



机器人学中的状态估计

Timothy Barfoot 著 高翔, 谢晓佳等 译

Slides by Xiang Gao

2020年春



第2讲 线性高斯系统的状态估计问题

- 离散时间的批量估计
- 离散时间的递归平滑算法
- 离散时间的滤波算法
- 连续时间的批量估计



第2讲 线性高斯系统的状态估计问题

- 离散时间的批量估计
- 离散时间的递归平滑算法
- 离散时间的滤波算法
- 连续时间的批量估计



离散时间的批量估计问题

- 上节课中，我们介绍了：
 - 概率密度函数的定义和各种性质
 - 这些性质在高斯分布上的情况
 - 高斯推断
 - 高斯分布的线性变换和非线性变换，非线性变换的线性化



离散时间的批量估计问题

- 本节课的内容：
 - 线性高斯系统
 - 批量问题的解法
 - 递归式问题的解法
 - 卡尔曼滤波器

离散时间的批量估计问题

- 考虑离散时间的线性时变系统 (Discrete-time, linear, time-varying)

$$\text{运动方程: } x_k = A_{k-1}x_{k-1} + v_k + w_k, \quad k = 1, \dots, K$$

$$\text{观测方程: } y_k = C_k x_k + n_k, \quad k = 0, \dots, K$$

- 各个变量含义:

系统状态: $x_k \in \mathbb{R}^N$

初始状态: $x_0 \in \mathbb{R}^N \sim \mathcal{N}(\check{x}_0, \check{P}_0)$

输入: $v_k \in \mathbb{R}^N$

过程噪声: $w_k \in \mathbb{R}^N \sim \mathcal{N}(0, Q_k)$

测量: $y_k \in \mathbb{R}^M$

测量噪声: $n_k \in \mathbb{R}^M \sim \mathcal{N}(0, R_k)$

而且:

- 除了 v_k 以外, 其他变量都是随机变量
- 各时刻的噪声都不相关
- A称为转移矩阵
- C称为观测矩阵

离散时间的批量估计问题

- 除系统模型之外，还知道：
 - 运动方程： $x_k = A_{k-1}x_{k-1} + v_k + w_k, \quad k = 1, \dots, K$
 - 观测方程： $y_k = C_k x_k + n_k, \quad k = 0, \dots, K$
- 1. 初始状态 \tilde{x}_0 ，以及相应的初始协方差矩阵 \tilde{P}_0 。有时候我们也不知道初始信息，那就必须在没有初始信息的情况下进行推导²。
 2. 输入量 v_k ，通常来自控制器，是已知的³；它的噪声协方差矩阵是 Q_k 。
 3. 观测数据 $y_{k,\text{meas}}$ 是观测变量 y_k 的一次实现（realization），它的协方差为 R_k 。
- 状态估计问题**：通过初始状态、各时刻的观测数据、输入数据，估计系统的真实状态

□ 带帽子的变量称为先验，上帽子称为后验

离散时间的批量估计问题

- 状态估计问题的分类：
 - 系统是否为线性的？ 线性系统/非线性系统
 - 噪声是否为高斯的？ 高斯系统/非高斯系统
 - 问题是批量式还是递归式？ Batch/Recursive方法
- 我们从最简单批量线性高斯系统（Batch Linear Gaussian）开始介绍，逐步扩展范围
- 对于批量LG系统，有两类方法可以估计其状态：
 1. 贝叶斯推断（Bayesian Inference）
 2. 最大后验估计（Maximum A Posteriori, MAP）
- 接下来我们分别介绍这两种方法。先讲MAP方法，再讲贝叶斯方法。

离散时间的批量估计问题

- 1. MAP方法：已知输入和观测时，求最大概率的状态

$$\hat{x} = \arg \max_x p(x|v, y)$$

- 把这种不带下标的变量定义为宏观变量：

$$x = x_{0:K} = (x_0, \cdots, x_K), \quad v = (\tilde{x}_0, v_{1:K}) = (\tilde{x}, v_1, \cdots, v_K)$$

$$y = y_{0:K} = (y_0, \cdots, y_K)$$

- 首先用贝叶斯公式重写目标函数：

$$\hat{x} = \arg \max_x p(x|v, y) = \arg \max_x \frac{p(y|x, v) p(x|v)}{p(y|v)} = \arg \max_x p(y|x) p(x|v)$$

□ 分母与x无关，舍去

离散时间的批量估计问题

$$\hat{x} = \arg \max_x p(x|v, y) = \arg \max_x \frac{p(y|x, v) p(x|v)}{p(y|v)} = \arg \max_x p(y|x) p(x|v)$$

- 由于各时刻观测、输入的噪声都是无关的，上面两个项可以因式分解：

$$p(x|v) = p(x_0|\check{x}_0) \prod_{k=1}^K p(x_k|x_{k-1}, v_k) \qquad p(y|x) = \prod_{k=0}^K p(y_k|x_k)$$

- 同时，对目标函数取对数，对数是个单调映射，不影响最优解：

$$\ln(p(y|x) p(x|v)) = \ln p(x_0|\check{x}_0) + \sum_{k=1}^K \ln p(x_k|x_{k-1}, v_k) + \sum_{k=0}^K \ln p(y_k|x_k)$$

- 这时因子相乘变成了对数项相加

离散时间的批量估计问题

- 高斯分布取对数之后有较好形式：

$$\ln p(x_0|\check{x}_0) = -\frac{1}{2}(x_0 - \check{x}_0)^T \check{P}_0^{-1} (x_0 - \check{x}_0) - \underbrace{\frac{1}{2} \ln \left((2\pi)^N \det \check{P}_0 \right)}_{\text{与 } x \text{ 无关}}$$

$$\ln p(x_k|x_{k-1}, v_k) = -\frac{1}{2}(x_k - A_{k-1}x_{k-1} - v_k)^T Q_k^{-1} (x_k - A_{k-1}x_{k-1} - v_k) - \underbrace{\frac{1}{2} \ln \left((2\pi)^N \det Q_k \right)}_{\text{与 } x \text{ 无关}}$$

$$\ln p(y_k|x_k) = -\frac{1}{2}(y_k - C_k x_k)^T R_k^{-1} (y_k - C_k x_k) - \underbrace{\frac{1}{2} \ln \left((2\pi)^M \det R_k \right)}_{\text{与 } x \text{ 无关}}$$

舍掉与x无关的那些项，定义：

$$J_{v,k}(x) = \begin{cases} \frac{1}{2} (x_0 - \check{x}_0)^T \check{P}_0^{-1} (x_0 - \check{x}_0), & k=0 \\ \frac{1}{2} (x_k - A_{k-1}x_{k-1} - v_k)^T Q_k^{-1} (x_k - A_{k-1}x_{k-1} - v_k), & k=1, \dots, K \end{cases}$$

$$J_{y,k}(x) = \frac{1}{2} (y_k - C_k x_k)^T R_k^{-1} (y_k - C_k x_k), \quad k=0, \dots, K$$

于是目标函数变为求这个式的最小化：

$$\hat{x} = \arg \min_x J(x) \quad J(x) = \sum_{k=0}^K (J_{v,k}(x) + J_{y,k}(x))$$

这个问题就是常见的**无约束最小二乘**

离散时间的批量估计问题

- 写成更紧凑的矩阵形式（提升形式）：

$$z = \begin{bmatrix} \check{x}_0 \\ v_1 \\ \vdots \\ v_K \\ y_0 \\ y_1 \\ \vdots \\ y_K \end{bmatrix}, \quad x = \begin{bmatrix} x_0 \\ \vdots \\ x_K \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & & & & & \\ -A_0 & 1 & & & & \\ & \ddots & \ddots & & & \\ & & & -A_{K-1} & 1 & \\ C_0 & & & & & \\ & C_1 & & & & \\ & & \ddots & & & \\ & & & C_K & & \end{bmatrix}$$

$$W = \begin{bmatrix} \check{P}_0 & & & & & \\ & Q_1 & & & & \\ & & \ddots & & & \\ & & & Q_K & & \\ \hline & & & & R_0 & \\ & & & & & R_1 \\ & & & & & \ddots \\ & & & & & & R_K \end{bmatrix}$$

- 把运动和观测写在一起： $z = Hx + W$

离散时间的批量估计问题

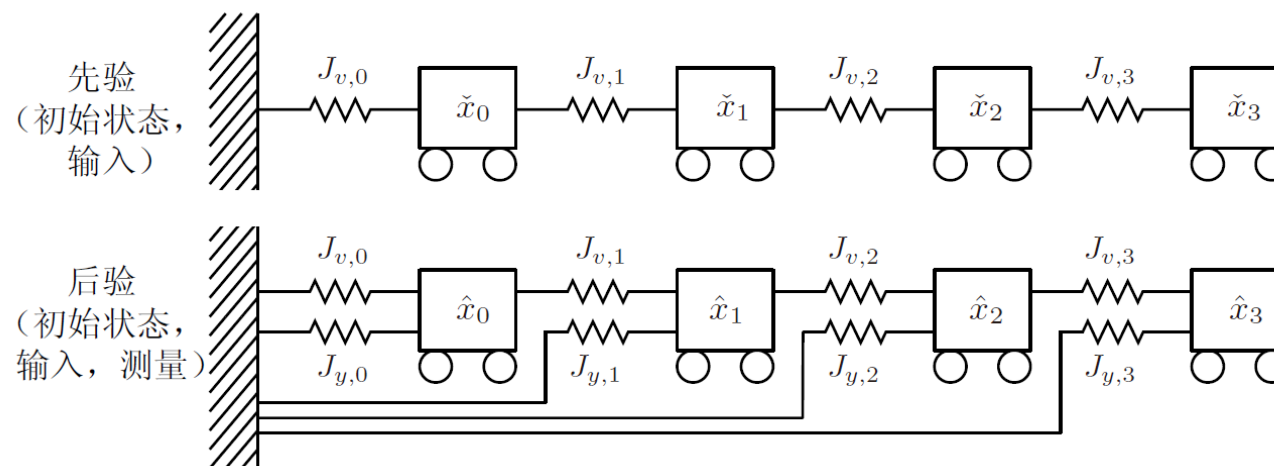
- 提升形式目标函数: $J(x) = \frac{1}{2}(z - Hx)^T W^{-1}(z - Hx)$
- 它是个二次的, 求其最小值, 只要令自变量导数为零:

$$\begin{aligned}\left. \frac{\partial J(x)}{\partial x^T} \right|_{\hat{x}} &= -H^T W^{-1}(z - H\hat{x}) = 0 \\ \Rightarrow (H^T W^{-1} H) \hat{x} &= H^T W^{-1} z\end{aligned}$$

- 于是就解析地得到了**最优解**
 - 这个解等价于经典的批量最小二乘法, 也等价于固定区间平滑算法, 或者也等价于伪逆
 - 由于上一页里H具有特殊的稀疏结构, 这个问题也有特殊的解法, 不需要暴力算矩阵求逆

离散时间的批量估计问题

- 最小二乘的直观解释
- 最小二乘系统可视为弹簧-重物系统，其最优解对应系统最小能量状态
- 可以想象成松开手后，系统最终的稳定状态



离散时间的批量估计问题

- 2. 贝叶斯推断
- 在LG系统中，可以根据运动方程和观测方程显式写出状态变量分布的变化过程

单个时刻: $x_k = A_{k-1}x_{k-1} + v_k + w_k$

提升形式: $x = A(v + w)$

□ 思考: v 是什么形式?

$$A = \begin{bmatrix} 1 & & & & \\ A_0 & 1 & & & \\ A_1 A_0 & A_1 & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ A_{K-2} \cdots A_0 & A_{K-2} \cdots A_1 & A_{K-2} \cdots A_2 & \cdots & 1 \\ A_{K-1} \cdots A_0 & A_{K-1} \cdots A_1 & A_{K-1} \cdots A_2 & \cdots & A_{K-1} & 1 \end{bmatrix}$$

- 提升形式里，右侧只有 v 和 w ，容易求得其均值和协方差：

□ 此处 v 是确定的， w 是高斯的，因此为高斯分布的线性变换

$$\tilde{x} = E[x] = E[A(v + w)] = Av$$

$$\check{P} = E[(x - E[x])(x - E[x])^T] = AQA^T$$



离散时间的批量估计问题

- 因此，先验部分写为： $p(x|v) = \mathcal{N}(\tilde{x}, \check{P}) = \mathcal{N}(Av, AQA^T)$

- 先验的意思是：仅考虑运动方程时的条件概率分布

- 再看观测模型：

$$y_k = C_k x_k + n_k$$

单次观测

$$y = Cx + n$$

提升形式

$$C = \text{diag}(C_0, C_1, \dots, C_K)$$

- 于是联合分布写为：

$$p(x, y|v) = \mathcal{N}\left(\begin{bmatrix} \tilde{x} \\ C\tilde{x} \end{bmatrix}, \begin{bmatrix} \check{P} & \check{P}C^T \\ C\check{P} & C\check{P}C^T + R \end{bmatrix}\right)$$

□ 已知 v 时， x 的先验分布已经确定，于是 y 的分布亦可确定



离散时间的批量估计问题

- 已知联合分布，欲求条件分布，如何求？

- 还记得上一节的高斯推断吗？联合=条件 * 边缘

$$p(\mathbf{x}, \mathbf{y} | \mathbf{v}) = p(\mathbf{x} | \mathbf{v}, \mathbf{y}) p(\mathbf{y} | \mathbf{v}) \quad p(\mathbf{x}, \mathbf{y} | \mathbf{v}) = \mathcal{N} \left(\begin{bmatrix} \tilde{\mathbf{x}} \\ C\tilde{\mathbf{x}} \end{bmatrix}, \begin{bmatrix} \check{\mathbf{P}} & \check{\mathbf{P}}C^T \\ C\check{\mathbf{P}} & C\check{\mathbf{P}}C^T + \mathbf{R} \end{bmatrix} \right)$$

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x} | \mathbf{y}) p(\mathbf{y})$$

$$p(\mathbf{x} | \mathbf{y}) = \mathcal{N}(\mu_{\mathbf{x}} + \Sigma_{\mathbf{x}\mathbf{y}} \Sigma_{\mathbf{y}\mathbf{y}}^{-1} (\mathbf{y} - \mu_{\mathbf{y}}), \Sigma_{\mathbf{x}\mathbf{x}} - \Sigma_{\mathbf{x}\mathbf{y}} \Sigma_{\mathbf{y}\mathbf{y}}^{-1} \Sigma_{\mathbf{y}\mathbf{x}})$$

$$p(\mathbf{y}) = \mathcal{N}(\mu_{\mathbf{y}}, \Sigma_{\mathbf{y}\mathbf{y}})$$

- 联合分布
- 条件分布
- 边缘分布

高斯推断

- 逐一代入，立得： $p(\mathbf{x} | \mathbf{v}, \mathbf{y}) = \mathcal{N} \left(\tilde{\mathbf{x}} + \check{\mathbf{P}}C^T (C\check{\mathbf{P}}C^T + \mathbf{R})^{-1} (\mathbf{y} - C\tilde{\mathbf{x}}), \right.$

$$\left. \check{\mathbf{P}} - \check{\mathbf{P}}C^T (C\check{\mathbf{P}}C^T + \mathbf{R})^{-1} C\check{\mathbf{P}} \right)$$

离散时间的批量估计问题

- 代入SMW式进行化简:

$$p(x|v, y) = \mathcal{N}\left(\tilde{x} + \check{P}C^T(C\check{P}C^T + R)^{-1}(y - C\tilde{x}), \right. \\ \left. \check{P} - \check{P}C^T(C\check{P}C^T + R)^{-1}C\check{P}\right)$$

- 得到:
$$p(x|v, y) = \mathcal{N}\left(\underbrace{(\check{P}^{-1} + C^T R^{-1} C)^{-1} (\check{P}^{-1} \tilde{x} + C^T R^{-1} y)}_{\text{即均值 } \hat{x}}, \right. \\ \left. \underbrace{(\check{P}^{-1} + C^T R^{-1} C)^{-1}}_{\text{即后验协方差 } \hat{P}}\right)$$

□ SMW(a)和(c)式

$$AB(D + CAB)^{-1} \equiv (A^{-1} + BD^{-1}C)^{-1} BD^{-1}$$

$$(A^{-1} + BD^{-1}C)^{-1} \equiv A - AB(D + CAB)^{-1} CA$$

离散时间的批量估计问题

- 来整理这里的均值项和协方差:

$$p(x|v, y) = \mathcal{N}\left(\underbrace{(\check{P}^{-1} + C^T R^{-1} C)^{-1}}_{\text{即均值 } \hat{x}} (\check{P}^{-1} \check{x} + C^T R^{-1} y), \underbrace{(\check{P}^{-1} + C^T R^{-1} C)^{-1}}_{\text{即后验协方差 } \hat{P}}\right)$$

$$\text{均值部分: } \underbrace{(\check{P}^{-1} + C^T R^{-1} C)}_{\hat{P}^{-1}} \hat{x} = \check{P}^{-1} \check{x} + C^T R^{-1} y$$

$$\text{代入 } \check{x} = A v \text{ 和 } \check{P}^{-1} = (A Q A^T)^{-1} = A^{-T} Q^{-1} A^{-1},$$

$$\text{得: } \underbrace{(A^{-T} Q^{-1} A^{-1} + C^T R^{-1} C)}_{\hat{P}^{-1}} \hat{x} = A^{-T} Q^{-1} v + C^T R^{-1} y$$

- 由于A的结构, A逆有特殊形式:

$$A = \begin{bmatrix} 1 & & & & \\ A_0 & 1 & & & \\ A_1 A_0 & A_1 & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ A_{K-2} \cdots A_0 & A_{K-2} \cdots A_1 & A_{K-2} \cdots A_2 & \cdots & 1 \\ A_{K-1} \cdots A_0 & A_{K-1} \cdots A_1 & A_{K-1} \cdots A_2 & \cdots & A_{K-1} & 1 \end{bmatrix}$$

$$A^{-1} = \begin{bmatrix} 1 & & & & \\ -A_0 & 1 & & & \\ & -A_1 & 1 & & \\ & & -A_2 & \ddots & \\ & & & \ddots & 1 \\ & & & & -A_{K-1} & 1 \end{bmatrix}$$

离散时间的批量估计问题

- 按照均值式：
$$\underbrace{(A^{-T}Q^{-1}A^{-1} + C^TR^{-1}C)}_{\hat{P}^{-1}} \hat{x} = A^{-T}Q^{-1}v + C^TR^{-1}y$$
- 定义一些矩阵：
$$z = \begin{bmatrix} v \\ y \end{bmatrix}, \quad H = \begin{bmatrix} A^{-1} \\ C \end{bmatrix}, \quad W = \begin{bmatrix} Q & \\ & R \end{bmatrix}$$
- 于是得：
$$(H^TW^{-1}H) \hat{x} = H^TW^{-1}z$$
- 这个结论与MAP结果完全一致！

▣ 回到MAP页



离散时间的批量估计问题

- MAP结果和贝叶斯推断结果一致，说明了什么？
 - MAP只关心达到最大后验概率的一个点，这个点的状态称为MAP估计
 - 而贝叶斯推断写出了 $p(x|y,v)$ 的完整形式，它是一个高斯分布，其均值与MAP估计相等；同时，给出了这个估计的协方差
 - 如果我们只关心状态估计变量取值，那么MAP给出了后验分布的模（Mode），贝叶斯推断给出了均值
 - 而在LG系统中，二者是一样的，使得这两类方法给出了同样的结果

离散时间的批量估计问题

- LG系统最优估计什么时候存在？什么时候唯一？

$$(H^T W^{-1} H) \hat{x} = H^T W^{-1} z \quad \hat{x} = (H^T W^{-1} H)^{-1} H^T W^{-1} z$$

- 显然这要求左侧部分可逆；
- x 的维度：每时刻状态维度为 N ，共 $0, \dots, K$ 总计 $K+1$ 个时刻，因此 $\dim(x)=N(K+1)$
- 所以可逆性要求： $\text{rank}(H^T W^{-1} H) = N(K+1)$
- 又由于协方差矩阵的对称正定性，即要求： $\text{rank}(H^T H) = \text{rank}(H^T) = N(K+1)$
- H 的具体形式取决于问题有没有0时刻的先验条件，对其分类讨论之。

离散时间的批量估计问题

- 1. 存在先验条件

$$\begin{aligned} & \text{rank } H^T \\ &= \text{rank} \left[\begin{array}{cccc|cccc} 1 & -A_0^T & & & C_0^T & & & \\ & 1 & -A_1^T & & & C_1^T & & \\ & & 1 & \ddots & & & C_2^T & \\ & & & \ddots & -A_{K-1}^T & & & \ddots \\ & & & & 1 & & & C_K^T \end{array} \right] \end{aligned}$$

- 显然这个是满秩的（每行有非零块打头），所以只要有先验，最优解就是存在的（且唯一）
- 大前提是噪声协方差为对称正定阵（默认成立）

离散时间的批量估计问题

• 2. 不存在先验

- 这个矩阵阶梯部分必然是满秩的，主要问题在于第一行
- 把第一行挪到最后 A_0^T ，不影响矩阵的秩
- 然后，用第一行乘 $A_0^T A_1^T$ 加至最后一行，再用第二行乘加至最后一行，依此类推；这些都是初等行变换，不影响矩阵求秩

- 于是得：

$$\text{rank } H^T = \text{rank} \left[\begin{array}{cccc|cccc} 1 & -A_1^T & & & & & & \\ & 1 & \ddots & & & & & \\ & & \ddots & -A_{K-1}^T & & & & \\ & & & 1 & & & & \\ \hline & & & & C_0^T & A_0^T C_1^T & A_0^T A_1^T C_2^T & \cdots A_0^T \cdots A_{K-1}^T C_K^T \end{array} \right]$$

$$\text{rank } H^T = \text{rank} \left[\begin{array}{cccc|cccc} -A_0^T & & & & C_0^T & & & \\ \hline 1 & -A_1^T & & & & C_1^T & & \\ & 1 & \ddots & & & & C_2^T & \\ & & \ddots & -A_{K-1}^T & & & & \ddots \\ & & & 1 & & & & C_K^T \end{array} \right]$$

$$\text{rank } H^T = \text{rank} \left[\begin{array}{cccc|cccc} 1 & -A_1^T & & & & C_1^T & & \\ & 1 & \ddots & & & & C_2^T & \\ & & \ddots & -A_{K-1}^T & & & & \ddots \\ & & & 1 & & & & C_K^T \\ \hline -A_0^T & & & & C_0^T & & & \end{array} \right]$$

离散时间的批量估计问题

- 最后判定条件为: $\text{rank} \begin{bmatrix} C_0^T & A_0^T C_1^T & A_0^T A_1^T C_2^T & \cdots & A_0^T \cdots A_{K-1}^T C_K^T \end{bmatrix} = N$

- 如果系统为时不变的, 即A,C矩阵随时间不变, 且 $K \gg N$, 那么:

$$\begin{aligned} & \text{rank} \begin{bmatrix} C^T & A^T C^T & A^T A^T C^T & \cdots & (A^T)^K C^T \end{bmatrix} \\ &= \text{rank} \begin{bmatrix} C^T & A^T C^T & \cdots & (A^T)^{(N-1)} C^T \end{bmatrix} \end{aligned}$$

- 这里要用到卡莱-哈密顿定理 (Caylay-Hamilton Theory), 该定理说明, 矩阵A满足自身的特征方程

- 把rank里面部分称为能观性矩阵:
$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{(N-1)} \end{bmatrix}$$

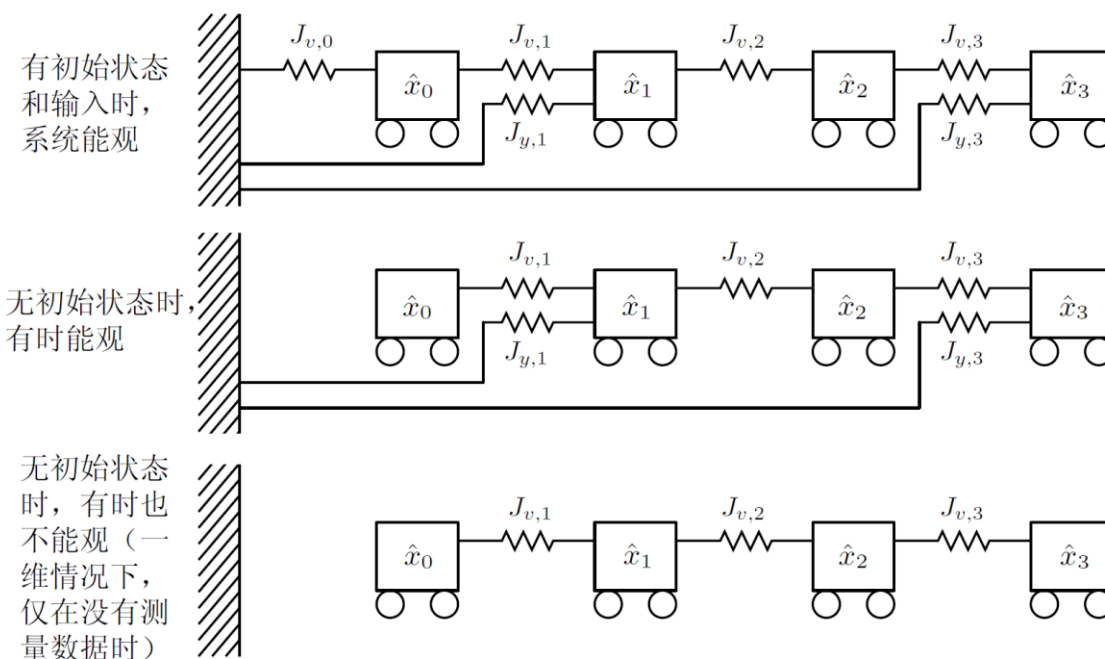
$$\det(\lambda I - A) = 0$$

- 矩阵特征方程:
是一个关于 λ 的N次方程。
- Caylay-Hamilton引理说明, 把A代入特征方程后, 仍然成立
- 这说明A的零次、一次、直到N次是线性相关的, 进而大于N次都可以写成0至N-1次的线性组合

- 能观性矩阵满不满秩, 可作为唯一解存在性判断条件

离散时间的批量估计问题

- 能观性的直观解释：能否通过观测变量唯一地确定系统状态
 - 相应地，自控理论中还有能控性的讨论，但状态估计问题中不涉及控制



□ 不能观：存在一个或多个自由度无法确定

离散时间的批量估计问题

- MAP估计的协方差
- 有时候，除了拿到MAP解之外，我们还关心MAP解的置信度（协方差）
- 实际上，在贝叶斯推断时，已经给出了后验协方差的形式：

$$p(x|v, y) = \mathcal{N}\left(\underbrace{(\check{P}^{-1} + C^T R^{-1} C)^{-1} (\check{P}^{-1} \check{x} + C^T R^{-1} y)}_{\text{即均值 } \hat{x}}, \underbrace{(\check{P}^{-1} + C^T R^{-1} C)^{-1}}_{\text{即后验协方差 } \hat{P}}\right)$$

$$\underbrace{(A^{-T} Q^{-1} A^{-1} + C^T R^{-1} C)}_{\hat{P}^{-1}} \hat{x} = A^{-T} Q^{-1} v + C^T R^{-1} y$$

- 所以，实际上这个方程左侧的系数矩阵即为协方差之逆： $\underbrace{(H^T W^{-1} H)}_{\text{协方差的逆}} \underbrace{\hat{x}}_{\text{均值}} = \underbrace{H^T W^{-1} z}_{\text{信息向量}}$

□ 所以大家以后可以不用再问MAP解置信度怎么算的问题了

第2讲 线性高斯系统的状态估计问题

- 离散时间的批量估计
- 离散时间的递归平滑算法
- 离散时间的滤波算法
- 连续时间的批量估计

离散时间的递归平滑算法

- 很多在线问题当中（比如定位），我们有上一个时刻的先验估计，希望通过这个时刻的控制和观测，计算这个时刻的状态估计
- 下面我们来推导递归问题的解法。
- 为了推出递归算法，首先来看批量问题的一种特殊的解法。批量问题的核心是解线性方程： $(H^T W^{-1} H) \hat{x} = H^T W^{-1} z$
- 之前我们提到左侧矩阵不必蛮力求解，因为它有特殊结构（三对角块）：

$$H = \begin{bmatrix} 1 & & & & \\ -A_0 & 1 & & & \\ & \ddots & \ddots & & \\ & & -A_{K-1} & 1 & \\ C_0 & & & & \\ & C_1 & & & \\ & & \ddots & & \\ & & & C_K & \end{bmatrix} \quad W = \begin{bmatrix} \check{P}_0 & & & & \\ & Q_1 & & & \\ & & \ddots & & \\ & & & Q_K & \\ \hline & & & & R_0 \\ & & & & R_1 \\ & & & & \ddots \\ & & & & & R_K \end{bmatrix} \quad H^T W^{-1} H = \begin{bmatrix} * & * & & & \\ * & * & * & & \\ & * & * & * & \\ & & \ddots & \ddots & \ddots \\ & & & * & * & * \\ & & & & * & * \end{bmatrix}$$

▣ 这一步请同学们务必自己验证一下（只需验证有无即可）

离散时间的递归平滑算法

为什么只提Cholesky而不提其他的？

□ 因为递归方法可以从Cholesky解法引出

- 求解这类方程的一种方式Cholesky分解

- 注意这只是其中一种求解的方式，当然还有其他的解方程方法
- Cholesky分解 $H^T W^{-1} H = L L^T$ ，对于一般矩阵而言，L是下三角阵
- 但是对于三对角块矩阵而言，L的形式更简单：



$$L = \begin{bmatrix} * & & & & & \\ * & * & & & & \\ & * & * & & & \\ & & \ddots & \ddots & & \\ & & & * & * & \\ & & & & * & * \end{bmatrix}$$

- Cholesky解方程的手段：

- 先解： $Ld = H^T W^{-1} z$ 得到d，从上往下解；
- 再解： $L^T \hat{x} = d$ 得到最优状态，从下往上解；
- 注意这种解法对一般线性方程也是有效的，不光是针对状态估计问题
- 这两步分别称为前向过程和后向过程（forward/backward），我们以例子来说明
- 下面几页会涉及到一些计算细节，部分公式以手写内容注明

离散时间的递归平滑算法

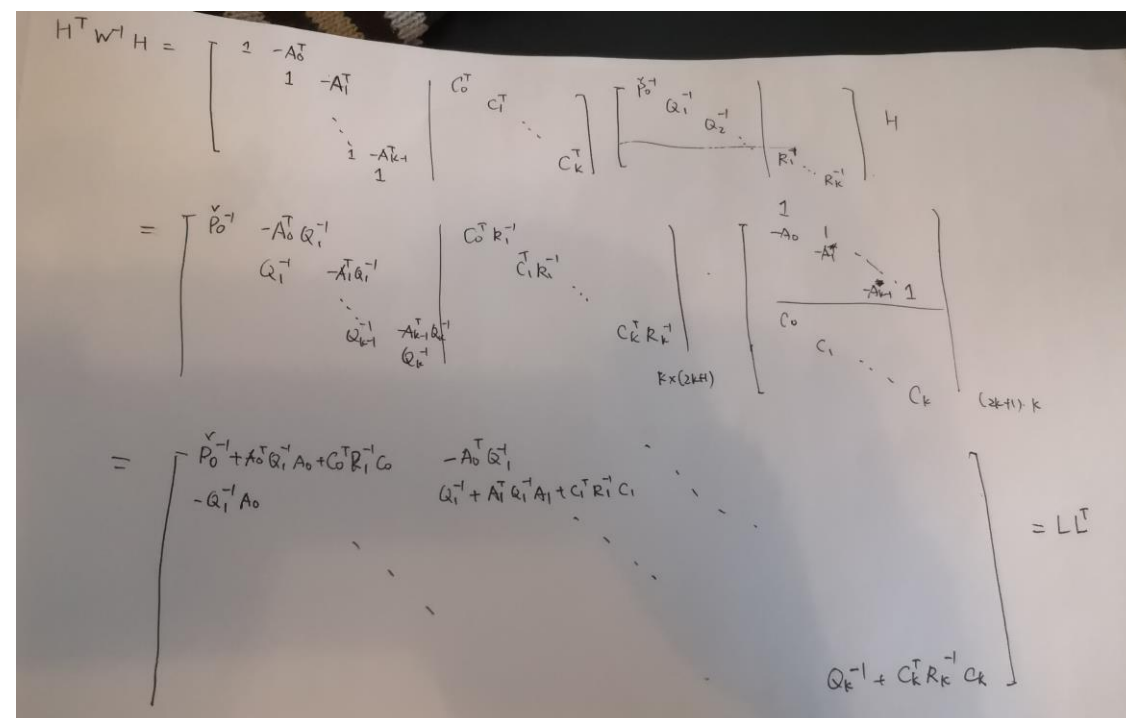
- L 的形式:

$$L = \begin{bmatrix} L_0 & & & & & \\ L_{10} & L_1 & & & & \\ & L_{21} & L_2 & & & \\ & & \ddots & \ddots & & \\ & & & L_{K-1,K-2} & L_{K-1} & \\ & & & & L_{K,K-1} & L_K \end{bmatrix}$$

第1步是Cholesky分解: $H^T W^{-1} H = LL^T$

$$H = \begin{bmatrix} 1 & & & & \\ -A_0 & 1 & & & \\ & \ddots & \ddots & & \\ & & -A_{K-1} & 1 & \\ C_0 & & & & \\ & C_1 & & & \\ & & \ddots & & \\ & & & C_K \end{bmatrix}$$

$$W = \begin{bmatrix} \tilde{P}_0 & & & & \\ & Q_1 & & & \\ & & \ddots & & \\ & & & Q_K & \\ & & & & R_0 \\ & & & & R_1 \\ & & & & & \ddots \\ & & & & & & R_K \end{bmatrix}$$



Handwritten derivation of the Cholesky decomposition for the discrete-time recursive smoothing algorithm. The derivation shows the step-by-step calculation of the elements of L and L^T , starting from the top-left corner and moving downwards and to the right. The final result is L^T , which is a block upper triangular matrix with blocks of size $(2k+1) \times k$.

□ 把这步写开, 见上

离散时间的递归平滑算法

- 对比左右块：

$$L_0 L_0^T = \underbrace{\check{P}_0^{-1} + C_0^T R_0^{-1} C_0 + A_0^T Q_1^{-1} A_0}_{I_0} \quad (3.61a)$$

$$L_{10} L_0^T = -Q_1^{-1} A_0 \quad (3.61b)$$

$$L_1 L_1^T = \underbrace{-L_{10} L_{10}^T + Q_1^{-1} + C_1^T R_1^{-1} C_1 + A_1^T Q_2^{-1} A_1}_{I_1} \quad (3.61c)$$

$$L_{21} L_1^T = -Q_2^{-1} A_1 \quad (3.61d)$$

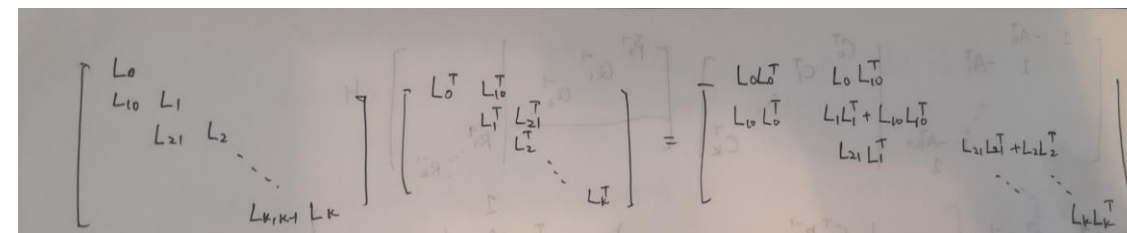
⋮

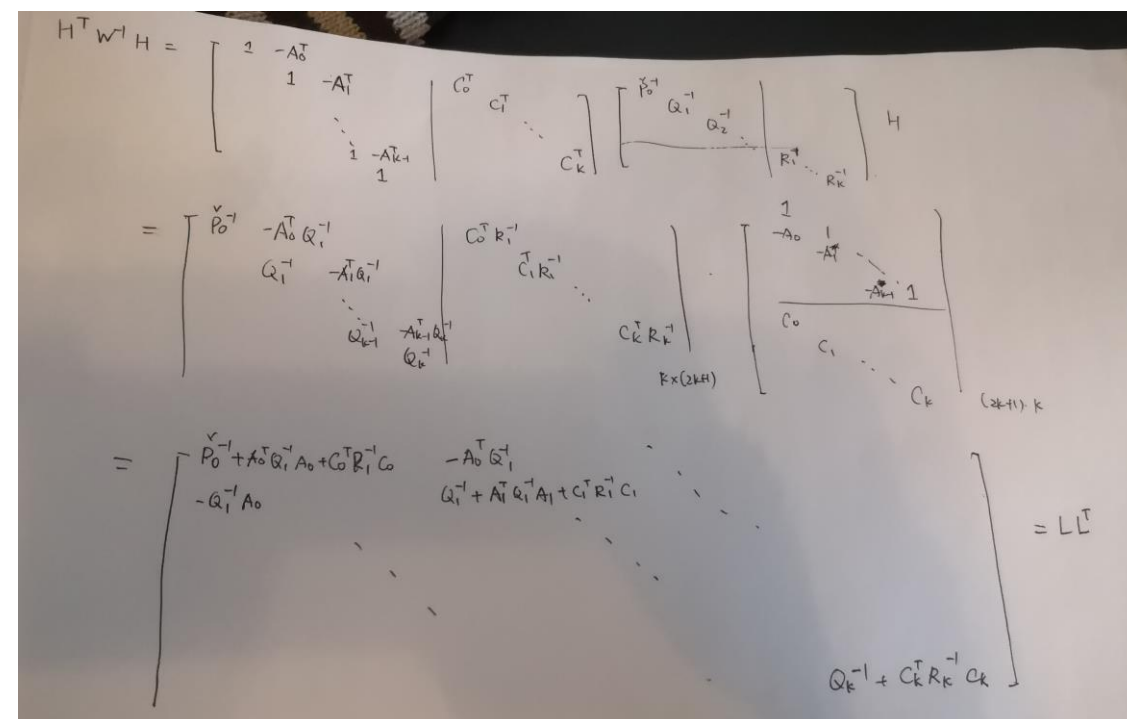
$$L_{K-1} L_{K-1}^T = \underbrace{-L_{K-1, K-2} L_{K-1, K-2}^T + Q_{K-1}^{-1} + C_{K-1}^T R_{K-1}^{-1} C_{K-1} + A_{K-1}^T Q_K^{-1} A_{K-1}}_{I_{K-1}} \quad (3.61e)$$

$$L_{K, K-1} L_{K-1}^T = -Q_K^{-1} A_{K-1} \quad (3.61f)$$

$$L_K L_K^T = \underbrace{-L_{K, K-1} L_{K, K-1}^T + Q_K^{-1} + C_K^T R_K^{-1} C_K}_{I_K} \quad (3.61g)$$

- 这里的 I 是中间变量，后面要用（且有物理意义）
- 解 L 的过程就是从上至下，把上面结果代入下方程
- 于是就解出了 L





离散时间的递归平滑算法

- 第2步，解方程 $Ld = H^T W^{-1} z$ ，未知量是 d

- 这里 q 依然是中间变量（且有物理意义）
- 这一串方程依然从上往下解，得到各个 d

$$L_0 d_0 = \underbrace{\check{P}_0^{-1} \check{x}_0 + C_0^T R_0^{-1} y_0}_{q_0} - A_0^T Q_1^{-1} v_1$$

$$L_1 d_1 = \underbrace{-L_{10} d_0 + Q_1^{-1} v_1 + C_1^T R_1^{-1} y_1}_{q_1} - A_1^T Q_2^{-1} v_2$$

$$\vdots$$

$$L_{K-1} d_{K-1} = \underbrace{-L_{K-1,K-2} d_{K-2} + Q_{K-1}^{-1} v_{K-1} + C_{K-1}^T R_{K-1}^{-1} y_{K-1}}_{q_{K-1}} - A_{K-1}^T Q_K^{-1} v_K$$

$$L_K d_K = \underbrace{-L_{K,K-1} d_{K-1} + Q_K^{-1} v_K + C_K^T R_K^{-1} y_K}_{q_K}$$

离散时间的递归平滑算法

- 第3步, 解 $L^T \hat{x} = d$
- 和前面基本一样:

$$L_K^T \hat{x}_K = d_K$$

$$L_{K-1}^T \hat{x}_{K-1} = -L_{K,K-1}^T \hat{x}_K + d_{K-1}$$

$$\vdots$$

$$L_1^T \hat{x}_1 = -L_{21}^T \hat{x}_2 + d_1$$

$$L_0^T \hat{x}_0 = -L_{10}^T \hat{x}_1 + d_0$$

- 但这个是从后往前的

总结: Cholesky方法共3步

1. 解L: $H^T W^{-1} H = L L^T$
2. 解d: $L d = H^T W^{-1} z$
3. 解x: $L^T \hat{x} = d$

第1, 2步是正向的, 第3步是反向的;

整个算法由一次正向迭代和一次反向迭代组成, 下面来归纳它的过程



离散时间的递归平滑算法

- 把中间量 l 和 q 也写进来，整理算法流程：

- 初始值： $I_0 = \check{P}_0^{-1} + C_0^T R_0^{-1} C_0$
 $q_0 = \check{P}_0^{-1} \check{x}_0 + C_0^T R_0^{-1} y_0$

- 前向迭代：

前向： $k = 1, \dots, K$

$$L_{k-1} L_{k-1}^T = I_{k-1} + A_{k-1}^T Q_k^{-1} A_{k-1}$$

$$L_{k-1} d_{k-1} = q_{k-1} - A_{k-1}^T Q_k^{-1} v_k$$

$$L_{k,k-1} L_{k-1}^T = -Q_k^{-1} A_{k-1}$$

$$I_k = -L_{k,k-1} L_{k,k-1}^T + Q_k^{-1} + C_k^T R_k^{-1} C_k$$

$$q_k = -L_{k,k-1} d_{k-1} + Q_k^{-1} v_k + C_k^T R_k^{-1} y_k$$

后向初始值： $\hat{x}_K = L_K^{-T} d_K$

后向迭代：

后向： $k = K, \dots, 1$

$$L_{k-1}^T \hat{x}_{k-1} = -L_{k,k-1}^T \hat{x}_k + d_{k-1}$$

于是就得到了递归的算法流程

这个过程等价于传统的Rauch-Tung-Striebel平滑算法，五个前向的迭代等价于著名的卡尔曼滤波器

离散时间的递归平滑算法

- Rauch-Tung-Striebel平滑算法 (RTS Smoother)

- RTS Smoother是Herber Rauch, Frank Tung, Charlotte Striebel共同提出的

- 先看前向迭代的五个式子

前向: $k = 1, \dots, K$

$$L_{k-1} L_{k-1}^T = I_{k-1} + A_{k-1}^T Q_k^{-1} A_{k-1}$$

$$L_{k-1} d_{k-1} = q_{k-1} - A_{k-1}^T Q_k^{-1} v_k$$

$$L_{k,k-1} L_{k-1}^T = -Q_k^{-1} A_{k-1}$$

$$I_k = -L_{k,k-1} L_{k,k-1}^T + Q_k^{-1} + C_k^T R_k^{-1} C_k$$

$$q_k = -L_{k,k-1} d_{k-1} + Q_k^{-1} v_k + C_k^T R_k^{-1} y_k$$

□ 由第3式解出 $L(k,k-1)$: $L_{k,k-1} = -Q_k^{-1} A_{k-1} L_{k-1}^{-T}$

□ 代入第4式: (第2行代入第1式)

$$\begin{aligned} I_k &= -(-Q_k^{-1} A_{k-1} L_{k-1}^{-T})(-L_{k-1}^{-1} A_{k-1}^T Q_k^{-T}) + Q_k^{-1} + C_k^T R_k^{-1} C_k \\ &= -(Q_k^{-1} A_{k-1} (L_{k-1} L_{k-1}^T)^{-1} A_{k-1}^T Q_k^{-T}) + Q_k^{-1} + C_k^T R_k^{-1} C_k \\ &= -(Q_k^{-1} A_{k-1} (I_{k-1} + A_{k-1}^T Q_k^{-1} A_{k-1}) A_{k-1}^T Q_k^{-T}) + Q_k^{-1} + C_k^T R_k^{-1} C_k \end{aligned}$$

□ 再使用SMW化简:

$$I_k = \underbrace{Q_k^{-1} - Q_k^{-1} A_{k-1} (I_{k-1} + A_{k-1}^T Q_k^{-1} A_{k-1})^{-1} A_{k-1}^T Q_k^{-1}}_{\text{由式 (2.75): } (A_{k-1} I_{k-1}^{-1} A_{k-1}^T + Q_k)^{-1}} + C_k^T R_k^{-1} C_k$$

□ SMW(2)式: $(D + CAB)^{-1} \equiv D^{-1} - D^{-1} C (A^{-1} + BD^{-1} C)^{-1} BD^{-1}$



离散时间的递归平滑算法

• 于是: $I_k = (A_{k-1}I_{k-1}^{-1}A_{k-1}^T + Q_k)^{-1} + C_k^T R_k^{-1} C_k$

• 设: $\hat{P}_{k,f} = I_k^{-1}$ 那么上式可写为两步:

$$\check{P}_{k,f} = A_{k-1} \hat{P}_{k-1,f} A_{k-1}^T + Q_k$$

□ 先验的

$$\hat{P}_{k,f}^{-1} = \check{P}_{k,f}^{-1} + C_k^T R_k^{-1} C_k$$

□ 后验的, 或者修正的

• 再把它写成经典卡尔曼滤波形式, 定义: $K_k = \hat{P}_{k,f} C_k^T R_k^{-1}$

• 那么上面第2式写为: $K_k = (\check{P}_{k,f}^{-1} + C_k^T R_k^{-1} C_k)^{-1} C_k^T R_k^{-1} = \check{P}_{k,f} C_k^T (C_k \check{P}_{k,f} C_k^T + R_k)^{-1}$

• 此式可用于计算K

$$\text{SMW(4): } (D + CAB)^{-1} CA \equiv D^{-1} C (A^{-1} + BD^{-1}C)^{-1}$$

• K已经算得, 那么代入上面2式, 即有:

$$\check{P}_{k,f}^{-1} = \hat{P}_{k,f}^{-1} - C_k^T R_k^{-1} C_k = \hat{P}_{k,f}^{-1} (1 - \underbrace{\hat{P}_{k,f} C_k^T R_k^{-1} C_k}_{K_k}) = \hat{P}_{k,f}^{-1} (1 - K_k C_k)$$

$$\text{即: } \hat{P}_{k,f} = (1 - K_k C_k) \check{P}_{k,f}$$

□ K为卡尔曼增益, 上式即为协方差更新

离散时间的递归平滑算法

- 回到前向迭代:

前向: $k = 1, \dots, K$

$$L_{k-1} L_{k-1}^T = I_{k-1} + A_{k-1}^T Q_k^{-1} A_{k-1}$$

$$L_{k-1} d_{k-1} = q_{k-1} - A_{k-1}^T Q_k^{-1} v_k$$

$$L_{k,k-1} L_{k-1}^T = -Q_k^{-1} A_{k-1}$$

$$I_k = -L_{k,k-1} L_{k,k-1}^T + Q_k^{-1} + C_k^T R_k^{-1} C_k$$

$$q_k = -L_{k,k-1} d_{k-1} + Q_k^{-1} v_k + C_k^T R_k^{-1} y_k$$

那么: $L_{k,k-1} d_{k-1} = -Q_k^{-1} A_{k-1} (L_{k-1} L_{k-1}^T)^{-1} (q_{k-1} - A_{k-1}^T Q_k^{-1} v_k)$

又可以代入1式:

$$q_k = \underbrace{Q_k^{-1} A_{k-1} (I_{k-1} + A_{k-1}^T Q_k^{-1} A_{k-1})^{-1}}_{\text{由式 (2.75): } (A_{k-1} I_{k-1}^{-1} A_{k-1}^T + Q_k)^{-1} A_{k-1} I_{k-1}^{-1}} q_{k-1} + \underbrace{\left(Q_k^{-1} - Q_k^{-1} A_{k-1} (I_{k-1} + A_{k-1}^T Q_k^{-1} A_{k-1})^{-1} A_{k-1}^T Q_k^{-1} \right)}_{\text{由式 (2.75): } (A_{k-1} I_{k-1}^{-1} A_{k-1}^T + Q_k)^{-1}} v_k + C_k^T R_k^{-1} y_k$$

- 由2式: $d_{k-1} = L_{k-1}^{-1} (q_{k-1} - A_{k-1}^T Q_k^{-1} v_k)$

- 由3式: $L_{k,k-1} = -Q_k^{-1} A_{k-1} L_{k-1}^{-T}$

离散时间的递归平滑算法

- 在 q_k 的表达式中代入前面的定义: $q_k = \underbrace{Q_k^{-1} A_{k-1} (I_{k-1} + A_{k-1}^T Q_k^{-1} A_{k-1})^{-1}}_{\text{由式 (2.75): } (A_{k-1} I_{k-1}^{-1} A_{k-1}^T + Q_k)^{-1} A_{k-1} I_{k-1}^{-1}}$ q_{k-1}

$$\hat{P}_{k,f} = I_k^{-1}$$

$$\check{P}_{k,f} = A_{k-1} \hat{P}_{k-1,f} A_{k-1}^T + Q_k$$

$$\hat{P}_{k,f}^{-1} = \check{P}_{k,f}^{-1} + C_k^T R_k^{-1} C_k$$

$$+ \underbrace{\left(Q_k^{-1} - Q_k^{-1} A_{k-1} (I_{k-1} + A_{k-1}^T Q_k^{-1} A_{k-1})^{-1} A_{k-1}^T Q_k^{-1} \right)}_{\text{由式 (2.75): } (A_{k-1} I_{k-1}^{-1} A_{k-1}^T + Q_k)^{-1}} v_k$$

$$+ C_k^T R_k^{-1} y_k$$

- 且令: $\hat{P}_{k,f}^{-1} \hat{x}_{k,f} = q_k$

- 于是得:

$$\check{x}_{k,f} = A_{k-1} \hat{x}_{k-1,f} + v_k$$

$$\hat{P}_{k,f}^{-1} \hat{x}_{k,f} = \check{P}_{k,f}^{-1} \check{x}_{k,f} + C_k^T R_k^{-1} y_k$$

$$q_k = \check{P}_{k,f}^{-1} A_{k-1} \hat{P}_{k-1,f} q_{k-1} + \check{P}_{k,f}^{-1} v_k + C_k^T R_k^{-1} y_k$$

$$\hat{P}_{k,f}^{-1} \hat{x}_{k,f} = \check{P}_{k,f}^{-1} A_{k-1} \hat{x}_{k-1,f} + \check{P}_{k,f}^{-1} v_k + C_k^T R_k^{-1} y_k$$

$$= \check{P}_{k,f}^{-1} \underbrace{(A_{k-1} \hat{x}_{k-1,f} + v_k)}_{\check{x}_{k,f}} + C_k^T R_k^{-1} y_k$$

离散时间的递归平滑算法

- 处理一下前面协方差逆的形式: $\hat{P}_{k,f}^{-1} \hat{x}_{k,f} = \check{P}_{k,f}^{-1} \check{x}_{k,f} + C_k^T R_k^{-1} y_k$
- 两侧左乘后验协方差矩阵: $\hat{x}_{k,f} = \underbrace{\hat{P}_{k,f} \check{P}_{k,f}^{-1}}_{1-K_k C_k} \check{x}_{k,f} + \underbrace{\hat{P}_{k,f} C_k^T R_k^{-1}}_{K_k} y_k$
- 于是得到均值更新方程: $\hat{x}_{k,f} = \check{x}_{k,f} + K_k (y_k - C_k \check{x}_{k,f})$

离散时间的递归平滑算法

- 最后来处理向后迭代

前向: $k = 1, \dots, K$

$$L_{k-1} L_{k-1}^T = I_{k-1} + A_{k-1}^T Q_k^{-1} A_{k-1}$$

$$L_{k-1} d_{k-1} = q_{k-1} - A_{k-1}^T Q_k^{-1} v_k$$

$$L_{k,k-1} L_{k-1}^T = -Q_k^{-1} A_{k-1}$$

$$I_k = -L_{k,k-1} L_{k,k-1}^T + Q_k^{-1} + C_k^T R_k^{-1} C_k$$

$$q_k = -L_{k,k-1} d_{k-1} + Q_k^{-1} v_k + C_k^T R_k^{-1} y_k$$

后向: $k = K, \dots, 1$

$$L_{k-1}^T \hat{x}_{k-1} = -L_{k,k-1}^T \hat{x}_k + d_{k-1}$$

后向式两侧乘 $L(k-1)$, 得:

$$\hat{x}_{k-1} = (L_{k-1} L_{k-1}^T)^{-1} L_{k-1} (-L_{k,k-1}^T \hat{x}_k + d_{k-1})$$

代入前三式:

$$\begin{aligned} \hat{x}_{k-1} &= (I_{k-1} + A_{k-1}^T Q_k^{-1} A_{k-1}) \cdot (-L_{k,k-1} L_{k-1}^T)^T \hat{x}_k + L_{k-1} d_{k-1} \\ &= (I_{k-1} + A_{k-1}^T Q_k^{-1} A_{k-1}) (A_{k-1}^T Q_k^{-1} \hat{x}_k + q_{k-1} - A_{k-1}^T Q_k^{-1} v_k) \\ &= (I_{k-1} + A_{k-1}^T Q_k^{-1} A_{k-1}) \cdot A_{k-1}^T Q_k^{-1} (\hat{x}_k - v_k) \\ &\quad + (I_{k-1} + A_{k-1}^T Q_k^{-1} A_{k-1}) q_{k-1} \end{aligned}$$

离散时间的递归平滑算法

• 再由SMW公式:
$$\hat{x}_{k-1} = \underbrace{(I_{k-1} + A_{k-1}^T Q_k^{-1} A_{k-1})^{-1} A_{k-1}^T Q_k^{-1}}_{\text{由式 (2.75): } I_{k-1}^{-1} A_{k-1}^T (A_{k-1} I_{k-1}^{-1} A_{k-1}^T + Q_k)^{-1}} (\hat{x}_k - v_k) + \underbrace{(I_{k-1} + A_{k-1}^T Q_k^{-1} A_{k-1})^{-1}}_{\text{由式 (2.75): } I_{k-1}^{-1} - I_{k-1}^{-1} A_{k-1}^T (A_{k-1} I_{k-1}^{-1} A_{k-1}^T + Q_k)^{-1} A_{k-1} I_{k-1}^{-1}} q_{k-1}$$

• 且有:
$$\hat{P}_{k,f} = I_k^{-1}$$

$$\begin{aligned} \check{P}_{k,f} &= A_{k-1} \hat{P}_{k-1,f} A_{k-1}^T + Q_k \\ \hat{P}_{k,f}^{-1} &= \check{P}_{k,f}^{-1} + C_k^T R_k^{-1} C_k \end{aligned}$$

$$\hat{P}_{k,f}^{-1} \hat{x}_{k,f} = q_k$$

$$\begin{aligned} \check{x}_{k,f} &= A_{k-1} \hat{x}_{k-1,f} + v_k \\ \hat{P}_{k,f}^{-1} \hat{x}_{k,f} &= \check{P}_{k,f}^{-1} \check{x}_{k,f} + C_k^T R_k^{-1} y_k \end{aligned}$$

$$\begin{aligned} \hat{x}_{k-1} &= \hat{P}_{k-1,f} A_{k-1}^T \check{P}_{k,f} (\hat{x}_k - v_k) + (\hat{P}_{k-1,f} - \hat{P}_{k-1,f} A_{k-1}^T \check{P}_{k,f} A_{k-1} \hat{P}_{k-1,f}^{-1}) \cdot \hat{P}_{k-1,f}^{-1} \hat{x}_{k-1,f} \\ &= \hat{x}_{k-1,f} + \hat{P}_{k-1,f} A_{k-1}^T \check{P}_{k,f} (\hat{x}_k - v_k - \underbrace{A_{k-1} \hat{x}_{k-1,f}}_{\check{x}_{k,f} = A_{k-1} \hat{x}_{k-1,f} + v_k}) \\ &= \hat{x}_{k-1,f} + \hat{P}_{k-1} A_{k-1}^T \check{P}_{k,f} (\hat{x}_k - \check{x}_{k,f}) \end{aligned}$$

离散时间的递归平滑算法

- 综合上面的式子，我们得到经典的RTS Smoother:

前向: $k = 1, \dots, K$

$$\check{P}_{k,f} = A_{k-1} \hat{P}_{k-1,f} A_{k-1}^T + Q_k$$

$$\check{x}_{k,f} = A_{k-1} \hat{x}_{k-1,f} + v_k$$

$$K_k = \check{P}_{k,f} C_k^T (C_k \check{P}_{k,f} C_k^T + R_k)^{-1}$$

$$\hat{P}_{k,f} = (1 - K_k C_k) \check{P}_{k,f}$$

$$\hat{x}_{k,f} = \check{x}_{k,f} + K_k (y_k - C_k \check{x}_{k,f})$$

后向: $k = K, \dots, 1$

$$\hat{x}_{k-1} = \hat{x}_{k-1,f} + \hat{P}_{k-1,f} A_{k-1}^T \check{P}_{k,f}^{-1} (\hat{x}_k - \check{x}_{k,f})$$

With 初始值:

$$\begin{aligned} \check{P}_{0,f} &= \check{P}_0 \\ \check{x}_{0,f} &= \check{x}_0 \\ \hat{x}_K &= \hat{x}_{K,f} \end{aligned}$$

- 前五个式子就是经典的卡尔曼滤波器
- 整个过程没有任何的近似
- 和批量解法本质上是一样的（核心即Cholesky分解解方程）
- 不过，卡尔曼滤波器也可以从很多其他的角度推出，此处的推导是从Cholesky解方程出发的，而卡尔曼滤波不必跟Cholesky分解挂钩

第2讲 线性高斯系统的状态估计问题

- 离散时间的批量估计
- 离散时间的递归平滑算法
- 离散时间的滤波算法
- 连续时间的批量估计

离散时间的滤波算法

- RTS Smoother是无法在线运行的（非因果的Not causal）
 - 它的后向迭代过程使用下个时刻的信息更新之前的估计
 - 初始值中需要知道 $x(K)$ 的后验
- 如果希望算法实时运行，就要去掉后向迭代的过程

平滑算法利用了所有能用到的数据来估计状态

$$\underbrace{\tilde{x}_0, y_0, v_1, y_1, v_2, y_2, \dots, v_{k-1}, y_{k-1}, v_k, y_k, v_{k+1}, y_{k+1}, \dots, v_K, y_K}_{\hat{x}_{k,f}}$$

滤波器仅利用了过去/现在的数据来估计状态

- 接下来我们来看滤波器的推导

离散时间的滤波算法

- 通过MAP推导卡尔曼滤波
- 因为只有前向过程，所以把下标中的f去掉
- 假设已有k-1时刻的状态均值和方差： $\{\hat{x}_{k-1}, \hat{P}_{k-1}\}$ ，目标是估计k时刻的状态： $\{\hat{x}_k, \hat{P}_k\}$
- 这其中需要用到k时刻的输入和观测，即 $\{\hat{x}_{k-1}, \hat{P}_{k-1}, v_k, y_k\} \mapsto \{\hat{x}_k, \hat{P}_k\}$
- 定义：

$$z = \begin{bmatrix} \hat{x}_{k-1} \\ v_k \\ y_k \end{bmatrix}, \quad H = \begin{bmatrix} 1 & & \\ -A_{k-1} & 1 & \\ & & C_k \end{bmatrix}, \quad W = \begin{bmatrix} \hat{P}_{k-1} & & \\ & Q_k & \\ & & R_k \end{bmatrix}, \quad \hat{x} = \begin{bmatrix} \hat{x}'_{k-1} \\ \hat{x}_k \end{bmatrix}$$

- 那么MAP的线性方程为： $(H^T W^{-1} H) \hat{x} = H^T W^{-1} z$

- 带撇表示用k时刻数据推出k-1时刻状态
- 最小二乘约束解释：既要符合先验，又要符合当前的运动和观测

离散时间的滤波算法

- 代入MAP最优解：

$$\begin{bmatrix} \hat{P}_{k-1}^{-1} + A_{k-1}^T Q_k^{-1} A_{k-1} & -A_{k-1}^T Q_k^{-1} \\ -Q_k^{-1} A_{k-1} & Q_k^{-1} + C_k^T R_k^{-1} C_k \end{bmatrix} \begin{bmatrix} \hat{x}'_{k-1} \\ \hat{x}_k \end{bmatrix} = \begin{bmatrix} \hat{P}_{k-1}^{-1} \hat{x}_{k-1} - A_{k-1}^T Q_k^{-1} v_k \\ Q_k^{-1} v_k + C_k^T R_k^{-1} y_k \end{bmatrix} \quad (3.110)$$

- 但是只关心 $x(k)$ ，所以对上式进行消元，把系数矩阵第2行第1列消去：

$$\begin{aligned} & \begin{bmatrix} \hat{P}_{k-1}^{-1} + A_{k-1}^T Q_k^{-1} A_{k-1} & -A_{k-1}^T Q_k^{-1} \\ 0 & Q_k^{-1} - Q_k^{-1} A_{k-1} \left(\hat{P}_{k-1}^{-1} + A_{k-1}^T Q_k^{-1} A_{k-1} \right)^{-1} \\ & \quad \times A_{k-1}^T Q_k^{-1} + C_k^T R_k^{-1} C_k \end{bmatrix} \begin{bmatrix} \hat{x}'_{k-1} \\ \hat{x}_k \end{bmatrix} \\ &= \begin{bmatrix} \hat{P}_{k-1}^{-1} \hat{x}_{k-1} - A_{k-1}^T Q_k^{-1} v_k \\ Q_k^{-1} A_{k-1} \left(\hat{P}_{k-1}^{-1} + A_{k-1}^T Q_k^{-1} A_{k-1} \right)^{-1} \left(\hat{P}_{k-1}^{-1} \hat{x}_{k-1} - A_{k-1}^T Q_k^{-1} v_k \right) \\ \quad + Q_k^{-1} v_k + C_k^T R_k^{-1} y_k \end{bmatrix} \end{aligned} \quad (3.112)$$

离散时间的滤波算法

- 于是最优解的线性方程为：

$$\underbrace{\left(Q_k^{-1} - Q_k^{-1} A_{k-1} \left(\hat{P}_{k-1}^{-1} + A_{k-1}^T Q_k^{-1} A_{k-1} \right)^{-1} A_{k-1}^T Q_k^{-1} + C_k^T R_k^{-1} C_k \right)}_{\text{由式 (2.75): } (Q_k + A_{k-1} \hat{P}_{k-1} A_{k-1}^T)^{-1}} \hat{x}_k$$

$$= Q_k^{-1} A_{k-1} \left(\hat{P}_{k-1}^{-1} + A_{k-1}^T Q_k^{-1} A_{k-1} \right)^{-1} \left(\hat{P}_{k-1}^{-1} \hat{x}_{k-1} - A_{k-1}^T Q_k^{-1} v_k \right)$$

$$+ Q_k^{-1} v_k + C_k^T R_k^{-1} y_k$$

- 定义先验和后验协方差：

$$\hat{P}_k^{-1} \hat{x}_k = Q_k^{-1} A_{k-1} \left(\hat{P}_{k-1}^{-1} + A_{k-1}^T Q_k^{-1} A_{k-1} \right)^{-1}$$

$$\times \left(\hat{P}_{k-1}^{-1} \hat{x}_{k-1} - A_{k-1}^T Q_k^{-1} v_k \right) + Q_k^{-1} v_k + C_k^T R_k^{-1} y_k$$

$$= \underbrace{Q_k^{-1} A_{k-1} \left(\hat{P}_{k-1}^{-1} + A_{k-1}^T Q_k^{-1} A_{k-1} \right)^{-1} \hat{P}_{k-1}^{-1} \hat{x}_{k-1}}_{\text{由下文: } \check{P}_k^{-1} A_{k-1}}$$

$$+ \underbrace{\left(Q_k^{-1} - Q_k^{-1} A_{k-1} \left(\hat{P}_{k-1}^{-1} + A_{k-1}^T Q_k^{-1} A_{k-1} \right)^{-1} A_{k-1}^T Q_k^{-1} \right)}_{\check{P}_k^{-1}} v_k + C_k^T R_k^{-1} y_k$$

$$= \check{P}_k^{-1} \underbrace{(A_{k-1} \hat{x}_{k-1} + v_k)}_{\hat{x}_k} + C_k^T R_k^{-1} y_k$$

$$\check{P}_k = Q_k + A_{k-1} \hat{P}_{k-1} A_{k-1}^T$$

$$\hat{P}_k = (\check{P}_k^{-1} + C_k^T R_k^{-1} C_k)^{-1}$$

- 那么：

$$Q_k^{-1} A_{k-1} \underbrace{\left(\hat{P}_{k-1}^{-1} + A_{k-1}^T Q_k^{-1} A_{k-1} \right)^{-1} \hat{P}_{k-1}^{-1}}_{\text{再次由式 (2.75)}}$$

$$= Q_k^{-1} A_{k-1} \left(\hat{P}_{k-1} - \hat{P}_{k-1} A_{k-1}^T \underbrace{\left(Q_k + A_{k-1} \hat{P}_{k-1} A_{k-1}^T \right)^{-1}}_{\check{P}_k^{-1}} \right)^{-1}$$

$$\times A_{k-1} \hat{P}_{k-1} \right) \hat{P}_{k-1}^{-1}$$

$$= \left(Q_k^{-1} - Q_k^{-1} \underbrace{A_{k-1} \hat{P}_{k-1} A_{k-1}^T}_{\check{P}_k - Q_k} \hat{P}_{k-1}^{-1} \right) A_{k-1}$$

$$= (Q_k^{-1} - Q_k^{-1} + \check{P}_k^{-1}) A_{k-1}$$

$$= \check{P}_k^{-1} A_{k-1}$$

离散时间的滤波算法

- 上式即为信息形式的卡尔曼滤波器：预测：

$$\check{P}_k = A_{k-1} \hat{P}_{k-1} A_{k-1}^T + Q_k$$

$$\check{x}_k = A_{k-1} \hat{x}_{k-1} + v_k$$

更新：

$$\hat{P}_k^{-1} = \check{P}_k^{-1} + C_k^T R_k^{-1} C_k$$

$$\hat{P}_k^{-1} \hat{x}_k = \check{P}_k^{-1} \check{x}_k + C_k^T R_k^{-1} y_k$$

- 定义卡尔曼增益： $K_k = \hat{P}_k C_k^T R_k^{-1}$

- 我们已经推过它可以写为： $K_k = \check{P}_k C_k^T (C_k \check{P}_k C_k^T + R_k)^{-1}$


[返回查看](#)

- 于是把上面式子整理为：

预测：

$$\check{P}_k = A_{k-1} \hat{P}_{k-1} A_{k-1}^T + Q_k$$

$$\check{x}_k = A_{k-1} \hat{x}_{k-1} + v_k$$

卡尔曼增益：

$$K_k = \check{P}_k C_k^T (C_k \check{P}_k C_k^T + R_k)^{-1}$$

更新：

$$\hat{P}_k = (1 - K_k C_k) \check{P}_k$$

$$\hat{x}_k = \check{x}_k + K_k \underbrace{(y_k - C_k \check{x}_k)}_{\text{更新量}}$$

- 这就是经典的卡尔曼滤波器
- 更新量可看成是一个修正量，而卡尔曼增益就是它的权重系数
- 它等价于RTS的前向过程，或等价于MAP估计
- 实际上也可以不写卡尔曼增益，直接用上方式子推出

离散时间的滤波算法

- 通过贝叶斯推断同样可以推出卡尔曼滤波器，过程更加简洁，下面推导之
- $k-1$ 时刻的后验分布 $p(x_{k-1}|\check{x}_0, v_{1:k-1}, y_{0:k-1}) = \mathcal{N}(\hat{x}_{k-1}, \hat{P}_{k-1})$
- K 时刻的先验由运动方程给出: $p(x_k|\check{x}_0, v_{1:k}, y_{0:k-1}) = \mathcal{N}(\check{x}_k, \check{P}_k)$
- 显然这是高斯分布的线性变换，于是有：

$$\check{P}_k = A_{k-1} \hat{P}_{k-1} A_{k-1}^T + Q_k$$

$$\check{x}_k = A_{k-1} \hat{x}_{k-1} + v_k$$

离散时间的滤波算法

- 对于更新部分，使用高斯推断，联合分布为：

$$\begin{aligned}
 p(x_k, y_k | \check{x}_0, v_{1:k}, y_{0:k-1}) &= \mathcal{N} \left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \right) \\
 &= \mathcal{N} \left(\begin{bmatrix} \check{x}_k \\ C_k \check{x}_k \end{bmatrix}, \begin{bmatrix} \check{P}_k & \check{P}_k C_k^T \\ C_k \check{P}_k & C_k \check{P}_k C_k^T + R_k \end{bmatrix} \right)
 \end{aligned}$$

- 那么条件分布为： $p(x_k | \check{x}_0, v_{1:k}, y_{0:k}) = \mathcal{N} \left(\underbrace{\mu_x + \Sigma_{xy} \Sigma_{yy}^{-1} (y_k - \mu_y)}_{\hat{x}_k}, \underbrace{\Sigma_{xx} - \Sigma_{xy} \Sigma_{yy}^{-1} \Sigma_{yx}}_{\hat{P}_k} \right)$

- 在结果中代入卡尔曼增益，易得：

$$\begin{aligned}
 K_k &= \check{P}_k C_k^T (C_k \check{P}_k C_k^T + R_k)^{-1} \\
 \hat{P}_k &= (1 - K_k C_k) \check{P}_k \\
 \hat{x}_k &= \check{x}_k + K_k (y_k - C_k \check{x}_k)
 \end{aligned}$$

离散时间的滤波算法

- 在滤波器问题中，我们看到了和批量问题中相同的结论：
 - MAP和贝叶斯推断给出了同样的结果，等价于卡尔曼滤波器
 - 原因依然是高斯分布的均值和模处在同一点上

离散时间的滤波算法

- 关于卡尔曼滤波器的结论
 - 卡尔曼滤波器给出了LG系统下最优线性无偏估计 (Best Linear Unbiased Estimate, BLUE)
 - 需要有初始状态
 - 卡尔曼滤波器即RTS Smoother的前向部分
 - 在非线性的场合下, 我们会使用扩展卡尔曼滤波器 (EKF), 但此时MAP、贝叶斯推断、EKF给出的结果均会不一样

离散时间的滤波算法

- 本次课程略去的内容：
 - 3.4 连续时间批量估计问题
 - 3.3.4 从增益最优化角度来看卡尔曼滤波
 - 3.3.6 误差动态过程
 - 3.3.7 存在性、唯一性和能观性



离散时间的滤波算法

- 习题：课本第1,2,6题