Real-world issue: Sensor faults

- Sometimes systems for which we would like a state estimate use sensors having intermittent faults.
- We would like to detect faulty measurements and discard them.
  - Time update steps of the KF still implemented.
  - Measurement update steps are skipped ($L_k = 0$).
- KF provides elegant theoretical means to accomplish this goal. Background:
  - Predicted measurement is $\hat{z}_k = C_k \hat{x}_k^- + D_k u_k$.
  - Prediction covariance (uncertainty) matrix is $\Sigma_{\tilde{z},k} = C_k \Sigma_{\tilde{x},k}^- C_k^T + \Sigma_{\tilde{v}}$.
  - The innovation is $\tilde{z}_k = z_k - \hat{z}_k$.
- By combining $\tilde{z}_k$ and $\Sigma_{\tilde{z},k}$ we can determine if the innovation is “too big,” which indicates a possible sensor fault.

Measurement validation gating

- Can place “measurement validation gate” on measurement using normalized estimation error squared (NEES):

$$e_k^2 = \tilde{z}_k^T \Sigma_{\tilde{z},k}^{-1} \tilde{z}_k.$$

- NEES e_k^2 has Chi-squared distribution with m degrees of freedom, where $z_k \in \mathbb{R}^m$.
- If e_k^2 is outside of bounding value for Chi-squared distribution for a desired confidence level, then measurement is discarded.
- Note: If a many measurements are discarded in a short time interval, the sensor may truly have failed, or the state estimate and covariance may have gotten “lost.”
- It is sometimes helpful to “bump up” covariance $\Sigma_{\tilde{x},k}^{\pm}$, which simulates additional process noise, to help Kalman filter to reacquire.
- Both done in practice to aid robustness of a real implementation.

NEES is chi-squared

- To prove NEES is chi-squared, define $y_k = M_k \tilde{z}_k$.
  - Mean of y_k is $\mathbb{E}[y_k] = \mathbb{E}[M_k \tilde{z}_k] = 0$.
  - Covariance of y_k is $\Sigma_{\tilde{y},k} = \mathbb{E}[M_k \tilde{z}_k \tilde{z}_k^T M_k^T] = M_k \Sigma_{\tilde{z},k} M_k^T$.
  - y_k is Gaussian (since it is a linear combination of Gaussians).
- If we define M_k such that $M_k^T M_k = \Sigma_{\tilde{z},k}^{-1}$, then:
  - M_k is the lower-triangular Cholesky factor of $\Sigma_{\tilde{z},k}^{-1}$.
  - Also, $y_k \sim \mathcal{N}(0, I)$ since:

$$\begin{aligned} \Sigma_{\tilde{y},k} &= M_k \left(M_k^T M_k\right)^{-1} M_k^T \\ &= M_k M_k^{-1} M_k^{-T} M_k^T = I. \end{aligned}$$

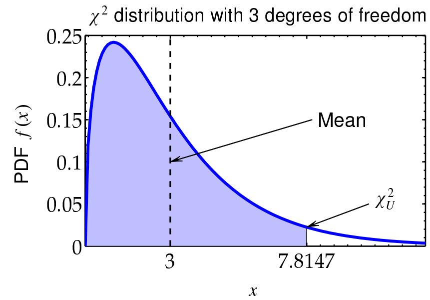
- NEES $e_k^2 = y_k^T y_k = \tilde{z}_k^T \Sigma_{\tilde{z},k}^{-1} \tilde{z}_k$ is the sum of squares of independent $\mathcal{N}(0, 1)$ RVs.
- So, e_k^2 is chi-square with m degrees of freedom, where m is the dimension of $\tilde{z}_k$.

What does this really mean?

- e_k^2 never negative (sum of squares); pdf also asymmetric.
- pdf of chi-square RV X having m degrees of freedom is:

$$f_X(x) = \frac{1}{2^{m/2}\Gamma(m/2)} x^{(m/2-1)} e^{-m/2}.$$

  - Tricky, but don't need to evaluate in real time.
- Instead, use value *precomputed* from pdf.
- For $1-\alpha$ confidence of a valid measurement, need to find χ_U^2 such that there is α area above χ_U^2 (figure drawn for $\alpha = 0.05$).
- Discard measurement if NEES greater than χ_U^2.



Computer calculation of χ_U^2

- In MATLAB (Statistics and Machine Learning Toolbox) can find χ_U^2 where inverse CDF is equal to $1-\alpha$:

```
X2U = chi2inv(1-0.01,2) % Upper critical value X2U = 9.2103
```

  - Function "chi2inv" is built in to Octave.
- Note that χ_U^2 needs to be computed once only, offline.
  - Based only on m and α, so doesn't need to be recalculated as KF runs.
- For hand calculations a χ^2-table is available on next page.
- If $e_k^2 > \chi_U^2$, then measurement is discarded ($L_k = 0$); else, measurement kept.

Manual table-lookup of χ_U^2

- For chi-squared distribution with m degrees of freedom, table entries list values of $\chi_U^2(\alpha, m)$ for specified upper tail area α:

Degrees of	Upper tail areas α					
freedom m	0.25	0.10	0.05	0.025	0.01	0.005
1	1.323	2.706	3.841	5.024	6.635	7.879
2	2.773	4.605	5.991	7.378	9.210	10.597
3	4.108	6.251	7.815	9.348	11.345	12.838
4	5.385	7.779	9.488	11.143	13.277	14.860
5	6.626	9.236	11.070	12.833	15.086	16.750
6	7.841	10.645	12.592	14.449	16.812	18.548

Integration into the Kalman filter

```
% KF Step 1c: Predict system output
zhat = Cd*xhat + Dd*u(:,k);
zerror = z(:,k) - zhat;
SigmaZ = C*SigmaX*C' + SigmaV;
nees = zerror'/SigmaZ*zerror;

% KF validation gate (X2U can be calculated outside of loop)
alpha = 0.01; confidence = 1 - alpha;
X2U = chi2inv(confidence,length(zhat)); % Upper critical value

if nees <= X2U
  % KF Step 2a: Compute Kalman gain matrix
  L = SigmaX*C'/SigmaZ;

  % KF Step 2b: State estimate measurement update
  xhat = xhat + L*zerror;

  % KF Step 2c: Estimation-error covariance measurement update
  SigmaX = SigmaX - L*C*SigmaX;
end
```


Summary

- KF has built-in mechanism that enables detecting sensor errors.
- Once only, off-line, precompute $\chi^2_U(\alpha, m)$ for $z_k \in \mathbb{R}^m$ and desired α.
- As KF executes, every time sample, compute $e_k^2 = \tilde{z}_k^T \Sigma_{\tilde{z},k}^{-1} \tilde{z}_k$.
  - If $e_k^2 > \chi^2_U(\alpha, m)$, then discard measurement (set $L_k = 0$).
  - Otherwise, apply measurement update as usual.
- If many sequential measurements discarded, consider "bumping up" covariance as $\Sigma^+_{\tilde{x},k} = Q\Sigma^+_{\tilde{x},k}$ where $Q > 1$.
- If problems persist, likely a permanent sensor fault.