,\mathbf{x}^i(k-1),\mathbf{z}^k) = \frac{p(\mathbf{x}^i(k) \mid \mathbf{x}^i(k-1))}{C_{k-1}p(\mathbf{z}(k) \mid \mathbf{z}^{k-1})}$$

then we have the recursive formula for the weights:

$$\bar{w}_k^i = p(\mathbf{z}(k) \mid \mathbf{x}^i(k))w_{k-1}^i$$

With this recursion, given $\hat{\mathbf{x}}(0)$, $P(0)$, N_s and Gaussian assumptions on $\mathbf{v}(k)$ and $\mathbf{w}(k)$, the particle filtering algorithm in section 8.5.2 becomes

1. set $k = 0$. For each $i \in \mathcal{S}$, sample $\mathbf{x}^i(0)$ from $\mathcal{N}(\hat{\mathbf{x}}(0), P(0))$ and set $w_0^i = \frac{1}{N_s}$.
2. for each $i \in \mathcal{S}$, generate the candidate state $\mathbf{x}^i(k+1)$ and its weight w_{k+1}^i as follows:
   (a) sample $\mathbf{v}^i(k)$ from $\mathcal{N}(\mathbf{0}, Q(k))$
   (b) compute $\mathbf{x}^i(k+1) = \mathbf{f}(j, \mathbf{x}^i(k), \mathbf{u}(k), \mathbf{v}^i(k))$
   (c) compute the unnormalized sample weight

$$\begin{aligned}
\bar{w}_{k+1}^i &= p(\mathbf{z}(k+1) \mid \mathbf{x}^i(k+1))w_k^i \\
&= \exp\Big\{ -\frac{1}{2}\big[\mathbf{z}(k+1) - \mathbf{h}(k+1, \mathbf{x}^i(k+1))\big]^T R(k+1)^{-1} \\
&\qquad * \big[\mathbf{z}(k+1) - \mathbf{h}(k+1, \mathbf{x}^i(k+1))\big]\Big\}w_k^i
\end{aligned}$$

Then compute the normalized weight

$$w_{k+1}^i = \frac{\bar{w}_{k+1}^i}{\sum_{j=1}^{N_s} \bar{w}_{k+1}^j}$$

3. compute the estimate

$$\hat{\mathbf{x}}(k+1) = \sum_{i=1}^{N_s} w_{k+1}^i \mathbf{x}^i(k+1)$$

and its covariance

$$P(k+1) = \sum_{i=1}^{N_s} w_{k+1}^i \left[\mathbf{x}^i(k+1) - \hat{\mathbf{x}}(k+1)\right]\left[\mathbf{x}^i(k+1) - \hat{\mathbf{x}}(k+1)\right]^T$$

4. increment k and go to step 2

8.5.4 Second improvement: resampling

The algorithm in section 8.5.3 is more efficient than the one in section 8.5.2, but the recursive formula for the weights creates a problem of its own. If one candidate state history $\mathbf{x}^i(0), \ldots, \mathbf{x}^i(k)$ fits the data well for several consecutive stages, then the weight w_k^i will get large. In the extreme case, we may find that $w_k^i \approx 1$, while $w_k^j \approx 0$ for all $j \neq i$.

This is a Bad Thing, because while the i^{th} history may have accurately reflected the true state in the past, there's no guarantee that it will continue to do so. Indeed, the sample $\mathbf{v}^i(k+1)$ may be very different from the true disturbance $\mathbf{v}(k+1)$. This will cause a decrease in w_{k+1}^i relative to w_k^i, but w_{k+1}^i may still dominate all the other w_{k+1}^j for $j \neq i$. This will cause the filter to disproportionately weight the i^{th} history in calculating $\hat{\mathbf{x}}(k+1)$, resulting in a loss of accuracy. This problem is known as **collapse of particle diversity**.

To prevent the collapse of particle diversity, or at least mitigate it, we can modify the algorithm in section 8.5.3 to shuffle the candidate states at each stage and reset their weights. This process is called **resampling**.

A common scheme for resampling is the following. For each $i \in \mathcal{S}$, randomly replace $\mathbf{x}^i(k)$, the k^{th} state in the i^{th} Monte Carlo history, with the k^{th} state from some other history. The replacement selection is done such that histories with higher weights are chosen preferentially: $\mathbf{x}^j(k)$ is chosen as a replacement state with probability w_k^j. Once the states have been shuffled, all weights are reset to $1/N_s$.

On average, the net effect of this algorithm is to clone the high-weight states and kill off the low-weight states. This will typically produce a cluster of high-weight states at stage k. When this cluster is propagated through the dynamics, the disturbance and nonlinearities in $\mathbf{f}$ will tend to disperse the cluster. This dispersal, in conjunction with resetting the weights, promotes diversity.

With this resampling scheme, given $\hat{\mathbf{x}}(0)$, $P(0)$, N_s and Gaussian assumptions on $\mathbf{v}(k)$ and $\mathbf{w}(k)$, the particle filtering algorithm in section 8.5.3 becomes

1. set $k = 0$. For each $i \in \mathcal{S}$, sample $\mathbf{x}^i(0)$ from $\mathcal{N}(\hat{\mathbf{x}}(0), P(0))$ and set $w_0^i = \frac{1}{N_s}$

2. for each $i \in \mathcal{S}$, generate the candidate state $\mathbf{x}^i(k+1)$ and its weight w_{k+1}^i as follows:

   (a) sample $\mathbf{v}^i(k)$ from $\mathcal{N}(\mathbf{0}, Q(k))$

   (b) compute $\mathbf{x}^i(k+1) = \mathbf{f}(k, \mathbf{x}^i(k), \mathbf{u}(k), \mathbf{v}^i(k))$

   (c) compute the natural logarithm of the unnormalized sample weight

$$\begin{aligned}\ln \bar{w}_{k+1}^i &= \ln w_k^i + \ln p(\mathbf{z}(k+1) \mid \mathbf{x}^i(k+1)) \\ &= \ln w_k^i - \frac{1}{2}\big[\mathbf{z}(k+1) - \mathbf{h}(k+1, \mathbf{x}^i(k+1))\big]^T R(k+1)^{-1} \\ &\qquad * \big[\mathbf{z}(k+1) - \mathbf{h}(k+1, \mathbf{x}^i(k+1))\big]\end{aligned}$$

   Then find the unique $i^* \in \mathcal{S}$ such that $\ln \bar{w}_{k+1}^i \leq \ln w_{k+1}^{i^*}$ for all $i \in \mathcal{S}$.
   For each $i \in \mathcal{S}$, define the alternative, unnormalized weight

$$\tilde{w}_{k+1}^i = \exp\big\{\ln \bar{w}_{k+1}^i - \ln \bar{w}_{k+1}^{i^*}\big\}$$

   and compute the normalized weight

$$w_{k+1}^i = \frac{\tilde{w}_{k+1}^i}{\sum_{j=1}^{N_s} \tilde{w}_{k+1}^j}$$

3. compute the estimate

$$\hat{\mathbf{x}}(k+1) = \sum_{i=1}^{N_s} w_{k+1}^i \mathbf{x}^i(k+1)$$

   and its covariance

$$P(k+1) = \sum_{i=1}^{N_s} w_{k+1}^i \big[\mathbf{x}^i(k+1) - \hat{\mathbf{x}}(k+1)\big]\big[\mathbf{x}^i(k+1) - \hat{\mathbf{x}}(k+1)\big]^T$$

4. resample the states according to their weights:

   (a) sample η from the uniform distribution, $U(0, \frac{1}{N_s})$

   (b) calculate the $N_s + 1$ *cdf* constants

$$c_i = \begin{cases} 0 & i = 1 \\ \sum_{l=1}^{i-1} w_k^l & i \in \{2, \ldots, N_s\} \\ 1 + \epsilon & i = N_s + 1 \end{cases}$$

   where ϵ is a very small number, perhaps 10^{-8}

(c) set $i = 1$ and $j = 1$

(d) set $\eta_i = \eta + (i-1)\frac{1}{N_s}$

(e) if $\eta_i \leq c_{j+1}$, go to step 5f; otherwise, increment j and return to step 5e

(f) let $\mathbf{x}^i_{\text{new}}(k) = \mathbf{x}^j(k)$

(g) if $i = N_s$, stop; otherwise, increment i and go to step 5d

5. set $w^i_{k+1} = \frac{1}{N_s}$ for all $i \in \mathcal{S}$

6. increment k and go to step 2

NB. Along with adding resampling, the above algorithm also works with natural logarithms in step 2c. This is a minor tweak for numerical robustness.

8.5.5 The regularized particle filter

The particle filter in section 8.5.4 is fairly efficient. It's still orders of magnitude slower than the EKF or UKF, but much faster than the particle filter in section 8.5.2. It could be more efficient, though: it resamples at every time k, even if the weights w^i_k are roughly equal. A better algorithm would only resample when necessary.

To address this problem, we define the effective number of particles,

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N_s}(w^i_k)^2} \in [1, N_s]$$

and only resample when N_{eff} dips below a threshold N_T. Here $N_T \in \mathcal{S}$ is a design parameter: a larger N_T means more frequent resampling, more diversity, and more computation time.

The filter in 8.5.4 also decreases the risk of losing particle diversity. It could be better at promoting diversity, though: it only selects replacement particles from the set of existing particles, and high-weight particles can be cloned multiple times. A better algorithm would allow selection of replacement particles in the neighborhood of high-weight particles. This process of "smearing" particles is known as **dithering**.

Recall that in information sampling, we approximate the true *pdf* $p(\mathbf{x})$ by a weighted sum of Dirac delta functions. In the case of particle filtering, this means we make the approximation

$$p(\mathbf{x}(k) \mid \mathbf{z}^k) \approx \sum_{i=1}^{N_s} w^i_k \delta(\mathbf{x}(k) - \mathbf{x}^i(k)) \tag{10}$$

The Dirac deltas are difficult to work with, so we would like to replace them with some other set of functions with approximately the same properties.

Recall that the Dirac delta function is an infinitely narrow, infinitely tall spike that integrates to 1. Loosely, then, the Dirac delta function can be viewed as the limiting case of a Gaussian *pdf*:

$$\delta(y - y_0) = \lim_{\sigma \to 0} \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{ -\frac{(y-y_0)^2}{2\sigma^2} \right\} = \lim_{\sigma \to 0} \mathcal{N}(y; y_0, \sigma^2)$$

so to "spread out" the scalar Dirac delta function, we could use a scalar Gaussian *pdf* with a small but finite variance.

It turns out that in addition to the Gaussian *pdf*, a whole family of functions, called **kernel densities**, can be viewed in a similar light. Let $K : \mathbf{R}^{n_x} \to \mathbf{R}$ be a kernel density. Then K satisfies the following properties:

1. normalization:
$$\int_{\mathbf{R}^{n_x}} K(\boldsymbol{\beta}) \mathrm{d}\boldsymbol{\beta} = 1$$
2. symmetry:
$$K(-\boldsymbol{\beta}) = K(\boldsymbol{\beta})$$
3. zero mean:
$$\int_{\mathbf{R}^{n_x}} \boldsymbol{\beta} K(\boldsymbol{\beta}) \mathrm{d}\boldsymbol{\beta} = \mathbf{0}$$
4. bounded covariance:
$$\int_{\mathbf{R}^{n_x}} \boldsymbol{\beta}\boldsymbol{\beta}^T K(\boldsymbol{\beta}) \mathrm{d}\boldsymbol{\beta} \prec \infty$$

Kernel densities are not unique. In a general filtering problem, the selection of the kernel density requires minimizing the mean integrated square error between the true $p(\mathbf{x}(k) \mid \mathbf{z}^k)$ and the kernel approximation in (10). In the special case wherein all the particle weights are equal, the optimal kernel density is the Epanechnikov kernel,

$$K_{\text{opt}}(\boldsymbol{\beta}) = \begin{cases} \frac{n_x+2}{2c(n_x)}(1 - \boldsymbol{\beta}^T\boldsymbol{\beta}) & \sqrt{\boldsymbol{\beta}^T\boldsymbol{\beta}} < 1 \\ 0 & \sqrt{\boldsymbol{\beta}^T\boldsymbol{\beta}} \geq 1 \end{cases}$$

where $c(n_x)$ is the hypervolume of the unit hypersphere in $\mathbf{R}^{n_x}$: $c(1) = 2$, $c(2) = \pi$, $c(3) = 4\pi/3$, and so on. If the weights are not all equal, the Epanechnikov kernel is often used anyway as a suboptimal but good kernel density.

Using this kernel function to "fatten" the Dirac delta functions, our Monte Carlo approximation of $p(\mathbf{x}(k) \mid \mathbf{z}^k)$ is

$$p(\mathbf{x}(k) \mid \mathbf{z}^k) \approx \sum_{i=1}^{N_s} w_k^i K_h\big(S_k^{-1}[\mathbf{x}(k) - \mathbf{x}^i(k)]\big)$$

where S_k is a matrix square root of $P(k)$, so $S_k S_k^T = P(k)$, and $K_h : \mathbf{R}^{n_x} \to \mathbf{R}$ is a rescaling of our kernel function, parameterized by h:

$$K_h(\boldsymbol{\alpha}) = \frac{1}{h^{n_x}} K\Big(\frac{1}{h}\boldsymbol{\alpha}\Big)$$

The rescaling parameter h depends on the distribution $p(\mathbf{x}(k) \mid \mathbf{z}^k)$. Assuming a Gaussian distribution, the optimal value of h is

$$h_{\text{opt}} = \frac{a}{N_s^{1/(n_x+4)}}$$

where

$$a = \left[\frac{8}{c(n_x)}(n_x+4)(2\sqrt{\pi})^{n_x}\right]^{1/(n_x+4)}$$

The **regularized particle filter** combines need-based resampling and dithering with the algorithm in section 8.5.4. Given $\hat{\mathbf{x}}(0)$, $P(0)$, N_s and Gaussian assumptions on $\mathbf{v}(k)$ and $\mathbf{w}(k)$, the regularized particle filtering algorithm is

1. set $k = 0$. For each $i \in \mathcal{S}$, sample $\mathbf{x}^i(0)$ from $\mathcal{N}(\hat{\mathbf{x}}(0), P(0))$ and set $w_0^i = \frac{1}{N_s}$
2. for each $i \in \mathcal{S}$, generate the candidate state $\mathbf{x}^i(k+1)$ and its weight w_{k+1}^i as follows:
   (a) sample $\mathbf{v}^i(k)$ from $\mathcal{N}(\mathbf{0}, Q(k))$
   (b) compute $\mathbf{x}^i(k+1) = \mathbf{f}(j, \mathbf{x}^i(k), \mathbf{u}(k), \mathbf{v}^i(k))$
   (c) compute the natural logarithm of the unnormalized sample weight

$$\begin{aligned}\ln \bar{w}_{k+1}^i &= \ln w_k^i + \ln p(\mathbf{z}(k+1) \mid \mathbf{x}^i(k+1)) \\ &= \ln w_k^i - \frac{1}{2}\big[\mathbf{z}(k+1) - \mathbf{h}(k+1, \mathbf{x}^i(k+1))\big]^T R(k+1)^{-1} \\ &\quad * \big[\mathbf{z}(k+1) - \mathbf{h}(k+1, \mathbf{x}^i(k+1))\big]\end{aligned}$$

   Then find the unique $i^* \in \mathcal{S}$ such that $\ln \bar{w}_{k+1}^i \leq \ln w_{k+1}^{i^*}$ for all $i \in \mathcal{S}$.
   For each $i \in \mathcal{S}$, define the alternative, unnormalized weight

$$\tilde{w}_{k+1}^i = \exp\left\{\ln \bar{w}_{k+1}^i - \ln \bar{w}_{k+1}^{i^*}\right\}$$

   and compute the normalized weight

$$w_{k+1}^i = \frac{\tilde{w}_{k+1}^i}{\sum_{j=1}^{N_s} \tilde{w}_{k+1}^j}$$

3. compute the estimate

$$\hat{\mathbf{x}}(k+1) = \sum_{i=1}^{N_s} w_{k+1}^i \mathbf{x}^i(k+1)$$

   and its covariance

$$P(k+1) = \sum_{i=1}^{N_s} w_{k+1}^i \big[\mathbf{x}^i(k+1) - \hat{\mathbf{x}}(k+1)\big]\big[\mathbf{x}^i(k+1) - \hat{\mathbf{x}}(k+1)\big]^T$$

4. compute the effective number of particles

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N_s} (w_k^i)^2}$$

   If $N_{\text{eff}} \geq N_T$, increment k and go to step 9; otherwise, go to step 5

5. resample the states according to their weights:

   (a) calculate the N_s *cdf* constants

$$c_i = \begin{cases} \sum_{l=1}^{i-1} w_k^l & i \in \{2, \ldots, N_s\} \\ 1 + \epsilon & i = N_s + 1 \end{cases}$$

   where ϵ is a very small number, perhaps 10^{-8}

   (b) set $i = 1$ and $j = 1$

   (c) sample η from the uniform distribution, $U(0, \frac{1}{N_s})$

   (d) set $\eta_i = \eta + (i-1)\frac{1}{N_s}$

   (e) if $\eta_i \leq c_{j+1}$, go to step 5f; otherwise, increment j and return to step 5e

   (f) let $\mathbf{x}_{\text{new}}^i(k) = \mathbf{x}^j(k)$

   (g) if $i = N_s$, stop; otherwise, increment i and go to step 5d

6. generate the N_s independent samples $\boldsymbol{\beta}^1, \ldots, \boldsymbol{\beta}^{N_s}$ from $K_{\text{opt}}(\boldsymbol{\beta})$

7. compute the matrix square root S_{k+1} such that $S_{k+1} S_{k+1}^T = P(k+1)$ and set

$$\mathbf{x}^i(k+1) = \mathbf{x}_{\text{new}}^i(k+1) + h_{\text{opt}} S_{k+1} \boldsymbol{\beta}^i$$

   where

$$h_{\text{opt}} = \frac{a}{N_s^{1/(n_x+4)}}, \quad a = \left[\frac{8}{c(n_x)} (n_x + 4)(2\sqrt{\pi})^{n_x} \right]^{1/(n_x+4)}$$

8. set $w_{k+1}^i = \frac{1}{N_s}$ for all $i \in \mathcal{S}$

9. increment k and go to step 2