Kalmann Filter 程序设计报告

刘沐林

June 20, 2024

Contents

1 背景

Kalmann Filter 是在实际工程应用中常用的状态预测、估计、修正方法, 能够使用观测量和动力学模型对实际的状态量进行修正和预测. 在本工作中, Kalmann 滤波被用来预测信号在下一组数据中的状态, 从而便于后续对信号的追踪.

2 技术参数

编写语言: C++
标准: C++17
依赖: 无外部依赖
测试环境: gcc on MacOS

3 Kalmann 滤波基础

在这一章节中, 我们简要介绍编写 Kalmann 滤波程序所依据的公式, 以及 Kalmann 滤波程序的运行原理.

Kalmann 滤波程序主要涉及预测 (Predict) 和更新 (Update) 两个过程, 从而对状态量 (State) 的估计 (Estimate) 和估计的协方差 (Covariance) 进行外推和修正. 在这一过程中, 预测需要考虑系统的动力学模型, 从前一状态对下一状态的状态量估计和协方差进行外推, 而更新过程通过观测 (Measurement) 和观测误差 (Measurement Error) 的考量, 对 Kalmann 增益 (Kalmann Gain) 进行计算, 从而综合预测和观测的结果, 对状态量估计和估计的协方差进行更新.

过程涉及五个基本方程:

(1) 状态量外推方程 (State Extrapolation Equation)

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n \tag{1}$$

(2) 协方差外推方程 (Covariance Extrapolation Equation)

$$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q} \tag{2}$$

(3) 状态量修正方程 (State Update Equation)

$$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1}) \tag{3}$$

(4) 协方差修正方程 (Covariance Update Equation)

$$\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\boldsymbol{P}_{n,n-1}(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T \tag{4}$$

也可以通过带入 Kalmann 增益方程进行恒等变换得到:

$$\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\boldsymbol{P}_{n,n-1} \tag{5}$$

然而, 虽然式 (4) 和式 (5) 在数学上等价, 但是在数值计算中, 式 (5) 容易因为浮点数截断误差的原因造成数值不稳定的现象出现.

(5) Kalmann 增益方程 (Kalmann Gain Equation)

$$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n)^{-1} \tag{6}$$

3.1 基本量表述约定

已知一个系统的真实状态量为 $\boldsymbol{x}_n$, 其中下标 n 代表状态量在 t_n 时刻的标号, 后面的表述中下标均沿用这样的约定. 相应的, 对系统真实状态量的观测有 t_n 时刻的**观测 (Measurement)** 量 $\boldsymbol{z}_n$, 且观测有**观测**协方差矩阵 $\boldsymbol{R}_n$. 使用 Kalmann 滤波方法我们希望可以得到在每一个时刻 t_n 对于系统真实状态量 $\boldsymbol{x}_n$ 的**估计 (Estimate)** 及估计的协方差矩阵. 设在 t_{n-1} 时刻对系统状态量的**估计**为 $\hat{\boldsymbol{x}}_{n-1,n-1}$, 且有**估计**的协方差矩阵为 $\boldsymbol{P}_{n-1.n-1}$, 于是经过状态量外推方程 (1) 和协方差外推方程 (2) 能够得到对于 t_n 时刻状态量和协方差矩阵的**预测 (Predict)** $\hat{\boldsymbol{x}}_{n,n-1}$、$\boldsymbol{P}_{n,n-1}$, 对于下标的二元表达"$*, **$", 若 $* = **$, 则代表在 t_* 时刻的**估计**, 若 $* \neq **$, 则代表 t_{**} 时刻对 t_* 时刻状态量的**预测**值. 对于状态量的**预测**值和**预测**的协方差矩阵, 可以结合**观测**对其进行修正, 从而得到新的 t_n 时刻的估计量 $\hat{\boldsymbol{x}}_{n,n}$ 和估计的协方差 $\boldsymbol{P}_{n,n}$.

3.2 状态量外推方程

如式 (1) 所示, 有状态量外推方程:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n$$

其中, $\boldsymbol{F}$ 为状态转移矩阵 (State Transition Matrix), $\boldsymbol{G}$ 式控制矩阵 (Control Matrix). $\boldsymbol{u}_n$ 代表 t_n 时刻的输入量. 考虑过程误差 (Process Noise), 即动力学模型的误差, 状态量外推方程也可以写为:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n + \boldsymbol{w}_n \tag{7}$$

其中, $\boldsymbol{w}_n$ 是过程误差矩阵, 在实际过程中往往是无法测量得到的.

考虑系统的动力学模型，在线性力学系统下, 常常可以写成如下形式:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \tag{8}$$

其中, $\boldsymbol{A}$ 为系统的动力学矩阵 (System's Dynamics Matrix), $\boldsymbol{B}$ 为系统的输入矩阵 (Input Matrix). 通过推导容易得到:

$$\begin{aligned} \boldsymbol{F} &= e^{\boldsymbol{A}\Delta t} \\ \boldsymbol{G} &= \boldsymbol{B}\int_0^{\Delta t} e^{\boldsymbol{A}t}\mathrm{d}t \end{aligned} \tag{9}$$

其中, $\Delta t = t_n - t_{n-1}$ 为外推的步长, $e^{\boldsymbol{A}\Delta t} = \boldsymbol{I} + \boldsymbol{A}\Delta t + \frac{(\boldsymbol{A}\Delta t)^2}{2!} + \frac{(\boldsymbol{A}\Delta t)^3}{3!} + \cdots$.

对于非线性力学系统, 其动力学方程可写作:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{\mathcal{F}}(\boldsymbol{x}, t) \tag{10}$$

假设有状态量 $\boldsymbol{x}^*(t)$ 满足该动力系统, 则有:

$$\begin{aligned} \Delta\dot{\boldsymbol{x}}(t) &= \dot{\boldsymbol{x}}(t) - \dot{\boldsymbol{x}}^*(t) \\ &= \boldsymbol{\mathcal{F}}(\boldsymbol{x}, t) - \boldsymbol{\mathcal{F}}(\boldsymbol{x}^*, t) \\ &= \frac{\partial \boldsymbol{\mathcal{F}}}{\partial \boldsymbol{x}}|_{\boldsymbol{x}=\boldsymbol{x}^*}\Delta\boldsymbol{x} + \mathcal{O}[(\Delta\boldsymbol{x})^2] \end{aligned}$$

将两边同时除去 $\Delta\boldsymbol{x}(t_0) = \Delta\boldsymbol{x}_0$, 有:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}_0} = \frac{\partial \boldsymbol{\mathcal{F}}}{\partial \boldsymbol{x}}|_{\boldsymbol{x}=\boldsymbol{x}^*}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}_0} \tag{11}$$

即有状态转移矩阵 $\boldsymbol{F} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}_0}$ 的方程:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{F} = \frac{\partial \boldsymbol{\mathcal{F}}}{\partial \boldsymbol{x}}|_{\boldsymbol{x}=\boldsymbol{x}^*}\boldsymbol{F} \tag{12}$$

将状态转移矩阵初值设为 $\boldsymbol{I}$ 从 t_0 时刻开始积分, 则得到任意 t 时刻的状态转移矩阵.

3.3 协方差外推方程

如式 (2) 所示, 有协方差外推方程:

$$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q}$$

其中, 下标的记法与状态量外推方程中的表述一致, $\boldsymbol{Q}$ 是过程噪声协方差矩阵 (Process Noise Covariance), 有 $\boldsymbol{Q} = \mathrm{E}(\boldsymbol{w}_n\boldsymbol{w}_n^T)$. 对于过程噪声的方针, 主要有两种方法: 离散噪声模型 (Discrete Noise Model) 和 (Continuous Noise Model). 在外推的开始, 可以将协方差外推方程设为 $\boldsymbol{I}$, 即单位矩阵.

3.4 状态量修正方程

如式 (3) 所示, 有状态量修正方程:

$$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1})$$

其中, $\boldsymbol{K}_n$ 为 t_n 时刻的 Kalmann 增益矩阵, $\boldsymbol{H}$ 为观测矩阵. 对于某一时刻系统的真实状态量 $\boldsymbol{x}_n$, 对它的观测有观测方程:

$$\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n + \boldsymbol{v}_n \tag{13}$$

其中, $\boldsymbol{v}_n$ 是观测误差.

3.5 协方差修正方程

如式 (4) 所示, 有协方程修正方程:

$$\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\boldsymbol{P}_{n,n-1}(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$$

其中, $\boldsymbol{R}_n$ 为 t_n 时刻的观测误差 (Measurement Error) 协方差矩阵, 有: $\boldsymbol{R}_n = \mathrm{E}(\boldsymbol{v}_n\boldsymbol{v}_n^T)$.

3.6 Kalmann 增益方程

如式 (6) 所示, 有 Kalmann 增益方程:

$$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n)^{-1}$$

方程反映了当前**估计**的误差情况和**观测**误差的权重选择情况. 若观测较为准确, 则观测量获得较大权重, 反之, **预测**和**估计**获得较大权重.

3.7 总结

Table 1: Kalmann 滤波五个基本方程

方程	说明
$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n$	状态量外推方程
$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q}$	协方差外推方程
$\hat{\boldsymbol{x}}_{n,n} = \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1})$	状态量修正方程
$\boldsymbol{P}_{n,n} = (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\boldsymbol{P}_{n,n-1}(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T$	协方差修正方程
$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n)^{-1}$	卡尔曼增益方程

Table 2: Kalmann 滤波参量

方程	说明
$\hat{\boldsymbol{x}}_{n+1,n}$	由 t_n 时刻状态量得到的 t_{n+1} 时刻状态量的**预测**
$\hat{\boldsymbol{x}}_{n,n}$	t_n 时刻对系统状态量的**估计**
$\boldsymbol{u}_n$	t_n 时刻的控制输入
$\boldsymbol{A}$, $\boldsymbol{B}$	系统满足的动力学方程: $\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t)$ (线性力学系统)
$\boldsymbol{F} = e^{\boldsymbol{A}\Delta t}$	状态转移矩阵 (线性力学系统)
$\boldsymbol{G} = \boldsymbol{B}\int_0^{\Delta t} e^{\boldsymbol{A}t}\mathrm{d}t$	控制矩阵 (线性力学系统)
$\boldsymbol{P}_{n+1,n}$	由 t_n 时刻的状态量协方差矩阵得到的 t_{n+1} 时刻的**预测**
$\boldsymbol{P}_{n,n}$	t_n 时刻对系统状态量**估计**的协方差矩阵
$\boldsymbol{Q} = \mathrm{E}(\boldsymbol{w}_n\boldsymbol{w}_n^T)$	过程噪声协方差
$\boldsymbol{z}_n$	t_n 时刻的观测量
$\boldsymbol{H}$	观测矩阵, 满足: $\boldsymbol{z}_n = \boldsymbol{H}\boldsymbol{x}_n + \boldsymbol{v}_n$
$\boldsymbol{R}_n = \mathrm{E}(\boldsymbol{v}_n\boldsymbol{v}_n^T)$	观测噪声协方差

4 代码实现

为便于后续使用和扩展, 程序充分利用 C++ 模版, 构建了含有一个模版参数的 KalmanFilter 类.

```
template <typename T>
class KalmanFilter
{
private:
    /// @brief Supplementary functions
    class SupplementFunctions
    {
      static std::vector<T> vector_minus(const std::vector<T> &vector1, const std::vector<T> &vector2);

    static std::vector<T> vector_plus(const std::vector<T> &vector1, const std::vector<T> &vector2);

    static std::vector<T> matrix_multiply_vector(const std::vector<std::vector<T>> &matrix, const std::vector<T> &vector);

    static std::vector<std::vector<T>> matrix_multiply_matrix(const std::vector<std::vector<T>> &matrix1, const std::vector<std::vector<T>> &matrix2);

    static std::vector<std::vector<T>> matrix_transpose(const std::vector<std::vector<T>> &matrix);
```

```
17
18      static std::vector<std::vector<T>> matrix_plus_matrix(const std::
        vector<std::vector<T>> &matrix1, const std::vector<std::vector<T>> &
        matrix2);
19
20      static std::vector<std::vector<T>> matrix_inverse(const std::vector
        <std::vector<T>> &matrix, unsigned int order);
21      };
22
23      /// @brief Estimated state
24      std::vector<T> mEstimateState;
25
26      /// @brief Covariance of the estimated state
27      std::vector<std::vector<T>> mEstimateStateCovariance;
28
29      /// @brief Pridicted state
30      std::vector<T> mPredictState;
31
32      /// @brief Covariance of the predicted state
33      std::vector<std::vector<T>> mPredictStateCovariance;
34
35      /// @brief If the next state is predicted
36      bool mNextStatePredicted;
37
38  public:
39      /// @brief The constructor without argument is deleted
40      KalmanFilter() = delete;
41
42      /// @brief The constructor that initiates the Kalman filter
43      /// @param initialState The initial state vector
44      /// @param initialStateCovariance The initial state covariance
45      KalmanFilter(const std::vector<T> &initialState,
46                   const std::vector<std::vector<T>> &
        initialStateCovariance);
47
48      /// @brief To calculate the next predicted state vector and
        covariance matrix
49      /// @param stateTransitionMatrix The state transition matrix
50      /// @param processNoiseCovariance The process noise covariance
        matrix
51      void predict(const std::vector<std::vector<T>> &
        stateTransitionMatrix,
52                   const std::vector<std::vector<T>> &
        processNoiseCovariance);
53
54      /// @brief To calculate the next predicted state vector and
```

```
    covariance matrix. Input control is considered.
55  /// @param stateTransitionMatrix The state transition matrix
56  /// @param processNoiseCovariance The process noise covariance
    matrix
57  /// @param controlMatrix The control matrix
58  /// @param controlInput The control input vector
59  void predict(const std::vector<std::vector<T>> &
    stateTransitionMatrix,
60               const std::vector<std::vector<T>> &
    processNoiseCovariance,
61               const std::vector<std::vector<T>> &controlMatrix,
62               const std::vector<T> &controlInput);
63
64  /// @brief To update the estimated state and estimated state
    covariance based on former predicted
65  ///        state and state covariance. The predict step should be
    called before this function is
66  ///        implemented.
67  /// @param measurementVector The measurement vector
68  /// @param measurementMatrix The measurement matrix
69  /// @param measurementNoiseCovariance The measurement noise
    covariance
70  void update(const std::vector<T> &measurementVector,
71              const std::vector<std::vector<T>> &measurementMatrix,
72              const std::vector<std::vector<T>> &
    measurementNoiseCovariance);
73
74  /// @brief To get the estimate state
75  /// @return
76  std::vector<T> getEstimateState();
77
78  /// @brief To get the next predicted state
79  /// @return
80  std::vector<T> getPredictState();
81
82  /// @brief To set the initial state and the initial state
    covariance
83  /// @param initialState
84  /// @param initialStateCovariance
85  void setInitialState(const std::vector<T> &initialState,
86                       const std::vector<std::vector<T>> &
    initialStateCovariance);
87 };
```

模版参数"T" 代表计算所使用的数据类型.

4.1 构造函数

构造函数删除了默认的无参数构造函数, 仅保留了含有输入参数的构造函数:

```
1     /// @brief The constructor that initiates the Kalman filter
2     /// @param initialState The initial state vector
3     /// @param initialStateCovariance The initial state covariance
4     KalmanFilter(const std::vector<T> &initialState,
5                  const std::vector<std::vector<T>> &
      initialStateCovariance);
```

输入参数初始化了 Kalman Filter 的初始状态 ($\hat{\boldsymbol{x}}_{0,0}$) 和初始的状态量协方差 ($\boldsymbol{P}_{0,0}$).

4.2 Predict 函数

成员函数 predict 用于计算基于当前估计状态 $\hat{\boldsymbol{x}}_{n,n}$ 和当前估计状态协方差 $\boldsymbol{P}_{n,n}$ 的对于下一状态及协方差的预测 $\hat{\boldsymbol{x}}_{n+1,n}$, $\boldsymbol{P}_{n+1,n}$. 外推过程利用了式 (1) 和式 (2) 定义的状态量外推方程和协方差外推方程:

$$\hat{\boldsymbol{x}}_{n+1,n} = \boldsymbol{F}\hat{\boldsymbol{x}}_{n,n} + \boldsymbol{G}\boldsymbol{u}_n$$
$$\boldsymbol{P}_{n+1,n} = \boldsymbol{F}\boldsymbol{P}_{n,n}\boldsymbol{F}^T + \boldsymbol{Q}$$

对于不同的输入参数, Predict 函数又对无控制和有控制的过程进行了区分:

```
1     /// @brief To calculate the next predicted state vector and
      covariance matrix
2     /// @param stateTransitionMatrix The state transition matrix
3     /// @param processNoiseCovariance The process noise covariance
      matrix
4     void predict(const std::vector<std::vector<T>> &
      stateTransitionMatrix,
5                  const std::vector<std::vector<T>> &
      processNoiseCovariance);
6
7     /// @brief To calculate the next predicted state vector and
      covariance matrix. Input control is considered.
8     /// @param stateTransitionMatrix The state transition matrix
9     /// @param processNoiseCovariance The process noise covariance
      matrix
10    /// @param controlMatrix The control matrix
11    /// @param controlInput The control input vector
```

```
12      void predict(const std::vector<std::vector<T>> &
    stateTransitionMatrix,
13                   const std::vector<std::vector<T>> &
    processNoiseCovariance,
14                   const std::vector<std::vector<T>> &controlMatrix,
15                   const std::vector<T> &controlInput);
```

对于无控制的过程, 只需要输入状态转移矩阵和过程误差矩阵; 对于有控制的过程, 需要额外输入控制矩阵和控制输入向量.

4.3 Update 函数

成员函数 update 用于计算基于预测状态 $\hat{\boldsymbol{x}}_{n,n-1}$、预测状态协方差 $\boldsymbol{P}_{n,n-1}$ 以及当前时刻观测量 $\boldsymbol{z}_n$ 及相应观测参数的当前状态的估计 $\hat{\boldsymbol{x}}_{n,n}$ 和当前状态协方差的估计 $\boldsymbol{P}_{n,n}$. 在计算过程中, 需要先调用 predict 函数, 对 predict 值进行更新:

```
1       /// @brief To update the estimated state and estimated state
    covariance based on former predicted
2       ///        state and state covariance. The predict step should be
    called before this function is
3       ///        implemented.
4       /// @param measurementVector The measurement vector
5       /// @param measurementMatrix The measurement matrix
6       /// @param measurementNoiseCovariance The measurement noise
    covariance
7       void update(const std::vector<T> &measurementVector,
8                   const std::vector<std::vector<T>> &measurementMatrix,
9                   const std::vector<std::vector<T>> &
    measurementNoiseCovariance)
```

计算时, 首先利用式 (6) 计算 Kalman 增益:

$$\boldsymbol{K}_n = \boldsymbol{P}_{n,n-1}\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}_{n,n-1}\boldsymbol{H}^T + \boldsymbol{R}_n)^{-1}$$

再通过式 (3) 和 (4):

$$\begin{aligned}\hat{\boldsymbol{x}}_{n,n} &= \hat{\boldsymbol{x}}_{n,n-1} + \boldsymbol{K}_n(\boldsymbol{z}_n - \boldsymbol{H}\hat{\boldsymbol{x}}_{n,n-1})\\ \boldsymbol{P}_{n,n} &= (\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})\boldsymbol{P}_{n,n-1}(\boldsymbol{I} - \boldsymbol{K}_n\boldsymbol{H})^T + \boldsymbol{K}_n\boldsymbol{R}_n\boldsymbol{K}_n^T\end{aligned}$$

对预测状态量和预测状态量协方差进行修正, 得到对状态量的估计和相应估计协方差矩阵.

5 测试算例

5.1 测量楼的高度

在实际运用中, 通常不需要使用 Kalman 滤波的方式去测量一栋楼的高度. 因为使用高度计, 或三角测距的方法已经可以将一栋楼的高度测量得比较准确了. 我们这里只是通过一个简单的算例来验证我们算法以及代码的准确性.

设楼的高度为 50.0 m, 且一般楼的高度不会变化 (至少短期内不会). 于是容易得到关于楼的高度在每两次测量之间的状态转移矩阵有:

$$\boldsymbol{F} = \begin{pmatrix} 1 \end{pmatrix}$$

设我们对这栋楼的高度有一个简单的目视的估计为 60.0 m, 于是有初值 $\hat{\boldsymbol{x}}_{0,0} = (60.0)^T$ 且初值估计的协方差为 $\boldsymbol{P}_{0,0} = (100.0)$, 即认为初值的误差为 $\sigma_0 = 10.0$ m. 每一次通过对楼的观测, 我们可以使用 Kalman 滤波的方式逼近真实的结果. 观测矩阵 $\boldsymbol{H} = (1.0)$, 观测噪声协方差可以写为 $\boldsymbol{R} = (9.0)$, 即观测结果标准差为 $\sigma_H = 3.0$ m (这是一个偏大的值, 但是 Kalman 滤波结果更好看). 表 3总结用到的所有参数:

Table 3: 测量楼的高度数值实验参数表

参数	说明
$\boldsymbol{F} = \begin{pmatrix} 1 \end{pmatrix}$	状态转移矩阵
$\boldsymbol{Q} = \begin{pmatrix} 0.0 \end{pmatrix}$	过程误差协方差
$\boldsymbol{H} = \begin{pmatrix} 1.0 \end{pmatrix}$	观测矩阵
$\boldsymbol{R} = \begin{pmatrix} 9.0 \end{pmatrix}$	观测误差协方差
$\hat{\boldsymbol{x}}_{0,0} = \begin{pmatrix} 60.0 \end{pmatrix}^T$	初始状态量估计
$\boldsymbol{P}_{0,0} = \begin{pmatrix} 100.0 \end{pmatrix}$	初始状态量协方差

经过 50 次观测, 得到结果如图 1所示. 虽然楼的高度的测量误差较大, 但是随着观测次数的增加, Kalman 滤波的结果逐渐逼近真实值 (这与我们的期望一致, 事实上, 这里的算法和计算平均值没有根本上的区别). 另一方面, 置信区间 (即估计状态量协方差) 也逐渐缩小, 观测在预测中的占比也逐渐缩小 (从 Kalman 增益公式可以看出).

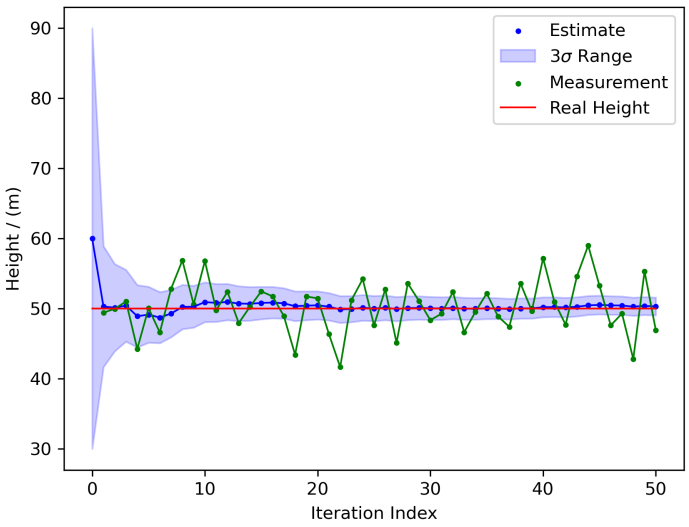

Figure 1: 测量楼的高度结果

5.2 匀速运动的载具

设有一个载具在 x-y 平面内做匀速运动, 其运动有:

$$
\begin{aligned}
x &= 0.0 + 1.0t \\
y &= 0.0 + 1.2t \\
\dot{x} &= 1.0 \\
\dot{y} &= 1.2
\end{aligned}
$$

于是容易得到, 对于其状态量 $\boldsymbol{x} = (x, y, \dot{x}, \dot{y})^T$ 有状态转移矩阵:

$$
\boldsymbol{F} = \begin{pmatrix}
1.0 & 0.0 & \Delta t & 0.0 \\
0.0 & 1.0 & 0.0 & \Delta t \\
0.0 & 0.0 & 1.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 1.0
\end{pmatrix}
$$

且对于该力学模型, 没有过程误差, 即:

$$
\boldsymbol{Q} = \begin{pmatrix}
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0
\end{pmatrix}
$$

考察对载具状态的观测, 我们假设我们只可以观测到载具的位置 (事实上, 速度信息在实际工程应用中也是容易得到的, 但是这里我们不使用速度信息), 即有观测矩阵:

$$\boldsymbol{H} = \begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \end{pmatrix}$$

且假设观测误差有 $\sigma_x = 1.0$ m, $\sigma_y = 1.0$ m, 于是观测误差协方差矩阵有:

$$\boldsymbol{R} = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$$

假设初始的状态量估计和状态量估计协方差为:

$$\hat{\boldsymbol{x}}_{0,0} = (2.0, 2.0, 2.0, 2.0)^T$$
$$\boldsymbol{P}_{0,0} = \begin{pmatrix} 4.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 4.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$$

采取观测间隔为 1 s, 得到结果如图 2所示. 即使没有对速度的观测结果, Kalman 滤波对于状态量的修正依旧得到了较为准确的位置和速度值. 这一结果基于我们对于系统动力学模型的充分了解, 在下一算例中我们将展示模型不够精确时的结果.

5.3 做匀加速直线运动的载具

设有一个载具在 x-y 平面内做匀加速直线运动, 且有:

$$\begin{aligned} x &= 0.0 + 1.0t + \frac{1}{2}0.5t^2 \\ y &= 0.0 + 1.2t + \frac{1}{2}0.3t^2 \\ \dot{x} &= 1.0 + 0.5t \\ \dot{y} &= 1.2 + 0.3t \end{aligned}$$

仿照上例构建数值实验, 参数如表 4所示. 选取观测间隔为 1 s, 得到结果如图 3所示. 容易看到, 虽然估计结果的协方差在减小, 但是真实值并不在置信区间内. 这是由于所选取的动力学模型 (状态转移矩阵) 不合适所引起的, 被称为 **Lag Error**. 于是想到通过修改过程噪声矩阵来尝试修正这一问题. 考虑匀加速直线运动, 最主要

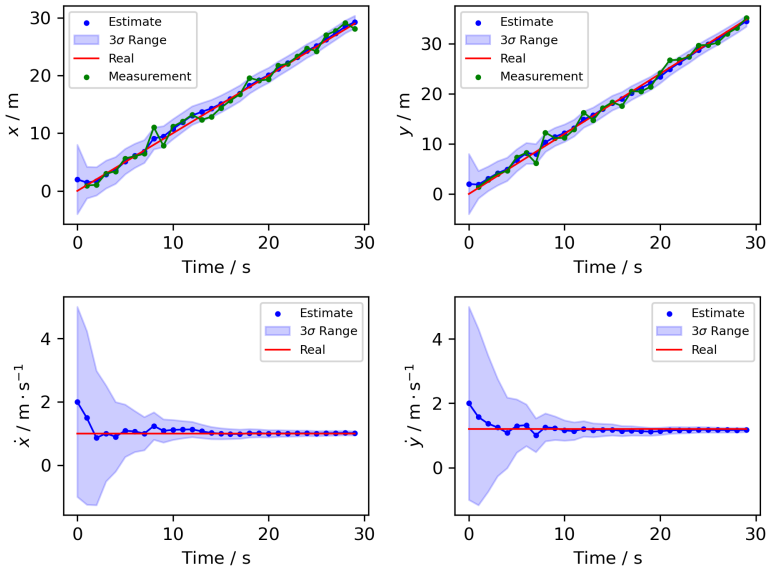

Figure 2: 匀速运动载具观测更新结果

的误差来自于加速度对速度的影响, 于是简单对过程噪声矩阵进行估计:

$$\boldsymbol{Q} = \begin{pmatrix} 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.25 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.09 \end{pmatrix}$$

注意, 这里的估计是很粗略的, 因为实际上过程误差对位置也有影响, 且位置速度的误差并不独立, 即矩阵并非只有对角线元素. 粗略的估计下得到的结果如图 4所示. 虽然置信区间受过程误差影响不再收敛, 但是预测结果准确性明显提升.

当然, 更好的方法是修正动力学模型, 应该将加速度加入载具的状态量中. 在这一例中, 准确的状态转移矩阵写为:

$$\boldsymbol{F} = \begin{pmatrix} 1.0 & 0.0 & \Delta t & 0.0 & \frac{1}{2}\Delta t^2 & 0.0 \\ 0.0 & 1.0 & 0.0 & \Delta t & 0.0 & \frac{1}{2}\Delta t^2 \\ 0.0 & 0.0 & 1.0 & 0.0 & \Delta t & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & \Delta t \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$$

对应状态量向量 $\boldsymbol{x} = (x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y})^T$. 同样考虑只能观测到位置信息的情况, 于是有输入参数如表 5所示. 所得结果如图 5所示, 结果准确且置信区间收敛.

Table 4: 观测匀加速直线运动的载具数值实验参数表 (使用四参数估计)

参数	说明
$\boldsymbol{F} = \begin{pmatrix} 1.0 & 0.0 & \Delta t & 0.0 \\ 0.0 & 1.0 & 0.0 & \Delta t \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$	状态转移矩阵
$\boldsymbol{Q} = \begin{pmatrix} 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$	过程误差协方差
$\boldsymbol{H} = \begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \end{pmatrix}$	观测矩阵
$\boldsymbol{R} = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$	观测误差协方差
$\hat{\boldsymbol{x}}_{0,0} = (2.0, 2.0, 2.0, 2.0)^T$	初始状态量估计
$\boldsymbol{P}_{0,0} = \begin{pmatrix} 4.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 4.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$	初始状态量协方差

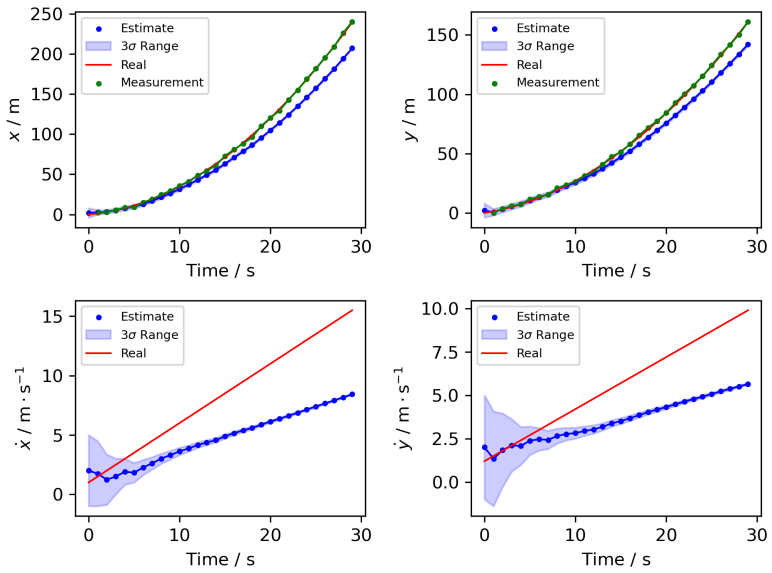

Figure 3: 匀加速运动载具观测更新结果 (使用四参数估计)

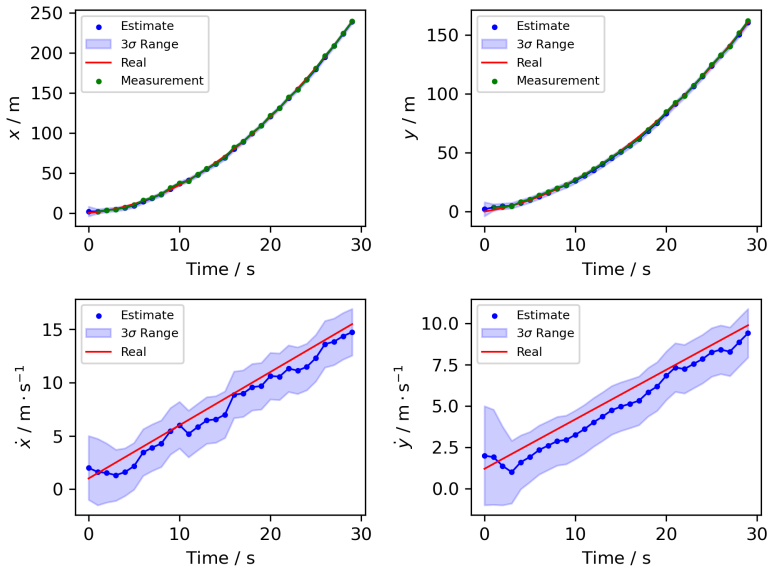

Figure 4: 匀加速运动载具观测更新结果 (使用四参数估计, 并添加过程噪声)

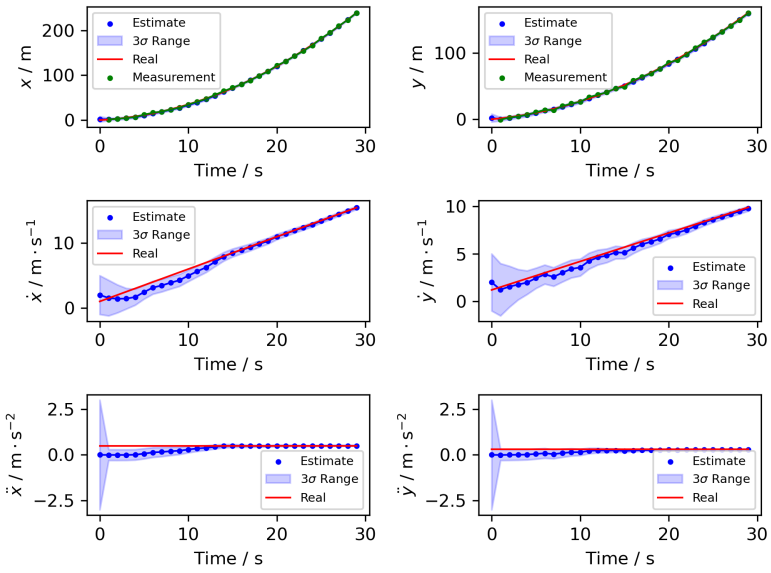

Figure 5: 匀加速运动载具观测更新结果 (使用六参数估计)

Table 5: 观测匀加速直线运动的载具数值实验参数表 (使用六参数估计)

参数	说明
$\boldsymbol{F} = \begin{pmatrix} 1.0 & 0.0 & \Delta t & 0.0 & \frac{1}{2}\Delta t^2 & 0.0 \\ 0.0 & 1.0 & 0.0 & \Delta t & 0.0 & \frac{1}{2}\Delta t^2 \\ 0.0 & 0.0 & 1.0 & 0.0 & \Delta t & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & \Delta t \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$	状态转移矩阵
$\boldsymbol{Q} = \begin{pmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$	过程误差协方差
$\boldsymbol{H} = \begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$	观测矩阵
$\boldsymbol{R} = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$	观测误差协方差
$\hat{\boldsymbol{x}}_{0,0} = (2.0, 2.0, 2.0, 2.0, 0.0, 0.0)^T$	初始状态量估计
$\boldsymbol{P}_{0,0} = \begin{pmatrix} 4.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 4.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$	初始状态量协方差

6 总结

本文介绍了 Kalman 滤波程序的基本原理和实现, 并提供了诸多测试案例供读者学习. 若对代码或案例有任何疑问, 可咨询作者: ajax0514@hotmail.com