The Kalman-Bucy Filter

Alexander Wittmond

University of Missouri

October 18, 2020

Table of Contents

Classical Models

Classically, time dependent systems are modeled by:

Classical Models

Classically, time dependent systems are modeled by: Differential equations

$$f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n \tag{1}$$

$$\frac{dx}{dt} = f(x, t) \tag{2}$$

Classical Models

Classically, time dependent systems are modeled by: Differential equations

$$f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n \tag{1}$$

$$\frac{dx}{dt} = f(x, t) \tag{2}$$

Solved by:

$$x : \mathbb{R} \to \mathbb{R}^n \tag{3}$$

The problem of noise

- reality is more complex

The problem of noise

- reality is more complex
- measurements of a system have a maximum resolution

The problem of noise

- reality is more complex
- measurements of a system have a maximum resolution
- beyond that resolution we see random noise caused by unknown variables

The problem of noise

- reality is more complex
- measurements of a system have a maximum resolution
- beyond that resolution we see random noise caused by unknown variables
- so our observations are better modeled by a random variable X on some probability space (Ω, P) than a point in $\mathbb{R}^n$

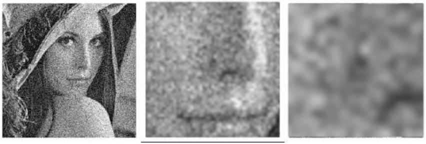

Figure: An example of a noisy measurement

Stochastic Models

Stochastic Models

Stochastic differential equations

$$dx_t = f(x_t, t)dt + g(x_t, t)d\beta_t \tag{4}$$

meaning

$$x_t - x_0 = \int_0^t f(x_t, t)dt + \int_0^t g(x_t, t)d\beta_t \tag{5}$$

Stochastic Models

Stochastic differential equations

$$dx_t = f(x_t, t)dt + g(x_t, t)d\beta_t \tag{4}$$

meaning

$$x_t - x_0 = \int_0^t f(x_t, t)dt + \int_0^t g(x_t, t)d\beta_t \tag{5}$$

Solved by a stochastic process

$$X : (\Omega, P) \times \mathbb{R} \to \mathbb{R}^n \tag{6}$$

$$\{X_t\}_{t \in R} \tag{7}$$

Stochastic Model of Measurement

In general we cannot measure a system directly and our measurement may be subject to noise.

Stochastic Model of Measurement

In general we cannot measure a system directly and our measurement may be subject to noise.

$$dx_t = f(x_t, t)dt + g(x_t, t)d\beta_t \tag{8}$$

$$y_n = h(x_t) + \nu_n \tag{9}$$

Stochastic Model of Measurement

In general we cannot measure a system directly and our measurement may be subject to noise.

$$dx_t = f(x_t, t)dt + g(x_t, t)d\beta_t \tag{8}$$

$$y_n = h(x_t) + \nu_n \tag{9}$$

We will assume our measurement is subject to white noise

$$v_n \sim N(0, R_k) \text{ i.i.d} \tag{10}$$

The Filtering Problem and its Solution

Given a set of observations $\{Y_{t_1}, \ldots, Y_{t_n}\} = Y_n$ we want to estimate the distribution of X_t.

The Filtering Problem and its Solution

Given a set of observations $\{Y_{t_1}, \dots, Y_{t_n}\} = Y_n$ we want to estimate the distribution of X_t.

The Filtering Problem and its Solution

Given a set of observations $\{Y_{t_1}, \ldots, Y_{t_n}\} = \mathcal{Y}_n$ we want to estimate the distribution of X_t.

- **Smoothing** $t < t_n$

The Filtering Problem and its Solution

Given a set of observations $\{Y_{t_1}, \ldots, Y_{t_n}\} = Y_n$ we want to estimate the distribution of X_t.

- **Smoothing** $t < t_n$
- **Filtering** $t = t_n$

The Filtering Problem and its Solution

Given a set of observations $\{Y_{t_1}, \ldots, Y_{t_n}\} = Y_n$ we want to estimate the distribution of X_t.

- **Smoothing** $t < t_n$
- **Filtering** $t = t_n$
- **Prediction** $t > t_n$

The Filtering Problem and its Solution

Given a set of observations $\{Y_{t_1}, \ldots, Y_{t_n}\} = Y_n$ we want to estimate the distribution of X_t.

- **Smoothing** $t < t_n$
- **Filtering** $t = t_n$
- **Prediction** $t > t_n$

We want to compute

$$p(x, t \mid Y_n) \tag{11}$$

The Filtering Problem and its Solution

Given a set of observations $\{Y_{t_1}, \ldots, Y_{t_n}\} = Y_n$ we want to estimate the distribution of X_t.

- **Smoothing** $t < t_n$
- **Filtering** $t = t_n$
- **Prediction** $t > t_n$

We want to compute

$$p(x, t \mid Y_n) \tag{11}$$

The minimum variance estimate of X_t is

$$\mathbb{E}[x \mid Y_t] \tag{12}$$

the mean of $p(x, t \mid Y_n)$

Table of Contents

The Linear Problem

Linear systems are given by:

The Linear Problem

Linear systems are given by:

$$dx_t = F(t)x_t dt + G(t)d\beta_t \tag{13}$$

$$y_k = M(t_k)x_{t_k} + \nu_k \tag{14}$$

$$\mathbb{E}[d\beta_t d\beta_t^T] = Q(t)dt, \quad \nu_k \sim N(0, R_k) \tag{15}$$

The Linear Problem

Linear systems are given by:

$$dx_t = F(t)x_t dt + G(t)d\beta_t \tag{13}$$

$$y_k = M(t_k)x_{t_k} + \nu_k \tag{14}$$

$$\mathbb{E}[d\beta_t d\beta_t^T] = Q(t)dt, \quad \nu_k \sim N(0, R_k) \tag{15}$$

Characterization of Linear Systems

- Linear systems are Gauss Markov processes with means described by deterministic linear systems.

$$x_t \sim N(\hat{x}_t, P_t) \tag{16}$$

- We only need to know the mean and variance of the system at a given time to characterize $P(x, t)$

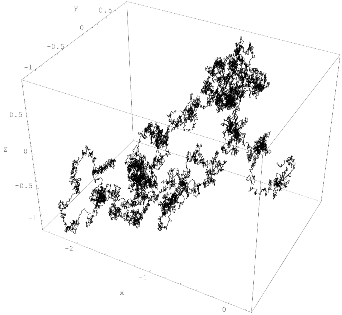

Figure: Three dimensional Brownian motion

The Solution To The Linear Discrete-Continuous Filtering Problem

The Kalman-Bucy filter consists of:

Evolution for

$$\text{Conditional Mean } \mathbb{E}[x_t \mid Y_{t_k}] = \hat{x}_t^{t_k} \tag{17}$$

$$\text{Conditional Variance } \mathbb{E}[(x_t - \hat{x}_t)(x_t - \hat{x}_t)^T \mid Y_{t_k}] = P_t^{t_k} \tag{18}$$

given some prior x_0 and P_0^0.

The Solution To The Linear Discrete-Continuous Filtering Problem

The Kalman-Bucy filter consists of:

Prediction between observations

$$\frac{d}{dt}\hat{x}_t^{t_k} = F(t)\hat{x}_t^{t_k} \tag{17}$$

$$\frac{d}{dt}P^{t_k} = F(t)P_t^{t_k} + P_t^{t_k}F^T(t) + G(t)Q(t)G^T(t) \tag{18}$$

$$t_k \leq t < t_{k+1} \tag{19}$$

The Solution To The Linear Discrete-Continuous Filtering Problem

The Kalman-Bucy filter consists of:

Update at observations

$$\hat{x}_{t_k}^{t_k} = \hat{x}_{t_k}^{t_{k-1}} + K(t_k)(y_k - M(t_k)\hat{x}_{t_k}^{t_{k-1}}) \tag{17}$$

$$P_{t_k}^{t_{k-1}} = P_{t_k}^{t_{k-1}} - K(t_k)M(t_k)P_{t_k}^{t_{k-1}} \tag{18}$$

where the **Kalman Gain** $K(t)$ is given by

$$P_{t_k}^{t_{k-1}} M^T(t_k)[M(t_k)P_{t_k}^{t_{k-1}} M^T(t_k) + R_k]^{-1} \tag{19}$$

In Code

```cpp
arma::dvec KalmanFilter::update(double t, arma::vec observation){

  const arma::dmat M = measurement->measurement_matrix(t),
    R = measurement->covariance(t);

    //predict
  const arma::dvec old_state = model->extrapolate( t );
  const arma::dmat P = matrix_model->extrapolate(t);

  // the Kalman gain
  const arma::dmat K =
      P * arma::trans(M) * arma::inv(M * P * arma::trans(M) + R);

  // adjust
  const arma::dmat KM = K * M;
  const arma::dmat I_KM = arma::eye(KM.n_rows,KM.n_cols) - KM;

  //correct with residual multiplied by Kalman Gain
  const arma::dvec new_state = old_state + K*(observation - M*old_state);

  //adjust covariance with the change in uncertainty caused by observation
  const arma::dmat new_covariance = I_KM*P*arma::trans(I_KM) + K*R*arma::trans(K);

  //now integrate from new initial estimates
  model->set_initial_conditions(t , new_state );
  matrix_model->set_initial_conditions(t , new_covariance);

  return new_state;
}
```

Figure: The Kalman Filter update in C++

Considerations

Considerations

- To integrate the prediction equations we need to fix the boundary conditions $\hat{x}_{t_0}$, P_{t_0}

Considerations

- To integrate the prediction equations we need to fix the boundary conditions $\hat{x}_{t_0}$, P_{t_0}
- Calculating the Kalman gain is $O(n^3)$ in the dimension of the measurement due to the matrix inversion

Considerations

- To integrate the prediction equations we need to fix the boundary conditions $\hat{x}_{t_0}$, P_{t_0}
- Calculating the Kalman gain is $O(n^3)$ in the dimension of the measurement due to the matrix inversion
- The Kalman gain can be precomputed

Considerations

- To integrate the prediction equations we need to fix the boundary conditions $\hat{x}_{t_0}$, P_{t_0}
- Calculating the Kalman gain is $O(n^3)$ in the dimension of the measurement due to the matrix inversion
- The Kalman gain can be precomputed
- The update is recursive so we only need to store our previous estimates at the time of the update

Extension to Non-Linear Problems

We start with the system

$$dx_t = f(x_t, t)dt + G(t)d\beta_t \tag{20}$$

$$y_k = h(x_{t_k}, t) + \nu_t \tag{21}$$

where

$$\mathbb{E}[d\beta_t d\beta_t^T] = Q(t)dt \tag{22}$$

Extension to Non-Linear Problems

We start with the system

$$dx_t = f(x_t, t)dt + G(t)d\beta_t \tag{20}$$

$$y_k = h(x_{t_k}, t) + \nu_t \tag{21}$$

where

$$\mathbb{E}[d\beta_t d\beta_t^T] = Q(t)dt \tag{22}$$

Then we pick a **reference trajectory** $\bar{x}_{t_0}(t)$ with some given $\bar{x}(t_0)$ such that

$$\frac{d\bar{x}_{t_0}}{dt}(t) = f(\bar{x}_{t_0}(t), t) \tag{23}$$

Extension to Non-Linear Problems

Then we can look at the **deviation**

$$\delta x(t) = x(t) - \bar{x}_{t_0}(t) \tag{24}$$

which statisfies

Extension to Non-Linear Problems

Then we can look at the **deviation**

$$\delta x(t) = x(t) - \bar{x}_{t_0}(t) \tag{24}$$

which statisfies

$$d(\delta x_t) = (f(x_t, t) - f(\bar{x}_{t_0}(t), t))dt + G(t)d\beta_t \tag{25}$$

Extension to Non-Linear Problems

Then we can look at the **deviation**

$$\delta x(t) = x(t) - \bar{x}_{t_0}(t) \tag{24}$$

which statisfies

$$d(\delta x_t) = (f(x_t, t) - f(\bar{x}_{t_0}(t), t))dt + G(t)d\beta_t \tag{25}$$

then do a Taylor approximation

$$f(x_t, t) - f(\bar{x}_t, t) \simeq Df(\bar{x}_{t_0}(t), t)\delta x_t \tag{26}$$

Extension to Non-Linear Problems

Then we can look at the **deviation**

$$\delta x(t) = x(t) - \bar{x}_{t_0}(t) \tag{24}$$

which statisfies

$$d(\delta x_t) = (f(x_t, t) - f(\bar{x}_{t_0}(t), t))dt + G(t)d\beta_t \tag{25}$$

then do a Taylor approximation

$$f(x_t, t) - f(\bar{x}_t, t) \simeq Df(\bar{x}_{t_0}(t), t)\delta x_t \tag{26}$$

Then we have the linear equation

$$d(\delta x_t) = Df(\bar{x}_{t_0}, t)\delta x_t dt + G(t)d\beta_t \tag{27}$$

Extension to Non-Linear Problems

We can linearize the measurement in the same way to get

$$\delta y_{t_k} = y_{t_k} - h(\bar{x}_{t_0}(t_k), t_k) + \nu_k \tag{28}$$

$$\delta y_{t_k} \simeq Dh(\bar{x}_{t_0}(t_k), t_k)\delta x_{t_k} + \nu_k \tag{29}$$

Extension to Non-Linear Problems

We can linearize the measurement in the same way to get

$$\delta y_{t_k} = y_{t_k} - h(\bar{x}_{t_0}(t_k), t_k) + \nu_k \tag{28}$$

$$\delta y_{t_k} \simeq Dh(\bar{x}_{t_0}(t_k), t_k)\delta x_{t_k} + \nu_k \tag{29}$$

Then we can process the system

$$d(\delta x_t) = Df(\bar{x}_{t_0}(t), t)\delta x_t dt + G(t)d\beta_t \tag{30}$$

$$\delta y_{t_k} = Dh(\bar{x}_{t_0}(t_k), t_k)\delta x_{t_k} + \nu_k \tag{31}$$

with linear filter.

Extension to Non-Linear Problems

We can then estimate $\hat{x}_{t_k}^{t_k}$ with

$$\hat{x}_{t_k}^{t_k} = \bar{x}_{t_0}(t_k) + \widehat{\delta x}_{t_k}^{t_k} \tag{32}$$

Our variance matrix $P_{t_k}^{t_k}$ estimates the variance of this estimation.

Extension to Non-Linear Problems

If at every observation $h(t_k)$, we relinearize around our estimate $\hat{x}_{t_k}^{t_k}$ by getting a new reference trajectory $\bar{x}_{t_k}(t)$ starting at this point, and then estimate the system using

$$d(\delta x_t) = Df(\bar{x}_{t_k}(t), t_k)\delta x_t dt + G(t)d\beta_t \tag{33}$$

$$\delta y_{t_{k+1}} \simeq Dh(\bar{x}_{t_k}(t_k), t_k)\delta x_{t_k} + \nu_k \tag{34}$$

then this is known as the **Extended Kalman Filter**

In Code

```cpp
arma::dvec ExtendedKalmanFilter::update(double t, arma::vec observation) {
  arma::dvec nominal_state = model->extrapolate(t );

  arma::dvec measurement_error = observation - measurement->measure(t , nominal_state);

  arma::dvec error = perturbationProcessFilter.update(t, measurement_error);

  arma::dvec state_estimate = nominal_state + error;

  model->set_initial_conditions(t , state_estimate);

  perturbationProcessFilter.get_model()->set_initial_conditions
    (t,arma::dvec(nominal_state.size(),arma::fill::zeros));


  return state_estimate;
}
```

Figure: The Extended Kalman Filter update in C++

Pros and Cons

Pros

- Recursiveness causes low memory requirements
- Relatively low cost to computing an update
- Most computationally expensive items can be computed in advance

Cons

- Is not adaptive, relies on the accuracy of the underlying model
- Can not capture multi-modalities
- Suffers from filter divergence

Table of Contents

Orbit Determination

The simulation will be filtering noisy measurements from a deterministic system.

Orbit Determination

The simulation will be filtering noisy measurements from a deterministic system.

We have

$$dx_t = f(x_t, t)dt \tag{35}$$

$$y_k = h(x_{t_k}, t) + \nu_t \tag{36}$$

$$\nu_t \sim N(0, Q) \tag{37}$$

Orbit Determination

$$x_t = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} \tag{38}$$

$$f(x_t, t) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ -\mu \frac{x}{x^2+y^2} \\ -\mu \frac{y}{x^2+y^2} \end{bmatrix} \tag{39}$$

If p gives the position of the sensor and

$$\bar{x} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{40}$$

then

$$h(x_t, t) = \begin{bmatrix} \|\bar{x} - p\| \\ \frac{(x-p_1)\dot{x}+(y-p_2)\dot{y}}{\|\bar{x}-p\|} \\ \frac{x \cdot p}{\|x\|} \end{bmatrix} \tag{41}$$

$$Q_t = \begin{bmatrix} n1 & 0 & 0 \\ 0 & n2 & 0 \\ 0 & 0 & n3 \end{bmatrix} \tag{42}$$

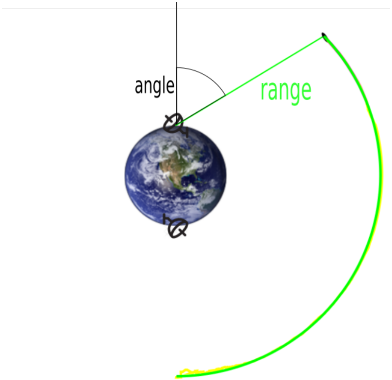

Figure: Measurements in Simulation