INVESTIGATION OF DIFFERENT EXTENDED KALMAN FILTER IMPLEMENTATIONS

Mitch Serpas
Juergen Hahn
OVERVIEW

- Introduction
  - Use of Estimators for Process Safety
  - Nonlinear filters
- Background
  - Extended Kalman Filter as a standard
- Method
  - Comparison of EKF Implementation Algorithms
  - Case Studies
- Results and Conclusions
USE OF ESTIMATORS FOR PROCESS SAFETY

- Enable fault detection of unmeasured process variables
FILTER TYPES

- **Kalman Filter**
  - Provides optimal solution
  - For linear systems only

- **Extended Kalman Filter (EKF)**
  - Uses Kalman Filter on linearized version of system
  - One of the most-widely used filters; often used as benchmark

- **Other Nonlinear Filters**
  - Unscented Kalman Filter
  - Moving Horizon Estimator
  - Particle Filter
LINEAR SYSTEMS: A SIMPLIFICATION

\[
\dot{x} = Ax + Bu + Gw \\
y = Cx + Du + v
\]

- \( x \) – system states
- \( u \) – input
- \( w \) – state model noise
- \( y \) – output
- \( v \) – measurement noise
- \( A, B, G, C, D \) – linear model parameters
KALMAN FILTER

- Optimal estimator for linear systems only

**Time Update (“Predict”)**

1. Project the state ahead
   \[ \hat{x}_k^- = A \hat{x}_{k-1} + Bu_{k-1} \]
2. Project the error covariance ahead
   \[ P_k^- = AP_{k-1}A^T + Q \]

**Measurement Update (“Correct”)**

1. Compute the Kalman gain
   \[ K_k = P_k^- HT (HP_k HT + R)^{-1} \]
2. Update estimate with measurement \( z_k \)
   \[ \hat{x}_k = \hat{x}_k^- + K_k (z_k - H \hat{x}_k^-) \]
3. Update the error covariance
   \[ P_k = (I - K_k H) P_k^- \]

Initial estimates for \( \hat{x}_{k-1} \) and \( P_{k-1} \)
EXTENDED KALMAN FILTER

- Nonlinear System Equations:

\[ \dot{x}(t) = f(x(t), u(t)) + Gw(t) \]
\[ y(t_k) = h(x(t_k)) + v_k \]
\[ w(t) \sim N(0, Q) \]
\[ v_k \sim N(0, R) \]
EXTENDED KALMAN FILTER

○ Linearization:

\[ \dot{x}(t) = f(x(t), u(t), w(t)) \]
\[ y(t_k) = h(x(t_k), v(t_k)) \]

\[ A = \left. \frac{\partial f}{\partial x} \right|_{\dot{x}} \quad C = \left. \frac{\partial h}{\partial x} \right|_{\dot{x}} \]
\[ B = \left. \frac{\partial f}{\partial u} \right|_{\dot{x}} \]
\[ L = \left. \frac{\partial f}{\partial w} \right|_{\dot{x}} \quad M = \left. \frac{\partial h}{\partial v} \right|_{\dot{x}} \]

\[ \dot{x} = Ax + Bu + Lw \]
\[ y = Cx + Mv \]
EXTENDED KALMAN FILTER

- Discretization:

\[ \dot{x}(t) = f(x(t), u(t), w(t)) \]

\[ \dot{x}(t_k) = f(x(t_k), u(t_k), v(t_k)) \]
EKF ALGORITHMS

Start with continuous-time nonlinear model
\[ \dot{x}(t) = f(x(t), u(t)) + Gw(t) \]

Via linearization

Continuous-time linear model
\[ \dot{x}(t) = Ax(t) + Bu(t) + Gw(t) \]

Via discretization

Discrete-time nonlinear model
\[ x_{k+1} = f(x_k, u_k) + Gw_k \]

Via continuous KF

\[ \dot{P} = AP + PA^T + GG^T \]

Evaluate A at x(t)

Algorithm 1

Evaluate A at x_k

Algorithm 1.1

Discrete-time linear model
\[ x_{k+1} = Ax_k + Bu_k \]

Accurate discretization

Evaluate discretization

Algorithm 2

Euler approximation

Evaluated A via Sensitivity Equation

Algorithm 2.1

Discrete-time linear model
\[ x_{k+1} = A_kx_k + B_ku_k + Gw_k \]

(\( A_k \) is not explicit)

Euler approximation & Linearization

Evaluated A via Finite Difference

Algorithm 3

Algorithm 3.01

Algorithm 3.1
ALGORITHM 1

Start with continuous-time nonlinear model
\[ \dot{x}(t) = f(x(t), u(t)) + Gw(t) \]

Via linearization

Continuous-time linear model
\[ \dot{x}(t) = Ax(t) + Bu(t) + Gw(t) \]

Via continuous KF

\[ \dot{P} = AP + PA' + GQG' \]
**ALGORITHM 1**

- Linearize and apply continuous EKF

<table>
<thead>
<tr>
<th>Initialization</th>
<th>$\hat{x}_0 = \tilde{x}<em>0, \quad P_0 = P</em>{x_0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prediction</strong></td>
<td>$\dot{\hat{x}} = f(\hat{x}, u)$</td>
</tr>
<tr>
<td></td>
<td>$\dot{P} = A(\hat{x})P + PA(\hat{x})' + GQG'$ where $A(\hat{x}) = \frac{\partial f}{\partial x} \bigg</td>
</tr>
<tr>
<td></td>
<td>$\hat{x}(kT) = \hat{x}_k^-, \quad P(kT) = P_k^-$</td>
</tr>
<tr>
<td><strong>Kalman gain</strong></td>
<td>$K_k = P_k^- H_k' (H_k P_k^- H_k' + R)^{-1}$, where $H_k = \frac{\partial h}{\partial x} \bigg</td>
</tr>
<tr>
<td><strong>Correction</strong></td>
<td>$P_k = (I - K_k H_k) P_k^-$</td>
</tr>
<tr>
<td></td>
<td>$\hat{x}_k = \hat{x}_k^- + K_k [y_k - h(\hat{x}_k^-)]$</td>
</tr>
</tbody>
</table>
ALGORITHM 2

Start with continuous-time nonlinear model
\[ \dot{x}(t) = f(x(t), u(t)) + Gw(t) \]

Via linearization

Continuous-time linear model
\[ \dot{x}(t) = Ax(t) + Bu(t) + Gw(t) \]

Via discretization

Discrete-time linear model
\[ x_{k+1} = A_k x_k + B_k u_k \]
**ALGORITHM 2**

- Linearize, discretize, then apply discrete EKF

<table>
<thead>
<tr>
<th>Initialization</th>
<th>$\hat{x}_0 = \bar{x}<em>0$, $P_0 = P</em>{x_0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction</td>
<td>$\hat{x} = f(\hat{x}, u)$</td>
</tr>
<tr>
<td></td>
<td>$P_{k+1}^- = A_k P A_k'^T + G Q_k G'$</td>
</tr>
<tr>
<td></td>
<td>where $A(\hat{x}_k) = \frac{df}{dx} \bigg</td>
</tr>
<tr>
<td>Kalman gain</td>
<td>$K_k = P_k^- H_k' (H_k P_k^- H_k' + R)^{-1}$, where $H_k = \frac{dh}{dx} \bigg</td>
</tr>
<tr>
<td>Correction</td>
<td>$P_k = (I - K_k H_k) P_k^-$</td>
</tr>
<tr>
<td></td>
<td>$\hat{x}_k = \hat{x}_k^- + K_k [y_k - h(\hat{x}_k^-)]$</td>
</tr>
</tbody>
</table>
ALGORITHM 3

Start with continuous-time nonlinear model:
\[ \dot{x}(t) = f(x(t), u(t)) + Gw(t) \]

Via discretization:

Discrete-time nonlinear model:
\[ x_{k+1} = f(x_k, u_k) + Gw_k \]

Accurate Discretization & Linearization:

Discrete-time linear model:
\[ x_{k+1} = A_k x_k + B_k u_k + Gw_k \]

(A_k is not explicit)
**ALGORITHM 3**

- Discretize, linearize, then apply discrete EKF

<table>
<thead>
<tr>
<th>Initialization</th>
<th>$\hat{x}_0 = \bar{x}<em>0$, $P_0 = P</em>{x_0}$, $A_0 = I$</th>
</tr>
</thead>
</table>
| Prediction    | $\hat{\dot{x}} = f(\hat{x}, u), \dot{A} = \frac{\partial f}{\partial x} A$  
               | $\hat{x}(kT) = \hat{x}_k^-, A(kT) = A_k$  
               | $P_{k+1}^- = A_k P_k A_k^T + GQ_k G^T$ |
| Kalman gain   | $K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R)^{-1}$, where $H_k = \frac{\partial h}{\partial x} \bigg|_{x=\hat{x}_k^-}$ |
| Correction    | $P_k = (I - K_k H_k) P_k^-$  
               | $\hat{x}_k = \hat{x}_k^- + K_k [y_k - h(\hat{x}_k^-)]$ |
EKF ALGORITHMS

Start with continuous-time nonlinear model
\[ \dot{x}(t) = f(x(t), u(t)) + Gw(t) \]

Via linearization

Continuous-time linear model
\[ \dot{x}(t) = Ax(t) + Bu(t) + Gw(t) \]

Via continuous KF

\[ \dot{P} = AP + PA^T + QG \]

Evaluate A at \( x(t) \)

Algorithm 1

Via discretization

Discrete-time nonlinear model
\[ x_{k+1} = f(x_k, u_k) + Gw_k \]

Via discretization

Discrete-time linear model
\[ x_{k+1} = Ax_k + Bu_k + Gw_k \]

Evaluate A

Algorithm 3

Euler approximation & Linearization

Algorithm 3.01

Evaluate A via Finite Difference

Algorithm 3.1

Euler approximation & Linearization

Algorithm 3.01

Evaluate A via Sensitivity Equation

Algorithm 2.1

Euler approximation

Algorithm 2

Accurate discretization

Algorithm 1.1

Evaluate A at \( x_k \)
EKF ALGORITHMS

Start with continuous-time nonlinear model
\[ \dot{x}(t) = f(x(t), u(t)) + Gw(t) \]

Via linearization

Continuous-time linear model
\[ \dot{x}(t) = Ax(t) + Bu(t) + Gw(t) \]

Via discretization

Discrete-time nonlinear model
\[ x_{k+1} = f(x_k, u_k) + Gw_k \]

Via discretization

\[ \dot{P} = AP + PA' + GQG' \]

Via continuous KF

Discrete-time linear model
\[ x_{k+1} = A_k x_k + B_k u_k \]

Evaluate A at x(t)
Algorithm 1

Evaluate A at x_k
Algorithm 1.1

Evaluate discretization
Algorithm 2

Euler approximation
Algorithm 2.1

Evaluate A via Sensitivity Equation
Algorithm 3

Evaluate A via Finite Difference
Algorithm 3.01

Euler approximation & Linearization
Algorithm 3.1

Accurate Discretization & Linearization

Discrete-time linear model
\[ x_{k+1} = A_k x_k + B_k u_k + Gw_k \]

(\( A_k \) is not explicit)
CASE STUDY – ISOTHERMAL CSTR

\[ A \xrightarrow{k_1} B \xrightarrow{k_2} C \]

\[ 2A \xrightarrow{k_3} D \]

\[
\begin{align*}
\frac{dC_A}{dt} &= \frac{u}{V} (C_{ain} - C_A) - k_1 e^{-E_1/RT} C_A - k_3 e^{-E_3/RT} C_A^2 \\
\frac{dC_B}{dt} &= -\frac{u}{V} C_B + k_1 e^{-E_1/RT} C_A - k_2 e^{-E_2/RT} C_B \\
\frac{dT}{dt} &= \frac{1}{\rho c_p} \left[ k_1 e^{-E_1/RT} C_A (-\Delta H_1) + k_2 e^{-E_2/RT} C_B (-\Delta H_2) + k_3 e^{-E_3/RT} C_A^2 (-\Delta H_3) \right] \\
&\quad + \frac{u}{V} (T_{in} - T) + \frac{Q}{V \rho c_p}
\end{align*}
\]
CASE STUDY – ISOTHERMAL CSTR
## RESULTS

- **Mean Square Errors**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>1</th>
<th>1.1</th>
<th>2</th>
<th>2.1</th>
<th>3</th>
<th>3.01</th>
<th>3.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta t = 0.02, R = 0.0001 I$</td>
<td>0.250</td>
<td>0.247</td>
<td>0.247</td>
<td>0.339</td>
<td>0.361</td>
<td>0.368</td>
<td>0.420</td>
</tr>
<tr>
<td>$\Delta t = 0.02, R = 1 I$</td>
<td>0.686</td>
<td>0.686</td>
<td>0.686</td>
<td>6.158</td>
<td>0.689</td>
<td>0.691</td>
<td>11.370</td>
</tr>
<tr>
<td>$\Delta t = 0.002, R = 0.01 I$</td>
<td>1.516</td>
<td>1.517</td>
<td>1.517</td>
<td>1.508</td>
<td>1.512</td>
<td>1.508</td>
<td>1.507</td>
</tr>
<tr>
<td>$\Delta t = 0.2, R = 0.01 I$</td>
<td>0.323</td>
<td>0.325</td>
<td>0.325</td>
<td>2.384</td>
<td>0.306</td>
<td>0.306</td>
<td>42.331</td>
</tr>
</tbody>
</table>
RESULTS

- **Mean Square Errors**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>1</th>
<th>1.1</th>
<th>2</th>
<th>2.1</th>
<th>3</th>
<th>3.01</th>
<th>3.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta t = 0.02, R = 0.0001I$</td>
<td>0.250</td>
<td>0.247</td>
<td>0.247</td>
<td>0.339</td>
<td>0.361</td>
<td>0.368</td>
<td>0.420</td>
</tr>
<tr>
<td>$\Delta t = 0.02, R = 1I$</td>
<td>0.686</td>
<td>0.686</td>
<td>0.686</td>
<td>6.158</td>
<td>0.689</td>
<td>0.691</td>
<td>11.370</td>
</tr>
<tr>
<td>$\Delta t = 0.002, R = 0.01I$</td>
<td>1.516</td>
<td>1.517</td>
<td>1.517</td>
<td>1.508</td>
<td>1.512</td>
<td>1.508</td>
<td>1.507</td>
</tr>
<tr>
<td>$\Delta t = 0.2, R = 0.01I$</td>
<td>0.323</td>
<td>0.325</td>
<td>0.325</td>
<td>2.384</td>
<td>0.306</td>
<td>0.306</td>
<td>42.331</td>
</tr>
</tbody>
</table>
## RESULTS

- **Mean Square Errors**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>1</th>
<th>1.1</th>
<th>2</th>
<th>2.1</th>
<th>3</th>
<th>3.01</th>
<th>3.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta t = 0.02, R = 0.0001I$</td>
<td>0.250</td>
<td>0.247</td>
<td>0.247</td>
<td>0.339</td>
<td>0.361</td>
<td>0.368</td>
<td>0.420</td>
</tr>
<tr>
<td>$\Delta t = 0.02, R = 1I$</td>
<td>0.686</td>
<td>0.686</td>
<td>0.686</td>
<td>6.158</td>
<td>0.689</td>
<td>0.691</td>
<td>11.370</td>
</tr>
<tr>
<td>$\Delta t = 0.002, R = 0.01I$</td>
<td>1.516</td>
<td>1.517</td>
<td>1.517</td>
<td>1.508</td>
<td>1.512</td>
<td>1.508</td>
<td>1.507</td>
</tr>
<tr>
<td>$\Delta t = 0.2, R = 0.01I$</td>
<td>0.323</td>
<td>0.325</td>
<td>0.325</td>
<td>2.384</td>
<td>0.306</td>
<td>0.306</td>
<td>42.331</td>
</tr>
</tbody>
</table>
RESULTS

- With 50% input change

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>1</th>
<th>1.1</th>
<th>2</th>
<th>2.1</th>
<th>3</th>
<th>3.01</th>
<th>3.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta t = 0.02, R = 0.01I$</td>
<td>1.297</td>
<td>1.297</td>
<td>1.297</td>
<td>17.090</td>
<td>1.296</td>
<td>1.296</td>
<td>380.76</td>
</tr>
<tr>
<td>$\Delta t = 0.02, R = 0.0001I$</td>
<td>0.978</td>
<td>0.986</td>
<td>0.986</td>
<td>7.658</td>
<td>1.095</td>
<td>1.140</td>
<td>24.654</td>
</tr>
<tr>
<td>$\Delta t = 0.2, R = 0.01I$</td>
<td>0.567</td>
<td>0.569</td>
<td>0.569</td>
<td>NaN</td>
<td>0.576</td>
<td>0.576</td>
<td>NaN</td>
</tr>
</tbody>
</table>
CONCLUSIONS

- Continuous evaluation of covariance P matrix always works well
- Evaluation at discrete times can be used in very specific situations
  - Low measurement noise
  - Small sampling interval time
  - “ideal”
- Euler approximations not recommended when sampling interval is not very small
THANKS

- Yunfei Chu
- Cheryl Qu
- Dr. Juergen Hahn
- Dr. Sam Mannan
- Mary Kay O’Connor Process Safety Center
QUESTIONS?