NONLINEAR KALMAN FILTERS

6.1: Extended Kalman filters

- We return to the basic problem of estimating the present hidden state (vector) value of a dynamic system, using noisy measurements that are somehow related to that state (vector).
- We now examine the nonlinear case, with system dynamics

$$x_k = f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})$$

$$z_k = h_k(x_k, u_k, v_k),$$

  where u_k is a known (deterministic/measured) input signal, w_k is a process-noise random input, and v_k is a sensor-noise random input.
- There are three basic nonlinear generalizations to KF
  - Extended Kalman filter (EKF): *Analytic linearization* of the model at each point in time. Problematic, but still popular.
  - Sigma-point (Unscented) Kalman filter (SPKF/UKF): *Statistical/ empirical linearization* of the model at each point in time. Much better than EKF, at same computational complexity.
  - Particle filters: The most precise, but often thousands of times more computations required than either EKF/SPKF. Directly approximates the integrals required to compute $f(x_k \mid \mathbb{Z}_k)$ using Monte-Carlo integration techniques.
- In this chapter, we present the EKF and SPKF.

The Extended Kalman Filter (EKF)

- The EKF makes two simplifying assumptions when adapting the general sequential inference equations to a nonlinear system:
  - In computing state estimates, EKF assumes $\mathbb{E}[\text{fn}(x)] \approx \text{fn}(\mathbb{E}[x])$;
  - In computing covariance estimates, EKF uses Taylor series to linearize the system equations around the present operating point.
- Here, we will show how to apply these approximations and assumptions to derive the EKF equations from the general six steps.

EKF step 1a: State estimate time update.

- The state prediction step is approximated as

$$\begin{aligned}\hat{x}_k^- &= \mathbb{E}[f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1}) \mid \mathbb{Z}_{k-1}] \\ &\approx f_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, \bar{w}_{k-1}),\end{aligned}$$

  where $\bar{w}_{k-1} = \mathbb{E}[w_{k-1}]$. (Often, $\bar{w}_{k-1} = 0$.)
- That is, we approximate the expected value of the state by assuming it is reasonable to propagate $\hat{x}_{k-1}^+$ and $\bar{w}_{k-1}$ through the state eqn.

EKF step 1b: Error covariance time update.

- The covariance prediction step is accomplished by first making an approximation for $\tilde{x}_k^-$.

$$\begin{aligned}\tilde{x}_k^- &= x_k - \hat{x}_k^- \\ &= f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1}) - f_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, \bar{w}_{k-1}).\end{aligned}$$

- The first term is expanded as a Taylor series around the prior operating "point" which is the set of values $\{\hat{x}_{k-1}^+, u_{k-1}, \bar{w}_{k-1}\}$

$$x_k \approx f_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, \bar{w}_{k-1}) + \underbrace{\left.\frac{\mathrm{d}f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})}{\mathrm{d}x_{k-1}}\right|_{x_{k-1}=\hat{x}_{k-1}^+}}_{\text{Defined as } \hat{A}_{k-1}} (x_{k-1} - \hat{x}_{k-1}^+) + \underbrace{\left.\frac{\mathrm{d}f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})}{\mathrm{d}w_{k-1}}\right|_{w_{k-1}=\bar{w}_{k-1}}}_{\text{Defined as } \hat{B}_{k-1}} (w_{k-1} - \bar{w}_{k-1}).$$

- This gives $\tilde{x}_k^- \approx \left(\hat{A}_{k-1}\tilde{x}_{k-1}^+ + \hat{B}_{k-1}\widetilde{w}_{k-1}\right)$.
- Substituting this to find the predicted covariance:

$$\begin{aligned}\Sigma_{\tilde{x},k}^- &= \mathbb{E}[(\tilde{x}_k^-)(\tilde{x}_k^-)^T] \\ &\approx \hat{A}_{k-1}\Sigma_{\tilde{x},k-1}^+\hat{A}_{k-1}^T + \hat{B}_{k-1}\Sigma_{\widetilde{w}}\hat{B}_{k-1}^T.\end{aligned}$$

- Note, by the chain rule of total differentials,

$$\begin{aligned}\mathrm{d}f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1}) = &\frac{\partial f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})}{\partial x_{k-1}}\mathrm{d}x_{k-1} + \\ &\frac{\partial f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})}{\partial u_{k-1}}\mathrm{d}u_{k-1} + \\ &\frac{\partial f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})}{\partial w_{k-1}}\mathrm{d}w_{k-1}\end{aligned}$$

$$\begin{aligned}\frac{\mathrm{d}f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})}{\mathrm{d}x_{k-1}} = &\frac{\partial f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})}{\partial x_{k-1}} + \\ &\frac{\partial f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})}{\partial u_{k-1}}\underbrace{\frac{\mathrm{d}u_{k-1}}{\mathrm{d}x_{k-1}}}_{0} + \\ &\frac{\partial f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})}{\partial w_{k-1}}\underbrace{\frac{\mathrm{d}w_{k-1}}{\mathrm{d}x_{k-1}}}_{0}\end{aligned}$$

$$= \frac{\partial f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})}{\partial x_{k-1}}.$$

- Similarly,

$$\frac{\mathrm{d} f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})}{\mathrm{d} w_{k-1}} = \frac{\partial f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})}{\partial w_{k-1}}.$$

- The distinction between the total differential and the partial differential is not critical at this point, but will be when we look at parameter estimation using extended Kalman filters.

EKF step 1c: Output estimate (where $\bar{v}_k = \mathbb{E}[v_k]$).

- The system output is estimated to be

$$\begin{aligned}\hat{z}_k &= \mathbb{E}[h_k(x_k, u_k, v_k) \mid \mathbb{Z}_{k-1}] \\ &\approx h_k(\hat{x}_k^-, u_k, \bar{v}_k).\end{aligned}$$

- That is, it is assumed that propagating $\hat{x}_k^-$ and the mean sensor noise is the best approximation to estimating the output.

EKF step 2a: Estimator gain matrix.

- The output prediction error may then be approximated

$$\begin{aligned}\tilde{z}_k &= z_k - \hat{z}_k \\ &= h_k(x_k, u_k, v_k) - h_k(\hat{x}_k^-, u_k, \bar{v}_k)\end{aligned}$$

using again a Taylor-series expansion on the first term.

$$z_k \approx h_k(\hat{x}_k^-, u_k, \bar{v}_k) + \underbrace{\left.\frac{\mathrm{d} h_k(x_k, u_k, v_k)}{\mathrm{d} x_k}\right|_{x_k = \hat{x}_k^-}}_{\text{Defined as } \hat{C}_k} (x_k - \hat{x}_k^-)$$

$$+ \underbrace{\left.\frac{\mathrm{d}h_k(x_k, u_k, v_k)}{\mathrm{d}v_k}\right|_{v_k=\bar{v}_k}}_{\text{Defined as } \hat{D}_k} (v_k - \bar{v}_k).$$

- Note, much like we saw in Step 1b,

$$\frac{\mathrm{d}h_k(x_k, u_k, v_k)}{\mathrm{d}x_k} = \frac{\partial h_k(x_k, u_k, v_k)}{\partial x_k}$$
$$\frac{\mathrm{d}h_k(x_k, u_k, v_k)}{\mathrm{d}v_k} = \frac{\partial h_k(x_k, u_k, v_k)}{\partial v_k}.$$

- From this, we can compute such necessary quantities as

$$\Sigma_{\tilde{z},k} \approx \hat{C}_k \Sigma^-_{\tilde{x},k} \hat{C}_k^T + \hat{D}_k \Sigma_{\tilde{v}} \hat{D}_k^T,$$
$$\Sigma^-_{\tilde{x}\tilde{z},k} \approx \mathbb{E}[(\tilde{x}_k^-)(\hat{C}_k \tilde{x}_k^- + \hat{D}_k \tilde{v}_k)^T] = \Sigma^-_{\tilde{x},k} \hat{C}_k^T.$$

- These terms may be combined to get the Kalman gain

$$L_k = \Sigma^-_{\tilde{x},k} \hat{C}_k^T \left[\hat{C}_k \Sigma^-_{\tilde{x},k} \hat{C}_k^T + \hat{D}_k \Sigma_{\tilde{v}} \hat{D}_k^T\right]^{-1}.$$

EKF step 2b: State estimate measurement update.

- The fifth step is to compute the *a posteriori* state estimate by updating the *a priori* estimate

$$\hat{x}_k^+ = \hat{x}_k^- + L_k(z_k - \hat{z}_k).$$

EKF step 2c: Error covariance measurement update.

- Finally, the updated covariance is computed as

$$\Sigma^+_{\tilde{x},k} = \Sigma^-_{\tilde{x},k} - L_k \Sigma_{\tilde{z},k} L_k^T = (I - L_k \hat{C}_k) \Sigma^-_{\tilde{x},k}.$$

Summary of the nonlinear extended Kalman filter

Nonlinear state-space model:

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1})$$
$$z_k = h(x_k, u_k, v_k),$$

where w_k and v_k are independent, Gaussian noise processes of covariance matrices $\Sigma_{\tilde{w}}$ and $\Sigma_{\tilde{v}}$, respectively.

Definitions:

$$\hat{A}_k = \left.\frac{\mathrm{d}f_k(x_k, u_k, w_k)}{\mathrm{d}x_k}\right|_{x_k=\hat{x}_k^+} \qquad \hat{B}_k = \left.\frac{\mathrm{d}f_k(x_k, u_k, w_k)}{\mathrm{d}w_k}\right|_{w_k=\bar{w}_k}$$
$$\hat{C}_k = \left.\frac{\mathrm{d}h_k(x_k, u_k, v_k)}{\mathrm{d}x_k}\right|_{x_k=\hat{x}_k^-} \qquad \hat{D}_k = \left.\frac{\mathrm{d}h_k(x_k, u_k, v_k)}{\mathrm{d}v_k}\right|_{v_k=\bar{v}_k}.$$

Initialization: For $k = 0$, set

$$\hat{x}_0^+ = \mathbb{E}[x_0]$$
$$\Sigma_{\tilde{x},0}^+ = \mathbb{E}\big[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T\big].$$

Computation: For $k = 1, 2, \ldots$ compute:

State estimate time update: $\hat{x}_k^- = f_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, \bar{w}_{k-1}).$

Error covariance time update: $\Sigma_{\tilde{x},k}^- = \hat{A}_{k-1}\Sigma_{\tilde{x},k-1}^+\hat{A}_{k-1}^T + \hat{B}_{k-1}\Sigma_{\tilde{w}}\hat{B}_{k-1}^T.$

Output estimate: $\hat{z}_k = h_k(\hat{x}_k^-, u_k, \bar{v}_k).$

Estimator gain matrix: $L_k = \Sigma_{\tilde{x},k}^-\hat{C}_k^T[\hat{C}_k\Sigma_{\tilde{x},k}^-\hat{C}_k^T + \hat{D}_k\Sigma_{\tilde{v}}\hat{D}_k^T]^{-1}.$

State estimate measurement update: $\hat{x}_k^+ = \hat{x}_k^- + L_k\big(z_k - \hat{z}_k\big).$

Error covariance measurement update: $\Sigma_{\tilde{x},k}^+ = (I - L_k\hat{C}_k)\Sigma_{\tilde{x},k}^-.$

6.2: An EKF example, with code

- Consider an example of the EKF in action, with the following system dynamics:

$$x_{k+1} = f_k(x_k, u_k, w_k) = \sqrt{5 + x_k} + w_k$$

$$z_k = h_k(x_k, u_k, v_k) = x_k^3 + v_k$$

with $\Sigma_{\tilde{w}} = 1$ and $\Sigma_{\tilde{v}} = 2$.

- To implement EKF, we must determine $\hat{A}_k$, $\hat{B}_k$, $\hat{C}_k$, and $\hat{D}_k$.

$$\hat{A}_k = \left.\frac{\partial f_k(x_k, u_k, w_k)}{\partial x_k}\right|_{x_k=\hat{x}_k^+} = \left.\frac{\partial\left(\sqrt{5 + x_k} + w_k\right)}{\partial x_k}\right|_{x_k=\hat{x}_k^+} = \frac{1}{2\sqrt{5 + \hat{x}_k^+}}$$

$$\hat{B}_k = \left.\frac{\partial f_k(x_k, u_k, w_k)}{\partial w_k}\right|_{w_k=\bar{w}_k} = \left.\frac{\partial\left(\sqrt{5 + x_k} + w_k\right)}{\partial w_k}\right|_{w_k=\bar{w}_k} = 1$$

$$\hat{C}_k = \left.\frac{\partial h_k(x_k, u_k, v_k)}{\partial x_k}\right|_{x_k=\hat{x}_k^-} = \left.\frac{\partial\left(x_k^3 + v_k\right)}{\partial x_k}\right|_{x_k=\hat{x}_k^-} = 3(\hat{x}_k^-)^2$$

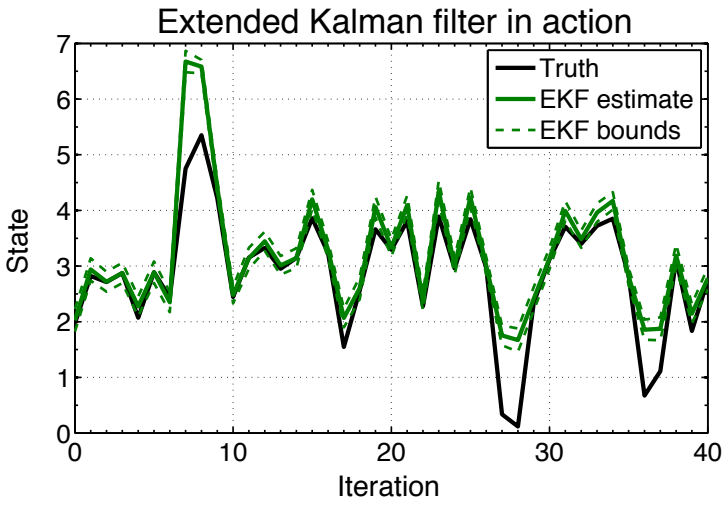
$$\hat{D}_k = \left.\frac{\partial h_k(x_k, u_k, v_k)}{\partial v_k}\right|_{v_k=\bar{v}_k} = \left.\frac{\partial\left(x_k^3 + v_k\right)}{\partial v_k}\right|_{v_k=\bar{v}_k} = 1.$$

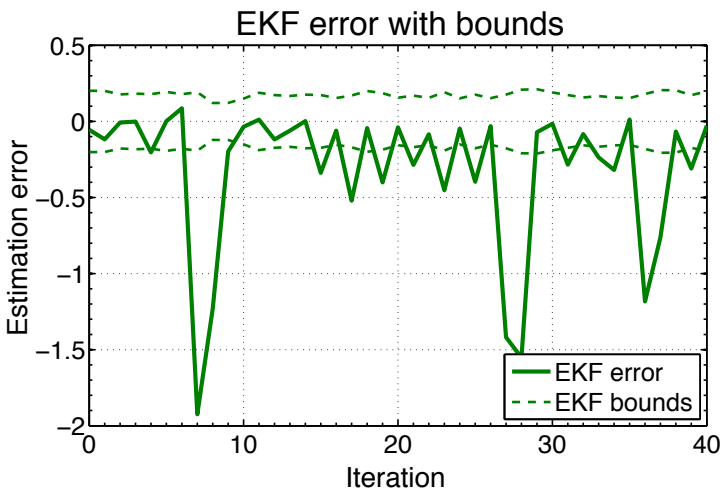
- The following is some sample code to implement an EKF.
  - Note that the steps for calculating the plant and the $\hat{A}$, $\hat{B}$, $\hat{C}$, and $\hat{D}$ matrices will depend on the nonlinear system underlying the estimation problem.

```
% Initialize simulation variables
SigmaW = 1; % Process noise covariance
SigmaV = 2; % Sensor noise covariance
maxIter = 40;
```

```
xtrue = 2 + randn(1);  % Initialize true system initial state
xhat = 2;                  % Initialize Kalman filter initial estimate
SigmaX = 1;            % Initialize Kalman filter covariance
u = 0;                 % Unknown initial driving input: assume zero

% Reserve storage for variables we might want to plot/evaluate
xstore = zeros(maxIter+1,length(xtrue)); xstore(1,:) = xtrue;
xhatstore = zeros(maxIter,length(xhat));
SigmaXstore = zeros(maxIter,length(xhat)^2);

for k = 1:maxIter,
  % EKF Step 0: Compute Ahat, Bhat
  % Note: For this example, x(k+1) = sqrt(5+x(k)) + w(k)
  Ahat = 0.5/sqrt(5+xhat); Bhat = 1;

  % EKF Step 1a: State estimate time update
  % Note: You need to insert your system's f(...) equation here
  xhat = sqrt(5+xhat);

  % EKF Step 1b: Error covariance time update
  SigmaX = Ahat*SigmaX*Ahat' + Bhat*SigmaW*Bhat';

  % [Implied operation of system in background, with
  % input signal u, and output signal z]
  w = chol(SigmaW)'*randn(1);
  v = chol(SigmaV)'*randn(1);
  ztrue = xtrue^3 + v;  % z is based on present x and u
  xtrue = sqrt(5+xtrue) + w;  % future x is based on present u

  % EKF Step 1c: Estimate system output
  % Note: You need to insert your system's h(...) equation here
  Chat = 3*xhat^2; Dhat = 1;
  zhat = xhat^3;

  % EKF Step 2a: Compute Kalman gain matrix
  L = SigmaX*Chat'/(Chat*SigmaX*Chat' + Dhat*SigmaV*Dhat');

  % EKF Step 2b: State estimate measurement update
  xhat = xhat + L*(ztrue - zhat);
  xhat = max(-5,xhat); % don't get square root of negative xhat!

  % EKF Step 2c: Error covariance measurement update
```

```
  SigmaX = SigmaX - L*Chat*SigmaX;

  % [Store information for evaluation/plotting purposes]
  xstore(k+1,:) = xtrue; xhatstore(k,:) = xhat;
  SigmaXstore(k,:) = SigmaX(:);
end;

figure(1); clf;
plot(0:maxIter-1,xstore(1:maxIter),'k-',0:maxIter-1,xhatstore,'b--', ...
  0:maxIter-1,xhatstore+3*sqrt(SigmaXstore),'m-.',...
  0:maxIter-1,xhatstore-3*sqrt(SigmaXstore),'m-.'); grid;
legend('true','estimate','bounds'); xlabel('Iteration'); ylabel('State');
title('Extended Kalman filter in action');

figure(2); clf;
plot(0:maxIter-1,xstore(1:maxIter)-xhatstore,'b-',0:maxIter-1, ...
  3*sqrt(SigmaXstore),'m--',0:maxIter-1,-3*sqrt(SigmaXstore),'m--');
grid; legend('Error','bounds',0);
title('EKF Error with bounds');
xlabel('Iteration'); ylabel('Estimation Error');
```





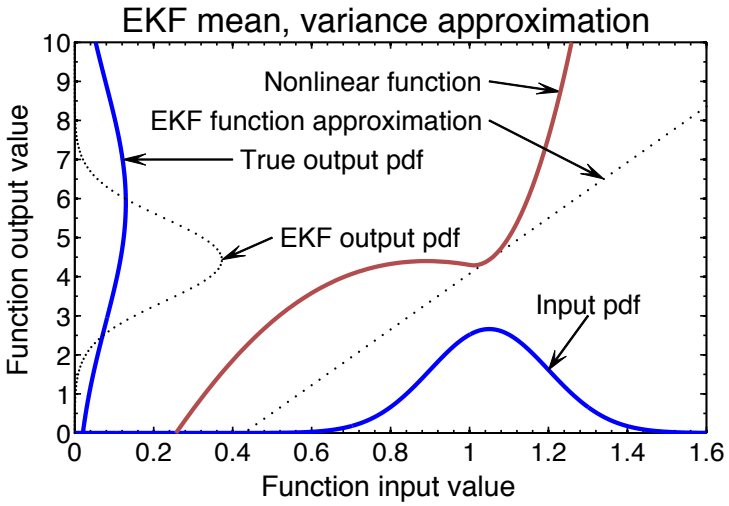
6.3: Problems with EKF, improved with sigma-point methods

- The EKF is the best known and most used nonlinear Kalman filter.
- However, it has serious flaws that can be remedied fairly easily.

ISSUE: How input mean and covariance are propagated through static nonlinear function to create output mean and covariance estimates.

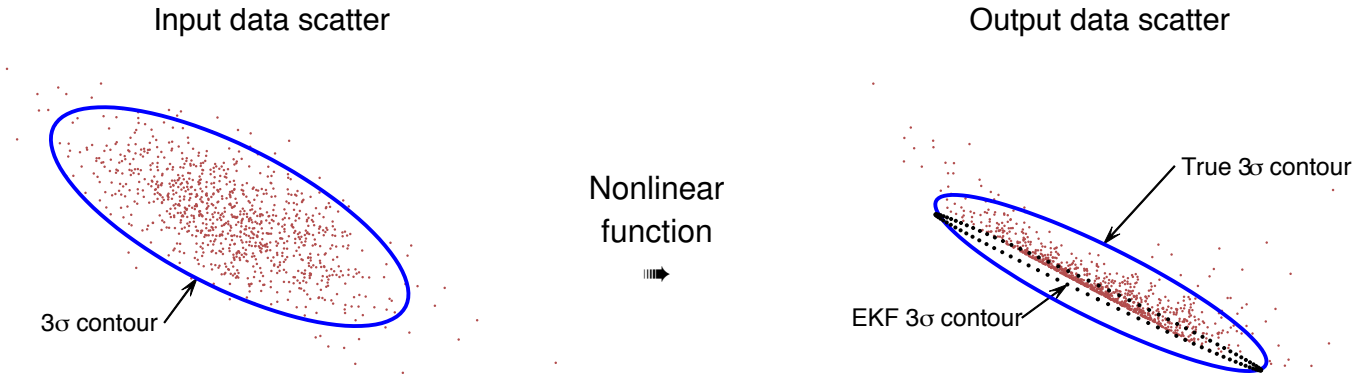
- Recall that the EKF, when computing mean estimates in Steps 1a and 1c, makes the simplification $\mathbb{E}[\text{fn}(x)] \approx \text{fn}(\mathbb{E}[x])$.
  - This is not true in general, and not necessarily even close to true (depending on "how nonlinear" the function $\text{fn}(\cdot)$ is).
- Also, in EKF Steps 1b and 2a, a Taylor-series expansion is performed as part of the calculation of output-variable covariance.
  - Nonlinear terms are dropped, resulting in a loss of accuracy.

- A simple one-dimensional example illustrates these two effects. Consider the figure:
- The nonlinear function is drawn, and the input random-variable PDF is shown on the horizontal axis, with mean 1.05.



- The straight dotted line is the linearized approximation used by the EKF to find the output mean and covariance.
- The EKF-approximated PDF is compared to a Gaussian PDF having same mean and variance of the true data on the vertical axis.

- We notice significant differences between the means and variances: EKF approach is not producing an accurate estimate of either.
- For a two-dimensional example, consider the following figure.



- Left frame shows a cloud of Gaussian-distributed random points used as input to this function, and
- Right frame shows the transformed set of output points.
- The actual 95 % confidence interval (indicative of a contour of the Gaussian PDF describing the output covariance and mean) is compared to EKF-estimated confidence interval.
  - Again, EKF is very far from the truth.
- We can improve on mean and covariance propagation through the state and output equations using a “sigma-point” approach.

6.4: Approximating statistics with sigma points

- We now look at a different approach to characterizing the mean and covariance of the output of a nonlinear function.
- We avoid Taylor-series expansion; instead, a number of function evaluations are performed whose results are used to compute estimated mean and covariance matrices.
- This has several advantages:
  1. Derivatives do not need to be computed (which is one of the most error-prone steps when implementing EKF), also implying
  2. The original functions do not need to be differentiable, and
  3. Better covariance approximations are usually achieved, relative to EKF, allowing for better state estimation,
  4. All with comparable computational complexity to EKF.
- A set of sigma points $\mathcal{X}$ is chosen so that the (possibly weighted) mean and covariance of the points exactly matches the mean $\bar{x}$ and covariance $\Sigma_{\tilde{x}}$ of the *a priori* random variable being modeled.
- These points are then passed through the nonlinear function, resulting in a transformed set of points $\mathcal{Z}$.
- The *a posteriori* mean $\bar{z}$ and covariance $\Sigma_{\tilde{z}}$ are then approximated by the mean and covariance of these transformed points $\mathcal{Z}$.
- Note that the sigma points comprise a fixed small number of vectors that are calculated deterministically—not like particle filter methods.
- Specifically, if input RV x has dimension L, mean $\bar{x}$, and covariance $\Sigma_{\tilde{x}}$, then $p + 1 = 2L + 1$ sigma points are generated as the set

$$\mathcal{X} = \{\bar{x}, \bar{x} + \gamma\sqrt{\Sigma_{\tilde{x}}}, \bar{x} - \gamma\sqrt{\Sigma_{\tilde{x}}}\},$$

with members of $\mathcal{X}$ indexed from 0 to p, and where the matrix square root $R = \sqrt{\Sigma}$ computes a result such that $\Sigma = RR^T$.

- Usually, the efficient *Cholesky decomposition* is used, resulting in lower-triangular R. (Take care: MATLAB, by default, returns an upper-triangular matrix that must be transposed.)

- The weighted mean and covariance of $\mathcal{X}$ are equal to the original mean and covariance of x for some $\{\gamma, \alpha^{(\mathrm{m})}, \alpha^{(\mathrm{c})}\}$ if we compute

$$\bar{x} = \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \mathcal{X}_i \qquad \text{and} \qquad \Sigma_{\tilde{x}} = \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} (\mathcal{X}_i - \bar{x})(\mathcal{X}_i - \bar{x})^T,$$

where $\mathcal{X}_i$ is the ith member of $\mathcal{X}$, and both $\alpha_i^{(m)}$ and $\alpha_i^{(\mathrm{c})}$ are real scalars where $\alpha_i^{(m)}$ and $\alpha_i^{(\mathrm{c})}$ must both sum to one.

- The various sigma-point methods differ only in the choices taken for these weighting constants.
- Values used by the *Unscented Kalman Filter* (UKF) and the *Central Difference Kalman Filter* (CDKF):

Method	γ	$\alpha_0^{(\mathrm{m})}$	$\alpha_k^{(\mathrm{m})}$	$\alpha_0^{(\mathrm{c})}$	$\alpha_k^{(\mathrm{c})}$
UKF	$\sqrt{L+\lambda}$	$\frac{\lambda}{L+\lambda}$	$\frac{1}{2(L+\lambda)}$	$\frac{\lambda}{L+\lambda} + (1-\alpha^2+\beta)$	$\frac{1}{2(L+\lambda)}$
CDKF	h	$\frac{h^2-L}{h^2}$	$\frac{1}{2h^2}$	$\frac{h^2-L}{h^2}$	$\frac{1}{2h^2}$

$\lambda = \alpha^2(L+\kappa) - L$ is a scaling parameter, with $(10^{-2} \leq \alpha \leq 1)$. Note that this α is different from $\alpha^{(\mathrm{m})}$ and $\alpha^{(\mathrm{c})}$. κ is either 0 or $3-L$. β incorporates prior information. For Gaussian RVs, $\beta = 2$. h may take any positive value. For Gaussian RVs, $h = \sqrt{3}$.

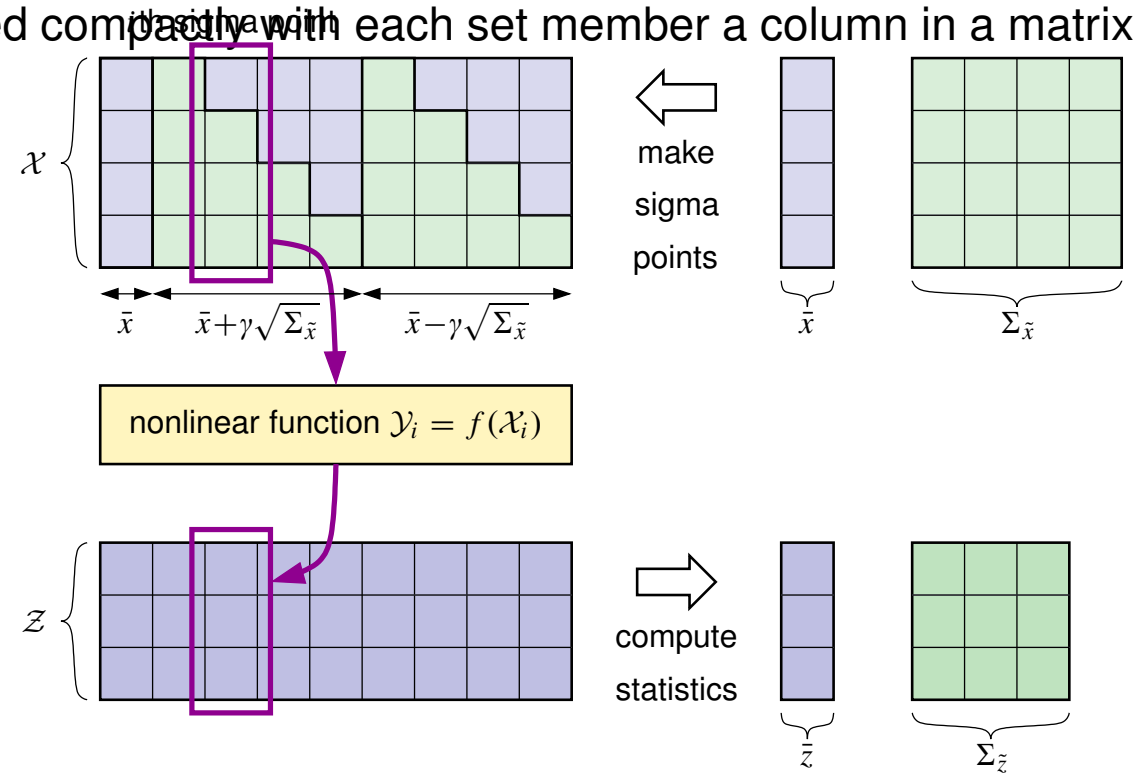
- UKF and CDKF are derived quite differently, but the final methods are essentially identical.

- CDKF has only one "tuning parameter" h, so implementation is simpler. It also has marginally higher theoretic accuracy than UKF.

- Output sigma points are computed: $\mathcal{Z}_i = f(\mathcal{X}_i)$. Then, the output mean and covariance are computed as well:

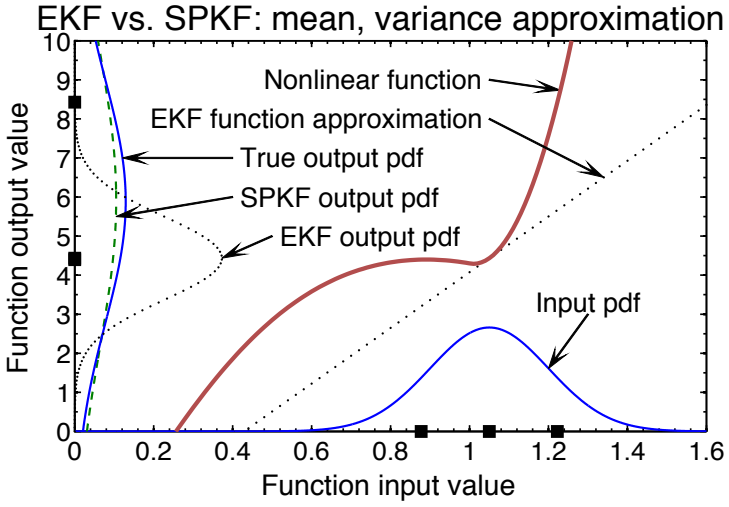
$$\bar{z} = \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \mathcal{Z}_i \qquad \text{and} \qquad \Sigma_{\tilde{z}} = \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} (\mathcal{Z}_i - \bar{z})(\mathcal{Z}_i - \bar{z})^T.$$

- The diagram illustrates the overall process, with the sets $\mathcal{X}$ and $\mathcal{Z}$ stored compactly with each set member a column in a matrix:

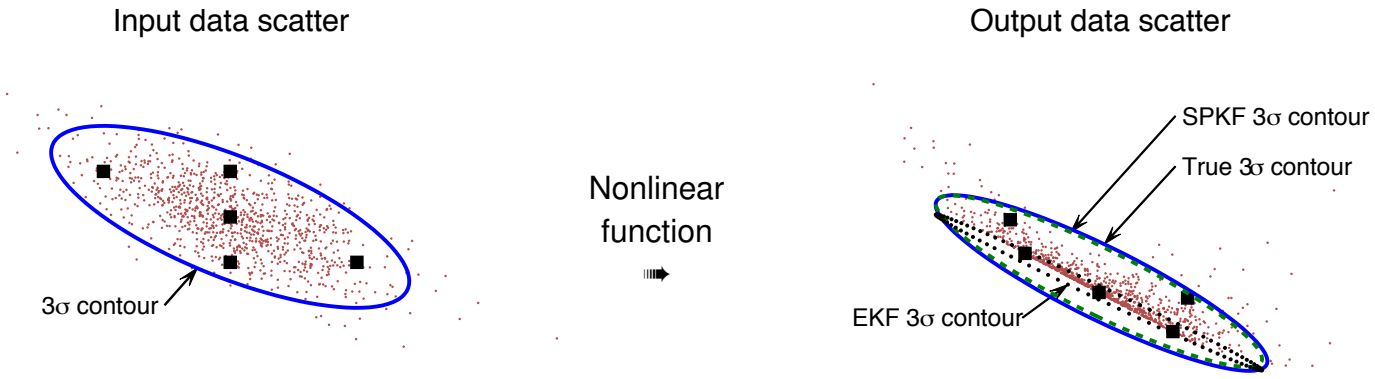


- Before introducing the SPKF algorithm, we re-examine the prior 1D/2D examples using sigma-point methods.

- In the 1D example, three input sigma points are needed and map to the output three sigma points shown.



- The mean and variance of the sigma-point method is shown as a dashed-line PDF and closely matches the true mean and variance.
- For the 2D example, five sigma points represent the input random-variable PDF (on left).



- These five points are transformed to five output points (right frame).
- We see that the mean and covariance of the output sigma points (dashed ellipse) closely match the true mean and covariance.
- Will the sigma-point method always be so much better than EKF?
  - The answer depends on the degree of nonlinearity of the state and output equations—the more nonlinear the better SPKF should be with respect to EKF.

6.5: The SPKF Steps

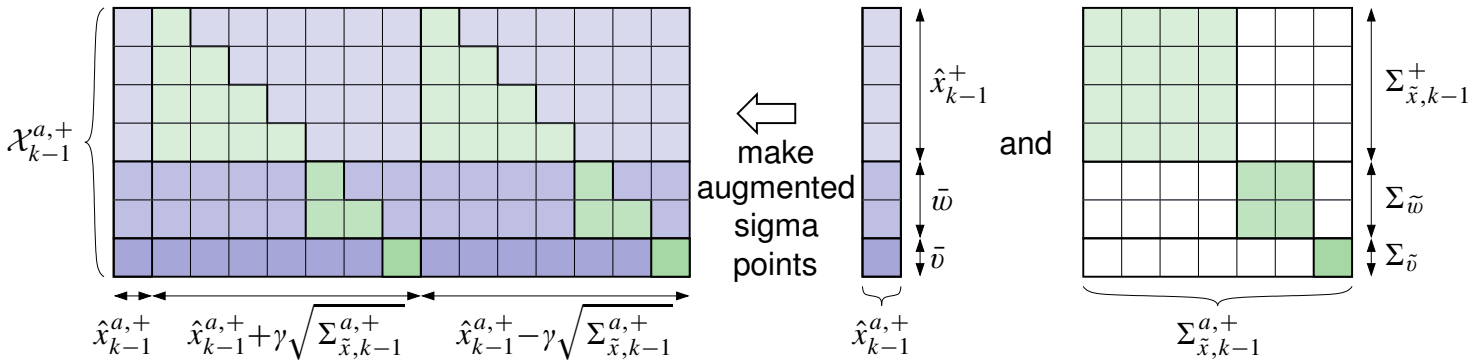
- We now apply the sigma-point approach of propagating statistics through a nonlinear function to the state-estimation problem.
- These sigma-points must jointly model all randomness: uncertainty of the state, process noise, and sensor noise.
- So we first define an augmented random vector x_k^a that combines these random factors at time index k.
- This augmented vector is used in the estimation process as described below.

SPKF step 1a: State estimate time update.

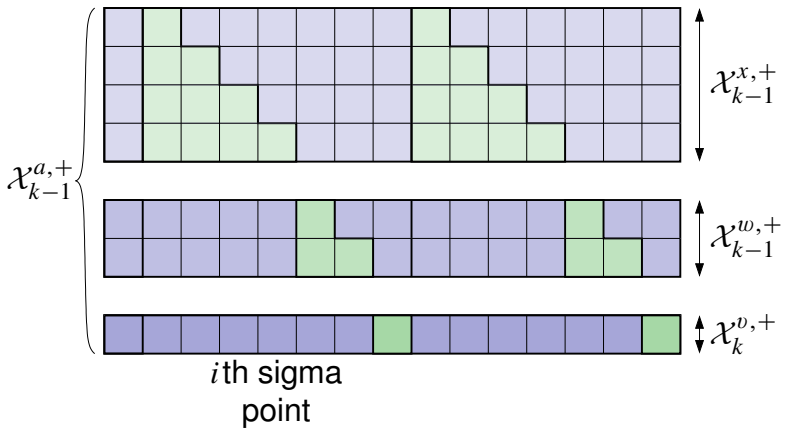
- First, form the augmented *a posteriori* state estimate vector for the previous time interval: $\hat{x}_{k-1}^{a,+} = \left[(\hat{x}_{k-1}^{+})^T, \bar{w}, \bar{v}\right]^T$, and the augmented *a posteriori* covariance estimate: $\Sigma_{\tilde{x},k-1}^{a,+} = \mathrm{diag}\left(\Sigma_{\tilde{x},k-1}^{+}, \Sigma_{\tilde{w}}, \Sigma_{\tilde{v}}\right)$.
- These factors are used to generate the $p+1$ augmented sigma points

$$\mathcal{X}_{k-1}^{a,+} = \left\{ \hat{x}_{k-1}^{a,+},\ \hat{x}_{k-1}^{a,+} + \gamma\sqrt{\Sigma_{\tilde{x},k-1}^{a,+}},\ \hat{x}_{k-1}^{a,+} - \gamma\sqrt{\Sigma_{\tilde{x},k-1}^{a,+}} \right\}.$$

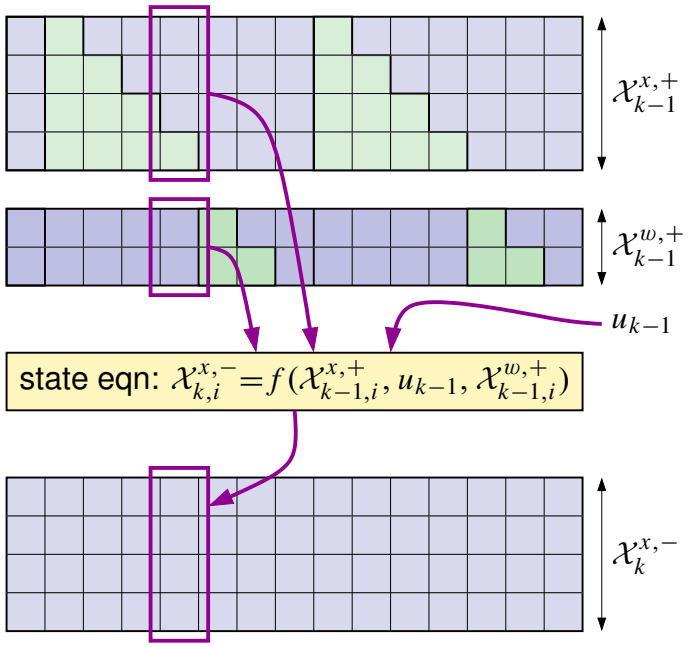
- Can be organized in convenient matrix form:



- Split augmented sigma points $\mathcal{X}_{k-1}^{a,+}$ into state portion $\mathcal{X}_{k-1}^{x,+}$, process-noise portion $\mathcal{X}_{k-1}^{w,+}$, and sensor-noise portion $\mathcal{X}_{k}^{v,+}$.



- Evaluate state equation using all pairs of $\mathcal{X}_{k-1,i}^{x,+}$ and $\mathcal{X}_{k-1,i}^{w,+}$ (where subscript i denotes that the ith vector is being extracted from the original set), yielding the *a priori* sigma points $\mathcal{X}_{k,i}^{x,-}$.
- That is, compute $\mathcal{X}_{k,i}^{x,-} = f(\mathcal{X}_{k-1,i}^{x,+}, u_{k-1}, \mathcal{X}_{k-1,i}^{w,+})$.

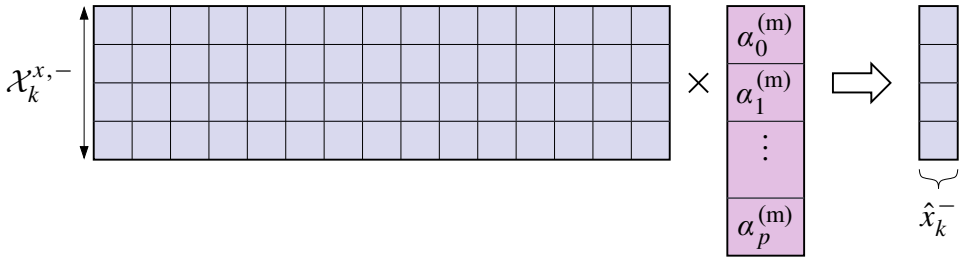


- Finally, the *a priori* state estimate is computed as

$$\hat{x}_k^- = \mathbb{E}\big[f(x_{k-1}, u_{k-1}, w_{k-1}) \mid \mathbb{Z}_{k-1}\big] \approx \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} f(\mathcal{X}_{k-1,i}^{x,+}, u_{k-1}, \mathcal{X}_{k-1,i}^{w,+})$$

$$= \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \mathcal{X}_{k,i}^{x,-}.$$

- Can be computed with a simple matrix multiply operation.



SPKF step 1b: Error covariance time update.

- Using the *a priori* sigma points from step 1a, the *a priori* covariance estimate is computed as

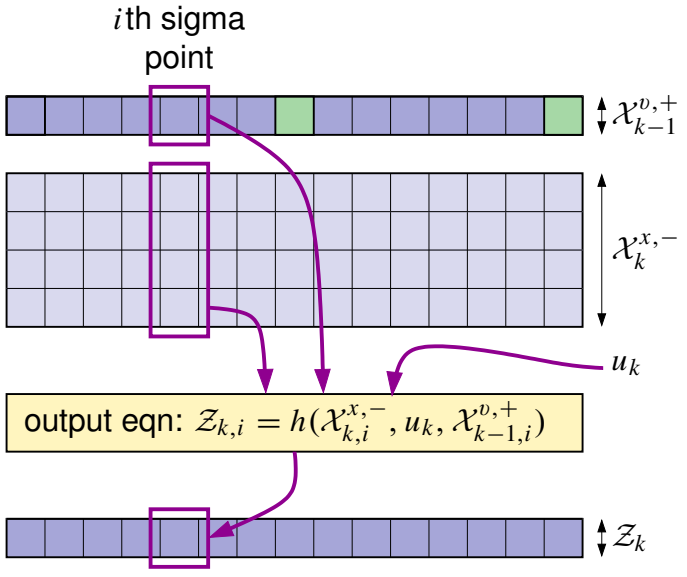
$$\Sigma^{-}_{\tilde{x},k} = \sum_{i=0}^{p} \alpha_i^{(c)} \left(\mathcal{X}^{x,-}_{k,i} - \hat{x}_k^-\right)\left(\mathcal{X}^{x,-}_{k,i} - \hat{x}_k^-\right)^T.$$

- To compute, recognize (noting that $\alpha_0^{(c)}$ may be negative)

$$\begin{aligned}\Sigma^{-}_{\tilde{x},k} &= \alpha_0^{(c)} \left(\mathcal{X}^{x,-}_{k,0} - \hat{x}_k^-\right)\left(\mathcal{X}^{x,-}_{k,0} - \hat{x}_k^-\right)^T \\ &\quad + \sum_{i=1}^{p} \underbrace{\left[\sqrt{\alpha_i^{(c)}}\left(\mathcal{X}^{x,-}_{k,i} - \hat{x}_k^-\right)\right]}_{\mathcal{X}_{\alpha,i}} \underbrace{\left[\sqrt{\alpha_i^{(c)}}\left(\mathcal{X}^{x,-}_{k,i} - \hat{x}_k^-\right)^T\right]}_{\mathcal{X}^T_{\alpha,i}} \\ &= \alpha_0^{(c)} \left(\mathcal{X}^{x,-}_{k,0} - \hat{x}_k^-\right)\left(\mathcal{X}^{x,-}_{k,0} - \hat{x}_k^-\right)^T + \mathcal{X}_\alpha \mathcal{X}_\alpha^T.\end{aligned}$$

SPKF step 1c: Estimate system output z_k.

- The output z_k is estimated by evaluating the model output equation using the sigma points describing the state and sensor noise.
- First, we compute the points $\mathcal{Z}_{k,i} = h(\mathcal{X}^{x,-}_{k,i}, u_k, \mathcal{X}^{v,+}_{k-1,i})$.



- The output estimate is then

$$\hat{z}_k = \mathbb{E}\left[h(x_k, u_k, v_k) \mid \mathbb{Z}_{k-1}\right] \approx \sum_{i=0}^{p} \alpha_i^{(m)} h(\mathcal{X}^{x,-}_{k,i}, u_k, \mathcal{X}^{v,+}_{k-1,i}) = \sum_{i=0}^{p} \alpha_i^{(m)} \mathcal{Z}_{k,i}.$$

- This can be computed with a simple matrix multiplication, as we did when calculating $\hat{x}_k^-$ at the end of step 1a.

SPKF step 2a: Estimator gain matrix L_k.

- To compute the estimator gain matrix, we must first compute the required covariance matrices.

$$\Sigma_{\tilde{z},k} = \sum_{i=0}^{p} \alpha_i^{(c)} \left(\mathcal{Z}_{k,i} - \hat{z}_k\right)\left(\mathcal{Z}_{k,i} - \hat{z}_k\right)^T$$

$$\Sigma_{\tilde{x}\tilde{z},k}^{-} = \sum_{i=0}^{p} \alpha_i^{(c)} \left(\mathcal{X}_{k,i}^{x,-} - \hat{x}_k^{-}\right)\left(\mathcal{Z}_{k,i} - \hat{z}_k\right)^T.$$

- These depend on the sigma-point matrices $\mathcal{X}_k^{x,-}$ and $\mathcal{Z}_k$, already computed in steps 1b and 1c, as well as $\hat{x}_k^-$ and $\hat{z}_k$, already computed in steps 1a and 1c.
- The summations can be performed using matrix multiplies, as we did in step 1b.
- Then, we simply compute $L_k = \Sigma_{\tilde{x}\tilde{z},k}^{-} \Sigma_{\tilde{z},k}^{-1}$.

SPKF step 2b: State estimate measurement update.

- The state estimate is computed as

$$\hat{x}_k^+ = \hat{x}_k^- + L_k(z_k - \hat{z}_k).$$

SPKF step 2c: Error covariance measurement update.

- The final step is calculated directly from the optimal formulation: $\Sigma_{\tilde{x},k}^{+} = \Sigma_{\tilde{x},k}^{-} - L_k \Sigma_{\tilde{z},k} L_k^T$.

Summary of the nonlinear sigma-point Kalman filter

Nonlinear state-space model:

$$x_k = f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})$$
$$z_k = h_k(x_k, u_k, v_k),$$

where w_k and v_k are independent, Gaussian noise processes with means $\bar{w}$ and $\bar{v}$ and covariance matrices $\Sigma_{\tilde{w}}$ and $\Sigma_{\tilde{v}}$, respectively.

Definitions: Let

$$x_k^a = \left[x_k^T,\ w_k^T,\ v_k^T\right]^T, \quad \mathcal{X}_k^a = \left[(\mathcal{X}_k^x)^T,\ (\mathcal{X}_k^w)^T,\ (\mathcal{X}_k^v)^T\right]^T, \quad p = 2 \times \dim(x_k^a).$$

Initialization: For $k = 0$, set

$$\hat{x}_0^+ = \mathbb{E}\left[x_0\right] \qquad \Sigma_{\tilde{x},0}^+ = \mathbb{E}\left[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T\right]$$

$$\hat{x}_0^{a,+} = \mathbb{E}\left[x_0^a\right] = \left[(\hat{x}_0^+)^T,\ \bar{w},\ \bar{v}\right]^T \qquad \Sigma_{\tilde{x},0}^{a,+} = \mathbb{E}\left[(x_0^a - \hat{x}_0^{a,+})(x_0^a - \hat{x}_0^{a,+})^T\right]$$
$$= \mathrm{diag}\left(\Sigma_{\tilde{x},0}^+,\ \Sigma_{\tilde{w}},\ \Sigma_{\tilde{v}}\right).$$

(continued...)

Summary of the nonlinear sigma-point Kalman filter (cont.)

Computation: For $k = 1, 2, \ldots$ compute:

State estimate time update:

$$\mathcal{X}_{k-1}^{a,+} = \left\{ \hat{x}_{k-1}^{a,+},\ \hat{x}_{k-1}^{a,+} + \gamma \sqrt{\Sigma_{\tilde{x},k-1}^{a,+}},\ \hat{x}_{k-1}^{a,+} - \gamma \sqrt{\Sigma_{\tilde{x},k-1}^{a,+}} \right\}.$$

$$\mathcal{X}_{k,i}^{x,-} = f_{k-1}(\mathcal{X}_{k-1,i}^{x,+}, u_{k-1}, \mathcal{X}_{k-1,i}^{w,+}).$$

$$\hat{x}_k^- = \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \mathcal{X}_{k,i}^{x,-}.$$

Error covariance time update:

$$\Sigma_{\tilde{x},k}^- = \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} \left(\mathcal{X}_{k,i}^{x,-} - \hat{x}_k^-\right)\left(\mathcal{X}_{k,i}^{x,-} - \hat{x}_k^-\right)^T.$$

Output estimate:

$$\mathcal{Z}_{k,i} = h_k(\mathcal{X}_{k,i}^{x,-}, u_k, \mathcal{X}_{k-1,i}^{v,+}).$$

$$\hat{z}_k = \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \mathcal{Z}_{k,i}.$$

Estimator gain matrix:

$$\Sigma_{\tilde{z},k} = \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} \left(\mathcal{Z}_{k,i} - \hat{z}_k\right)\left(\mathcal{Z}_{k,i} - \hat{z}_k\right)^T.$$

$$\Sigma_{\tilde{x}\tilde{z},k}^- = \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} \left(\mathcal{X}_{k,i}^{x,-} - \hat{x}_k^-\right)\left(\mathcal{Z}_{k,i} - \hat{z}_k\right)^T.$$

$$L_k = \Sigma_{\tilde{x}\tilde{z},k}^- \Sigma_{\tilde{z},k}^{-1}.$$

State estimate meas. update:

$$\hat{x}_k^+ = \hat{x}_k^- + L_k\left(z_k - \hat{z}_k\right).$$

Error covariance meas. update:

$$\Sigma_{\tilde{x},k}^+ = \Sigma_{\tilde{x},k}^- - L_k \Sigma_{\tilde{z},k} L_k^T.$$

6.6: An SPKF example, with code

- Consider the same example used to illustrate EKF:

$$x_{k+1} = \sqrt{5 + x_k} + w_k$$

$$z_k = x_k^3 + v_k$$

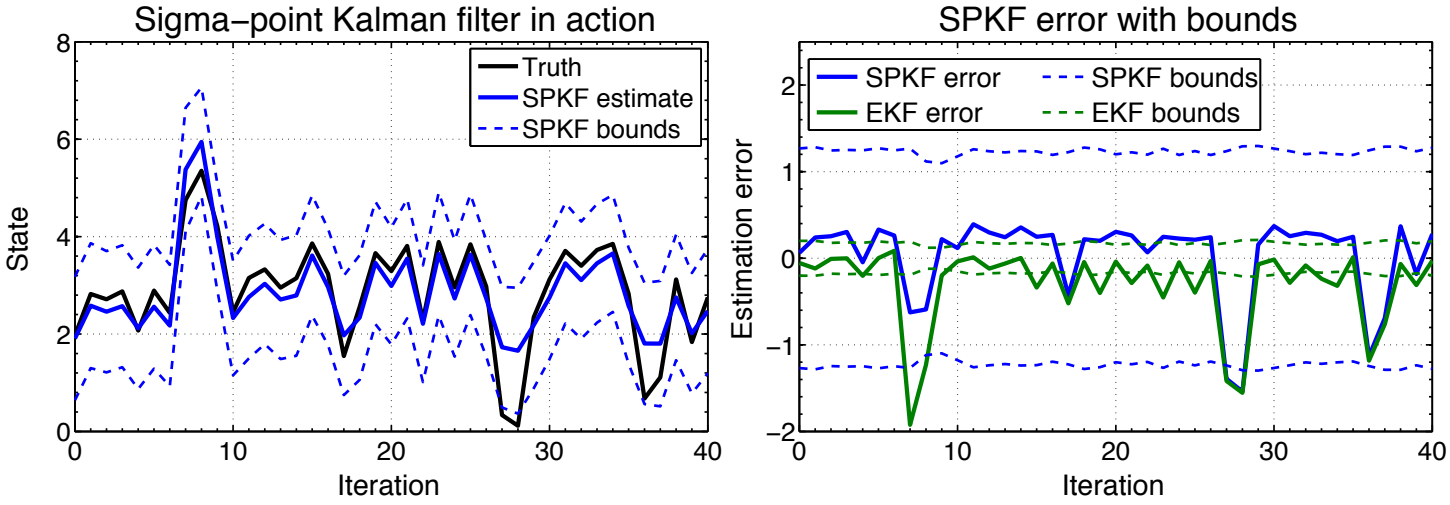
```
% Define size of variables in model
Nx = 1;      % state  = 1x1 scalar
Nxa = 3;     % augmented state has also w(k) and v(k) contributions
Nz = 1;      % output = 1x1 scalar

% Some constants for the SPKF algorithm. Use standard values for
% cases with Gaussian noises.  (These are the weighting matrices
% comprising the values of alpha(c) and alpha(m) organized in a way to
% make later computation efficient).
h = sqrt(3);
Wmx(1) = (h*h-Nxa)/(h*h); Wmx(2) = 1/(2*h*h); Wcx=Wmx;
Wmxz = [Wmx(1) repmat(Wmx(2),[1 2*Nxa])]';

% Initialize simulation variables
SigmaW = 1; % Process noise covariance
SigmaV = 2; % Sensor noise covariance
maxIter = 40;
xtrue = 2 + randn(1);  % Initialize true system initial state
xhat = 2;              % Initialize Kalman filter initial estimate
SigmaX = 1;          % Initialize Kalman filter covariance

% Reserve storage for variables we might want to plot/evaluate
xstore = zeros(maxIter+1,length(xtrue)); xstore(1,:) = xtrue;
xhatstore = zeros(maxIter,length(xhat));
SigmaXstore = zeros(maxIter,length(xhat)^2);

for k = 1:maxIter,
  % SPKF Step 1a: State estimate time update
  % 1a-i: Calculate augmented state estimate, including ...
  xhata = [xhat; 0; 0]; % process and sensor noise mean
  % 1a-ii: Get desired Cholesky factor
  Pxa = blkdiag(SigmaX,SigmaW,SigmaV);
  sPxa = chol(Pxa,'lower');
  % 1a-iii: Calculate sigma points (strange indexing of xhat to avoid
```



```
  % "repmat" call, which is very inefficient in Matlab)
  X = xhata(:,ones([1 2*Nxa+1])) + h*[zeros([Nxa 1]), sPxa, -sPxa];
  % 1a-iv: Calculate state equation for every element
  % Hard-code equation here for efficiency
  Xx = sqrt(5+X(1,:)) + X(2,:);
  xhat = Xx*Wmxz;

  % SPKF Step 1b: Covariance of prediction
  Xs = (Xx(:,2:end) - xhat(:,ones([1 2*Nxa])))*sqrt(Wcx(2));
  Xs1 = Xx(:,1) - xhat;
  SigmaX = Xs*Xs' + Wcx(1)*Xs1*Xs1';

  % [Implied operation of system in background, with
  % input signal u, and output signal z]
  w = chol(SigmaW)'*randn(1);
  v = chol(SigmaV)'*randn(1);
  ztrue = xtrue^3 + v;  % z is based on present x and u
  xtrue = sqrt(5+xtrue) + w;  % future x is based on present u

  % SPKF Step 1c: Create output estimate
  % Hard-code equation here for efficiency
  Z = Xx.^3 + X(3,:);
  zhat = Z*Wmxz;

  % SPKF Step 2a: Estimator gain matrix
  Zs = (Z(:,2:end) - zhat*ones([1 2*Nxa])) * sqrt(Wcx(2));
  Zs1 = Z(:,1) - zhat;
  SigmaXZ = Xs*Zs' + Wcx(1)*Xs1*Zs1';
  SigmaZ  = Zs*Zs' + Wcx(1)*Zs1*Zs1';
  Lx= SigmaXZ/SigmaZ;

  % SPKF Step 2b: Measurement state update
  xhat = xhat + Lx*(ztrue-zhat); % update prediction to estimate

  % SPKF Step 2c: Measurement covariance update
  SigmaX = SigmaX - Lx*SigmaZ*Lx';

  % [Store information for evaluation/plotting purposes]
  xstore(k+1,:) = xtrue;
  xhatstore(k,:) = xhat;
  SigmaXstore(k,:) = SigmaX(:);
end
```



```
figure(1); clf;
plot(0:maxIter-1,xstore(1:maxIter),'k-',0:maxIter-1,xhatstore,'b--', ...
  0:maxIter-1,xhatstore+3*sqrt(SigmaXstore),'m-.',...
  0:maxIter-1,xhatstore-3*sqrt(SigmaXstore),'m-.'); grid;
legend('true','estimate','bounds'); xlabel('Iteration'); ylabel('State');
title('Sigma-point Kalman filter in action');

figure(2); clf;
plot(0:maxIter-1,xstore(1:maxIter)-xhatstore,'b-',0:maxIter-1, ...
  3*sqrt(SigmaXstore),'m--',0:maxIter-1,-3*sqrt(SigmaXstore),'m--');
grid; legend('Error','bounds',0);
title('SPKF Error with bounds');
xlabel('Iteration'); ylabel('Estimation Error');
```



- Note improved estimation accuracy, greatly improved error bounds.

6.7: Efficient square-root sigma-point Kalman filters

- Sigma-point Kalman filters generally produce better state estimates and *much* better covariance estimates than EKF.
- The computational complexity is $\mathcal{O}(L^3)$, where L is the dimension of the augmented state (equivalent to EKF).
- They may also be used for parameter estimation, as will be described in a later chapter, but the computational complexity remains $\mathcal{O}(L^3)$, whereas the corresponding EKF method has complexity $\mathcal{O}(L^2)$.
- The bottleneck in the SPKF algorithm is the computation of the matrix square root $\mathcal{S}_k\mathcal{S}_k^T = \Sigma_k$ each time step, which has computational complexity $\mathcal{O}(L^3/6)$ using a Cholesky factorization.
- A variant of the SPKF, the square-root SPKF (SR-SPKF), propagates $\mathcal{S}_k$ directly without needing to re-factor each time step.
- This approach has several advantages: there are improved numeric properties as the covariances are guaranteed to be positive semi-definite, and although the state-estimation update is still $\mathcal{O}(L^3)$, the parameter update may now be done in $\mathcal{O}(L^2)$. Therefore, for the same computational complexity of EKF, better results are obtained.
- The SR-SPKF is described below. There are many similarities to the SR-KF we saw in notes chapter 5.

SR-SPKF step 1a: State estimate time update.

- As with SPKF, each measurement interval, the state estimate time update is computed by first forming the augmented *a posteriori* state estimate vector for the previous time interval:

$$\hat{x}_{k-1}^{a,+} = \left[(\hat{x}_{k-1}^{+})^T, \bar{w}, \bar{v}\right]^T.$$

- With SR-SPKF, the square-root augmented *a posteriori* covariance estimate is computed:

$$\mathcal{S}^{a,+}_{\tilde{x},k-1} = \text{diag}\left(\mathcal{S}^{+}_{\tilde{x},k-1}, \mathcal{S}_{\tilde{w}}, \mathcal{S}_{\tilde{v}}\right).$$

- These factors are used to generate the $p+1$ sigma points:

$$\mathcal{X}^{a,+}_{k-1} = \left\{\hat{x}^{a,+}_{k-1},\ \hat{x}^{a,+}_{k-1} + \gamma\,\mathcal{S}^{a,+}_{\tilde{x},k-1},\ \hat{x}^{a,+}_{k-1} - \gamma\,\mathcal{S}^{a,+}_{\tilde{x},k-1}\right\}.$$

- From the augmented sigma points, the $p+1$ vectors comprising the state portion $\mathcal{X}^{x,+}_{k-1}$ and the $p+1$ vectors comprising the process-noise portion $\mathcal{X}^{w,+}_{k-1}$ are extracted.

- The state equation is evaluated using all pairs of $\mathcal{X}^{x,+}_{k-1,i}$ and $\mathcal{X}^{w,+}_{k-1,i}$, yielding the *a priori* sigma points $\mathcal{X}^{x,-}_{k,i}$ for time step k.

- Finally, the *a priori* state estimate is computed as $\hat{x}^{-}_{k} = \sum_{i=0}^{p} \alpha^{(\text{m})}_{i}\,\mathcal{X}^{x,-}_{k,i}$.

SR-SPKF step 1b: Error covariance time update.

- Note that we want to compute the square root of $\Sigma^{-}_{\tilde{x},k}$, where we recall

$$\Sigma^{-}_{\tilde{x},k} = \sum_{i=0}^{p} \alpha^{(\text{c})}_{i}\left(\mathcal{X}^{x,-}_{k,i} - \hat{x}^{-}_{k}\right)\left(\mathcal{X}^{x,-}_{k,i} - \hat{x}^{x}_{k,i}\right)^{T},$$

  which may also be written as $\Sigma^{-}_{\tilde{x},k} = AA^{T}$ where

$$A = \left[\sqrt{\alpha^{(\text{c})}_{i}}\left(\mathcal{X}^{x,-}_{k,(0:p)} - \hat{x}^{-}_{k}\right)\right].$$

- Using the *a priori* sigma points from step 1a, the square-root *a priori* covariance estimate is computed as

$$\mathcal{S}^{-}_{\tilde{x},k} = \text{qr}\left\{\left[\sqrt{\alpha^{(\text{c})}_{i}}\left(\mathcal{X}^{x,-}_{k,(0:p)} - \hat{x}^{-}_{k}\right)^{T}\right]\right\}^{T}.$$

SR-SPKF step 1c: Estimate system output z_k.

- The output z_k is estimated by evaluating the model output equation using the sigma points describing the state and sensor noise.
- First, we compute the points $\mathcal{Z}_{k,i} = h_k(\mathcal{X}^{x,-}_{k,i}, u_k, \mathcal{X}^{v,+}_{k-1,i})$.
- The output estimate is then $\hat{z}_k = \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \mathcal{Z}_{k,i}$.

SR-SPKF step 2a: Estimator gain matrix L_k.

- To compute the estimator gain matrix, we must first compute the required covariance and square-root covariance matrices.

$$\mathcal{S}_{\tilde{z},k} = \mathrm{qr}\left\{\left[\sqrt{\alpha_i^{(\mathrm{c})}}\left(\mathcal{Z}_{k,(0:p)} - \hat{z}_k\right)^T\right]\right\}^T$$

$$\Sigma^{-}_{\tilde{x}\tilde{z},k} = \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})}\left(\mathcal{X}^{x,-}_{k,i} - \hat{x}^{-}_k\right)\left(\mathcal{Z}_{k,i} - \hat{z}_k\right)$$

- Then, we simply compute $L_k = \Sigma^{-}_{\tilde{x}\tilde{z},k}\Sigma^{-1}_{\tilde{z},k}$, solved by backsubstitution.

SR-SPKF step 2b: State estimate measurement update.

- The state estimate is computed as

$$\hat{x}^{+}_k = \hat{x}^{-}_k + L_k(z_k - \hat{z}_k).$$

SR-SPKF step 2c: Error covariance measurement update.

- The final step computes the square-root form of

$$\Sigma^{+}_{\tilde{x},k} = \Sigma^{-}_{\tilde{x},k} - L_k\Sigma_{\tilde{z},k}L_k^T$$

as

$$\mathcal{S}^{+}_{\tilde{x},k} = \texttt{cholupdate}\left(\left(S^{-}_{\tilde{x},k}\right)^T, \left(L_k\mathcal{S}_{\tilde{z},k}\right)^T, \text{'-'}\right)^T.$$

Summary of the square-root sigma-point Kalman filter for state estimation.

Nonlinear state-space model:

$$x_k = f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})$$
$$z_k = h_k(x_k, u_k, v_k),$$

where w_k and v_k are independent Gaussian noise processes with means $\bar{w}$ and $\bar{v}$ and covariance matrices $\Sigma_{\tilde{w}}$ and $\Sigma_{\tilde{v}}$, respectively.

Definitions: Let

$$x_k^a = \left[x_k^T,\ w_k^T,\ v_k^T\right]^T, \quad \mathcal{X}_k^a = \left[(\mathcal{X}_k^x)^T,\ (\mathcal{X}_k^w)^T,\ (\mathcal{X}_k^v)^T\right]^T, \quad p = 2 \times \dim(x_k^a).$$

Initialization: For $k = 0$, set

$$\hat{x}_0^+ = \mathbb{E}\left[x_0\right] \qquad \hat{x}_0^{a,+} = \mathbb{E}\left[x_0^a\right] = \left[(\hat{x}_0^+)^T,\ \bar{w},\ \bar{v}\right]^T.$$

$$S_{\tilde{x},0}^+ = \text{chol}\left\{\mathbb{E}\left[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T\right]\right\} \qquad S_{\tilde{x},0}^{a,+} = \text{chol}\left\{\mathbb{E}\left[(x_0^a - \hat{x}_0^{a,+})(x_0^a - \hat{x}_0^{a,+})^T\right]\right\}$$
$$= \text{diag}\left(\mathcal{S}_{\tilde{x},0}^+, \mathcal{S}_{\tilde{w}}, \mathcal{S}_{\tilde{v}}\right).$$

(continued...)

Summary of the square-root SPKF for state estimation (continued).

Computation: For $k = 1, 2, \ldots$ compute:

State estimate time update:

$$\mathcal{X}_{k-1}^{a,+} = \left\{ \hat{x}_{k-1}^{a,+},\ \hat{x}_{k-1}^{a,+} + \gamma\, \mathcal{S}_{\tilde{x},k-1}^{a,+},\ \hat{x}_{k-1}^{a,+} - \gamma\, \mathcal{S}_{\tilde{x},k-1}^{a,+} \right\}.$$

$$\mathcal{X}_{k,i}^{x,-} = f_{k-1}(\mathcal{X}_{k-1,i}^{x,+}, u_{k-1}, \mathcal{X}_{k-1,i}^{w,+}).$$

$$\hat{x}_k^- = \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \mathcal{X}_{k,i}^{x,-}.$$

Error covariance time update:

$$\mathcal{S}_{\tilde{x},k}^- = \mathrm{qr}\left\{ \left[\sqrt{\alpha_i^{(\mathrm{c})}} \left(\mathcal{X}_{k,(0:p)}^{x,-} - \hat{x}_k^- \right)^T \right] \right\}^T.$$

Output estimate:

$$\mathcal{Z}_{k,i} = h_k(\mathcal{X}_{k,i}^{x,-}, u_k, \mathcal{X}_{k-1,i}^{v,+}).$$

$$\hat{z}_k = \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \mathcal{Z}_{k,i}.$$

Estimator gain matrix:

$$\mathcal{S}_{\tilde{z},k} = \mathrm{qr}\left\{ \left[\sqrt{\alpha_i^{(\mathrm{c})}} \left(\mathcal{Z}_{k,(0:p)} - \hat{z}_k \right)^T \right] \right\}^T.$$

$$\Sigma_{\tilde{x}\tilde{z},k}^- = \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} \left(\mathcal{X}_{k,i}^{x,-} - \hat{x}_k^- \right) \left(\mathcal{Z}_{k,i} - \hat{z}_k \right)^T.$$

$$L_k = \Sigma_{\tilde{x}\tilde{z},k}^- \left(\mathcal{S}_{\tilde{z},k}^T \mathcal{S}_{\tilde{z},k} \right)^{-1}.$$

(L_k is computed via backsubstitution).

State estimate meas. update:

$$\hat{x}_k^+ = \hat{x}_k^- + L_k \left(z_k - \hat{z}_k \right).$$

Error covariance meas. update:

$$\mathcal{S}_{\tilde{x},k}^+ = \texttt{cholupdate}\left(\left(S_{\tilde{x},k}^- \right)^T, \left(L_k \mathcal{S}_{\tilde{z},k} \right)^T, \text{'-'} \right)^T.$$

