第 2 课 动态系统状态估计

作者：范泽宣

在上一节课中，你和 Dsy 一同推导了固定量的递推方法，你发现这种推演得到的滤波效果很好，你打算把你的钢铁侠盔甲改造的更加完善，增加更多的功能。你想在从高处跳下的整个过程都可控（还记得你的计划吧，新年那天要实现钢铁侠的飞行梦），首先你在你家的仓库中敲敲打打，把钢铁侠系统完善了，增加了更多的动力装置，加入了陀螺仪、加速度计、GPS 传感器、角速度传感器等一大堆测控传感器。

有了更多的传感器系统，你就有了控制的基础，钢铁侠盔甲也就有了感知能力，可是更加精确的控制整个跳下过程，就需要控制量的加入，上一节课中你了解到的固定值的最优估计中没有这个可控量。你又找来了 Dsy 帮忙。

“好的，还记得上节课最后提到的动态系统的状态估计吗，我先帮你从程序入手，然后再告诉你程序背后的数学秘密吧，哈哈”。Dsy 微笑的对你说。

首先让我们重新回顾第一节课的两个公式：

$$X_k = a\,X_{k-1} \qquad (1.1)$$

$$Z_k = X_k + V_k \qquad (1.2)$$

第一个公式是整个钢铁侠盔甲系统的下落模型，现在你想在下落过程中控制整个系统，比如完成难度系数 3.0 的抱膝旋转 360 度，或是简单的一边下落一边前进。所以你需要重构 1.1，加入控制变量。

$$X_k = a\,X_{k-1} + bU_k \qquad (2.1)$$

你在新的系统中加入了控制量 U 和比例 b。

并完善 1.2，加入比例 c。

$$Z_k = c\,X_k + V_k \qquad (2.2)$$

所以，第一节课中的卡尔曼公式完善如下：

Predict:

$$X_k = a\,X_{k-1} + bU_k$$

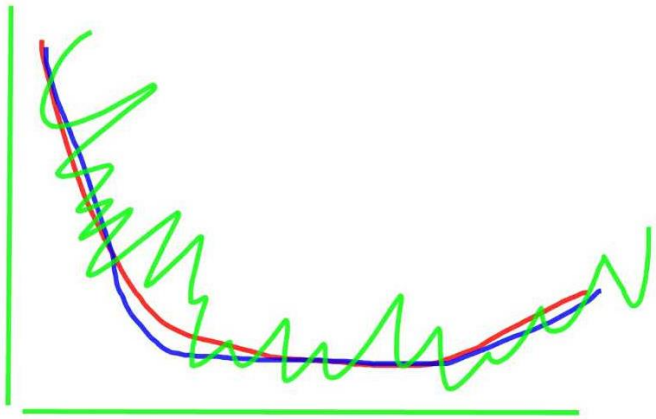
$$P_k = a*P_k*a$$

Update:

$$g_k = p_{k-1}\,c \,/\, (c\,p_{k-1}\,c + r)$$

$$\hat{x}_k = \hat{x}_{k-1} + g_k(z_k - c\hat{x}_{k-1})$$

$$p_k = (1 - g_k c)p_{k-1}$$



你按照公式重新编写程序，发现红色的 kalman 滤波值非常接近于蓝色的真实值，而绿色的传感器直接测量值几乎没办法使用，所以 kalman 的滤波效果可谓完美。

一切复杂的系统都是在使用简单模型的基础上发展推广的，上一节你们已经实现了 1.1 式的建立。

$$\text{altitude}_{\text{current\_time}} = 0.98*\text{altitude}_{\text{previous\_time}} \qquad (1.1)$$

$$X_k = a\,X_{k-1} \qquad (1.1)$$

由于你在钢铁侠系统中加入了丰富的度传感器系统，可以返回速度，加速度值，为了同时进行滤波。怎么把 1 维的位置系统扩展成多维又符合熟悉的 1.1 模型呢，大学的基础课线性代数帮上了忙。

聪明的你发现只要将 1.1 中的 x 扩展成矩阵，以及变量 a 扩展成 A(矩阵)。

$$x_k = \begin{bmatrix} \text{distance} \\ velocity \\ acceleration \end{bmatrix} \tag{2.3}$$

$$\mathrm{A} = \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

新的 1.1 变成了 $X_k = A X_{k-1}$ （2.5），其实只是每个变量变成了矩阵形式，扩展的 2.5 表达式如下：

$$\begin{bmatrix} \text{distance}_k \\ velocity_k \\ acceleration_k \end{bmatrix} = \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \text{distance}_{k-1} \\ velocity_{k-1} \\ acceleration_{k-1} \end{bmatrix}$$

按照线性代数的知识展开，就可以看到熟悉的高中物理公式了。而把所有一直掌握的 kalman 滤波的公式中一维变量都延伸成矩阵形式就可以得到动态系统中的 kalman 公式了。

Predict:

$$X_k = A X_{k-1} + BU_k$$

$$P_k = A * P_k * A$$

Update:

$$G_k = P_{k-1} C / (C P_{k-1} C + R)$$

$$\hat{x}_k = \hat{x}_{k-1} + G_k(z_k - C\hat{x}_{k-1})$$

$$P_k = (I - G_k c) P_{k-1}$$

这种方法很简单就是把所有的变量变成矩阵形式就可以进行多维的扩展了。但是这种多维系统的数学推演是怎样的呢。

你和 Dsy 进行了详细的理论推演，Dsy 告诉你想要理解动态系统的状态估计首先要做一些假设，因为现实中的模型太过复杂，为了简化成数学的形式。我们假设我们钢铁侠系统是输入信号已知，噪声期望值为 0(W 为系统噪声)的系统。

$$\hat{x}_{(k+1)|k} = A\hat{x}_{k|k} + BU_k + GW \tag{2.6}$$

由于估计误差的期望值为 0，且状态与噪声相互独立，因此估计误差的协方差可以化简为：

$$P(k+1|k) = E\left\{\left(X(k+1)\middle|Y(k) - \hat{x}_{(k+1)|k}\right)\left((X(k+1)|Y(k) - \hat{x}_{(k+1)|k}\right)^T\right\}$$

$$= AE\left\{\left(X(kT) - \hat{x}_{k|k}\right)\left(X(kT) - \hat{x}_{k|k}\right)^T\right\} A^T + GQG^T$$

再简化：

$$P(k+1|k) = AP(k|k)A^T + GQG^T$$

$$P(k|k)= E\left\{\left(X(kT) - \hat{x}_{k|k}\right)\left(X(kT) - \hat{x}_{k|k}\right)^T\right\}$$

这就是估计误差的协方差，我们推导出来了，从而可以得出 k+1 时刻的估计值。可是 Dsy 问你，现在我们知道了新的测量时，怎么合并改善估计呢，这难不倒你，你很快写出了 k+1 时刻获得测量，新的估计是：

$$E\{X((k+1)T)|Y(k+1)\}= \hat{x}_{(k+1)|k}+E\{(X(k+1)T)- \hat{x}_{(k+1)|k})|(Y_{k+1} - H\hat{x}_{(k+1)|k})\}$$

注意到，在等式右边条件期望所有量都是零均值的情况，所以可以简化为：

$$K(k+1) = P(k+1|k)H^T[HP(k+1|k)H^T + R]^{-1}$$

$$P(k+1|k+1)=[I-K(k+1)H]P(k+1|k)$$

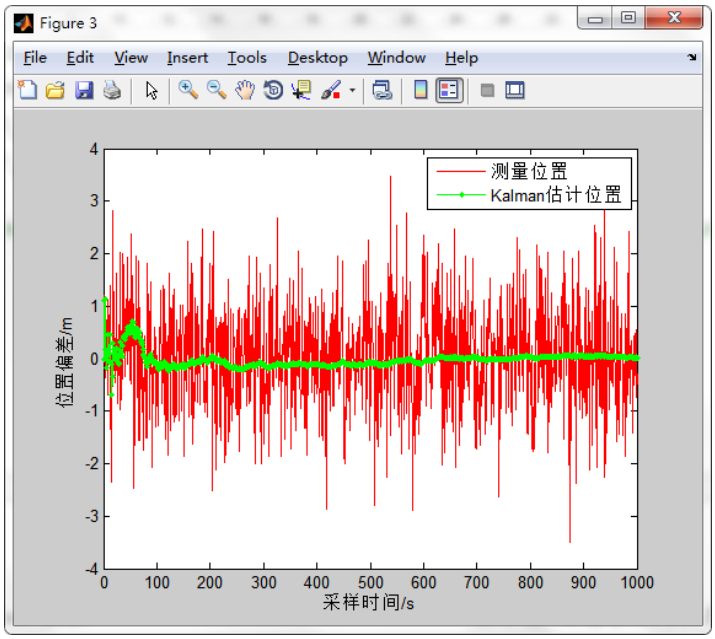
这是估计误差的协方差，系统噪声和量测噪声是独立高斯白噪声时，滤波器所得结果就是系统的最优状态估计。

按照以上算法对自由落体的物体，使用 Kalman 滤波进行目标位置预测的 matlab 程序如下：

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Kalman滤波跟踪动态目标（自由落体）
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function main
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N = 1000;
Q = [0,0;0,0];
R = 1;
W = sqrt(Q)*randn(2,N);
V = sqrt(R)*randn(1,N);
A = [1,1;0,1];
 B = [0.5;1];
U = -1;
H = [1,0];
X = zeros(2,N);
 X(:,1) = [95;1];
P0 = [10,0;0,1];
 Z = zeros(1,N);
Z(1) = H*X(:,1);
Xkf = zeros(2,N);
Xkf(:,1) = X(:,1);
err_P = zeros(N,2);
err_P(1,1) = P0(1,1);
err_P(1,2) = P0(2,2);
I = eye(2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k = 2:N
    X(:,k) = A*X(:,k-1)+B*U + W(k);
```

```
    Z(k) = H*X(:,k) + V(k);
    X_pre = A*Xkf(:,k-1) + B*U;
    P_pre = A*P0*A' + Q;
    Kg = P_pre*H'/(H*P_pre*H' + R);
    Xkf(:,k) = X_pre + Kg*(Z(k)-H*X_pre);
    P0 = (I - Kg*H)*P_pre;
    err_P(k,1) = P0(1,1);
    err_P(k,2) = P0(2,2);

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
messure_err_x = zeros(1,N);
 kalman_err_x = zeros(1,N);
 kalman_err_v = zeros(1,N);
for k = 1:N
    messure_err_x(k) = Z(k) - X(1,k);
    kalman_err_x(k) = Xkf(1,k)-X(1,k);
    kalman_err_v(k) = Xkf(2,k) - X(2,k);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```



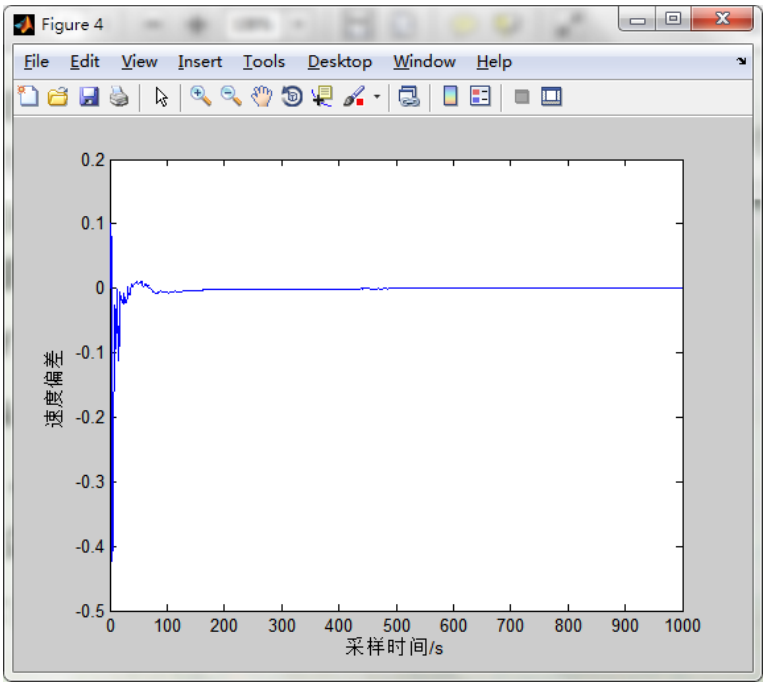


Figure 4
File Edit View Insert Tools Desktop Window Help
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
速度偏差
0 100 200 300 400 500 600 700 800 900 1000
采样时间/s