# IRLS

# 思路

1. 首先通过上一次实验的目标函数得到的线性回归方程归一化得到，得到各项系数的矩阵表达式。
2. 通过新定义的目标函数，求其关于xk的导函数，令其导函数为0得到IRLS中xk的最优估计值表达式，该最优估计值表达式中包含的各项相关的线性回归系数都已经在思路1中得到。
3. 在中，c如果取得无穷大，那么该IRLS就变为了普通的最小二乘解，因为ρ函数可以看成第一行的函数，其目标函数又变为了最原先的最小二乘函数的目标函数。只不过此时的情况不能很好地优化存在大量outlier点的情况。
4. 如果将c的值设置为合理的值（意味着一旦碰到outlier，ri就会变得很大，因为ri=yi-ait\*x,zi变得很大，而x估计值不变，那么ri就会变得很大），那么ρ函数的第二行就可以降低此次异常对于目标函数的影响，但是关键还是取决于c的取值。
5. 由于IRLS会考虑到对异常的优化，所以难免会损失一些精度，在刻画样本方差和实际的P误差协方差矩阵的时候，随着时间的增长，误差方差会变得越来越大。
6. 从结果出可以看到，尽管该实验模拟的是只观测物体位移，且不存在空气阻力的情况。在普通的卡尔曼滤波中加入异常Zk，可以看到速度的样本方差不会受到影响。但是在IRLS中，速度的样本方差却受到了影响。这说明在更新Xk估计值的公式1.8中，有影响速度的因子。但是在卡尔曼的5个公式中，Xk的更新公式中没有影响速度的因子。

# 公式



**公式1.1 线性回归形式**



**公式1.2**

其中，,,，该新型回归的误差协方差矩阵，将等式两边同时左乘Lk^(-1)即可归一化得公式1.3

****

**公式1.3 由公式1.2左乘Lk^(-1)得到的新的线性回归**

将上述的线性回归写成公式1.4的形式

****

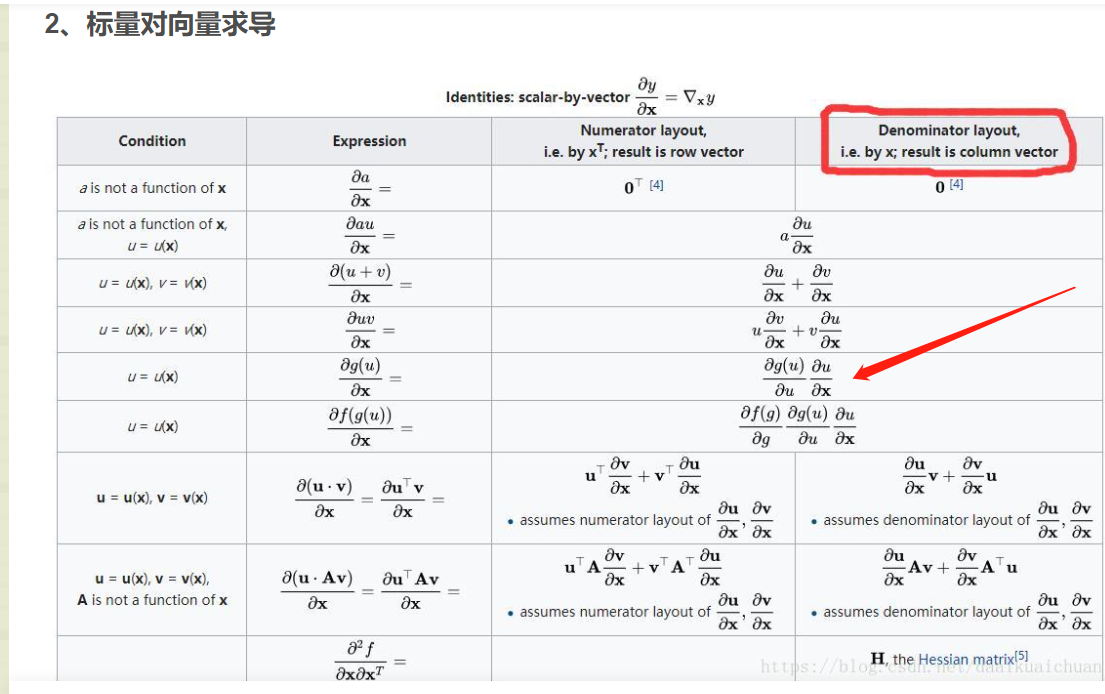
**公式1.4 新的线性回归形式**

**其中，， ，**

新的最小二乘目标函数如公式1.5所示



**公式1.5 新的目标函数**

在公式1.5中，我们默认ri=wi=1，接下来，要求Xk的估计值，就要将J(x)对x求导，令导函数为0。由于ri是关于x的标量，故应当用这个求导公式。最后的是一个标量，所以ai写在前或者后不重要。



公式1.6 最终的求导形式

令其为0，通过公式1.6最终可以写成公式1.7



公式1.7 导函数为0的矩阵形式

其中，,,以及ρ函数



通过公式1.7和论文中提及的[31][42]两篇论文，可以得到公式1.8



公式1.8 IRLS的x估计值

若做100时刻的采样，每个时刻重复做n次，在每个时刻的每次实验还需要重复迭代k次才能得到该时刻的最优估计值。k的取值取决于阈值设定的大小，只有当第v+1次的迭代得到的估计值减去第v次的估计值的差小于该阈值的时候才停止迭代。每次迭代由于改变了ri残差，所以导致Q矩阵也会发生改变（因为Q矩阵是关于ri的）。

The class of M-estimators is robust but only effective against the observation and innovation outliers’ influence in residual.该估计器只在outlier存在的情况下有效，事实上，人为改动某个时刻观测值Z，可以发现，IRLS的拟合效果比OLS的效果要更好(在c正常的情况下，不为无穷大）。

# 源代码

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%function main3\_3\_2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

N=100; %仿真时间，时间序列总数

vector=zeros(100,100); % 位移的误差

vector1=zeros(100,100); % 速度的误差

c=0.35;

for kk=1:100

% 定义一个行向量存储每次实验每个时刻的方差

% 噪声

Q=[0,0;0,0]; % 过程噪声方差为0，即下落过程忽略空气阻力

R=1; % 观测噪声方差

W=sqrt(Q)\*randn(2,N);% 既然Q为0，则W=0；在此写出，方便对照理解

V=sqrt(R)\*randn(1,N);% 测量噪声V(k)

% 系统矩阵

A=[1,1;0,1]; %状态转移矩阵

B=[0.5;1]; %控制量

U=-1;

H=[1,0]; %观测矩阵

% 初始化

X=zeros(2,N); % 物体真实状态

X(:,1)=[95;1]; % 初始位移和速度

P0=[10,0;0,1]; % 初始误差

Z=zeros(1,N);

Z(1)=H\*X(:,1); % 初始观测值

Xirls=zeros(2,N);% Kalman估计状态初始化

Xirls(:,1)=X(:,1)+sqrt(P0)\*randn(2,1); % 在初始化的时候加入误差方差估计值 从而保持吻合

err\_P=zeros(N,2);

err\_P(1,1)=P0(1,1);

err\_P(1,2)=P0(2,2);

I=eye(2); % 二维系统

% 第一列

vector(kk,1)=Xirls(1,1)-X(1,1);

vector1(kk,1)=Xirls(2,1)-X(2,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for k=2:N

%物体下落，受状态方程的驱动

X(:,k)=A\*X(:,k-1)+B\*U+W(k);

% 位移传感器对目标进行观测

Z(k)=H\*X(:,k)+V(k);

if k>50 && k<60

Z(k)=100;

end

% 1 状态预测

Xirls\_pre=A\*Xirls(:,k-1)+B\*U;

P\_pre=A\*P0\*A'+Q;

Lt=[R zeros(1,2);zeros(2,1) P\_pre]; % 归一化之前的线性回归的误差协方差矩阵 Lt=Lk\*Lk'

% 归一化之后的线性回归的各项系数

Lk=Lt^(0.5);

yk=Lk^(-1)\*[Z(k);Xirls\_pre];

Ak=Lk^(-1)\*[H;eye(2)];

Kg=P\_pre\*H'\*(H\*P\_pre\*H'+R)^(-1); %计算Kalman增益

% 2 求误差ri 用先验估计值 ri是一个(3x1)向量

ri=yk-Ak\*Xirls\_pre;

% 求Q

Q1=zeros(3,3);

% 在该循环中 求出ρ函数关于ri向量的导数

for k1=1:3

ri\_row=ri(k1); % ri的第k行的值

if ri\_row==0 % 如果ri\_row=0 将其设置为很小的一个数

ri\_row=0.001;

end

roul\_der=roul\_derive(ri\_row,c); % 求导

Q1(k1,k1)=roul\_der\*(1/ri\_row); % 设置Q1对角矩阵对角线上的值

end

% 求出估计值

Xirls\_after=(Ak'\*Q1\*Ak)^(-1)\*Ak'\*Q1\*yk;

% 因为有两个状态变量（位移和速度），分别设置两个阈值，只有当两个状态变量k时刻和k-1时刻的值的差值大于阈值，不断迭代

while Xirls\_after(1)-Xirls\_pre(1)>1 && Xirls\_after(2)-Xirls\_pre(2)>0.1

Xirls\_pre=Xirls\_after; % save

% 与进入while循环之前相同

ri=yk-Ak\*Xirls\_after;

for k1=1:3

ri\_row=ri(k1);

if ri\_row==0

ri\_row=0.001;

end

roul\_der=roul\_derive(ri\_row,c);

Q1(k1,k1)=roul\_der\*(1/ri\_row);

end

Xirls\_after=(Ak'\*Q1\*Ak)^(-1)\*Ak'\*Q1\*yk;

end

P0=(I-Kg\*H)\*P\_pre\*(I-Kg\*H)'+Kg\*R\*Kg';%方差更新

Xirls(:,k)=Xirls\_after; % 将该时刻的最后一次迭代值存入Xirls矩阵

vector(kk,k)=Xirls(1,k)-X(1,k); % 第kk次实验的第k个时刻 位移的误差

vector1(kk,k)=Xirls(2,k)-X(2,k); % 第kk次实验的第k个时刻 速度的误差

% 误差均方值

err\_P(k,1)=P0(1,1);

err\_P(k,2)=P0(2,2);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end

x\_hat(:,1)=Xirls(:,1);

P(:,:,1)=[10,0;0,1];

for k=2:N

xp=A\*x\_hat(:,k-1)+B\*U;

Pp=A\*P(:,:,k-1)\*A'+Q;

K=Pp\*H'\*(H\*Pp\*H'+R)^(-1);

x\_hat(:,k)=xp+K\*(Z(k)-H\*xp);

P(:,:,k)=(eye(2)-K\*H)\*Pp;

end

% 计算vector矩阵中 每一列的方差 存储在行向量中

Difference=var(vector,0,1);

Difference1=var(vector1,0,1);

errPx=transpose(err\_P);

errPx1=errPx(1,:);

errPx2=errPx(2,:);

% 将err\_P(k,1)和Difference()画在同一张图中，比较卡尔曼滤波估计出来的Pk与实际的方差相差多少.

figure

plot(Difference,'-bo'); % 实际的位移误差方差

hold on;

plot(errPx1,'-g+');

legend('实际位移误差方差','kalman估计方差');

xlabel('采样时间/s');

ylabel('方差');

figure

plot(Difference1,'-bo'); %实际的速度误差方差

hold on;

plot(errPx2,'-g+');

legend('实际速度误差方差','kalman估计方差');

xlabel('采样时间/s');

ylabel('方差');

figure(3);

hold on;

plot(X(1,:));

plot(Xirls(1,:),'--');

plot(x\_hat(1,:),'g--');

legend('true state','IRLS estimate','KF estimate');

figure(4);

hold;

plot(X(2,:));

plot(Xirls(2,:),'--');

plot(x\_hat(2,:),'g--');

legend('true state','IRLS estimate','KF estimate');

function roul\_der = roul\_derive(ri,c)

if abs(ri)<c

roul\_der=ri;

elseif ri>0

roul\_der=c;

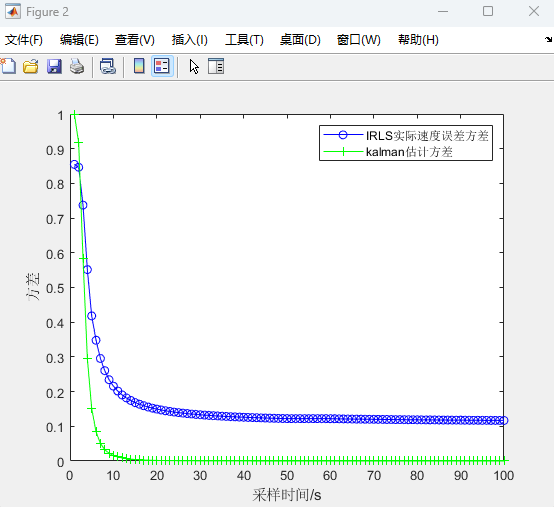
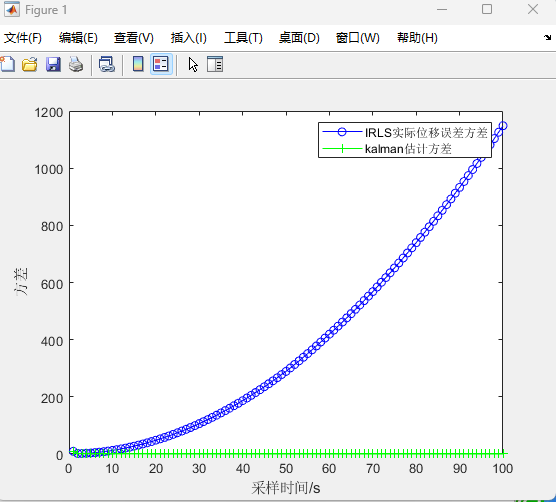
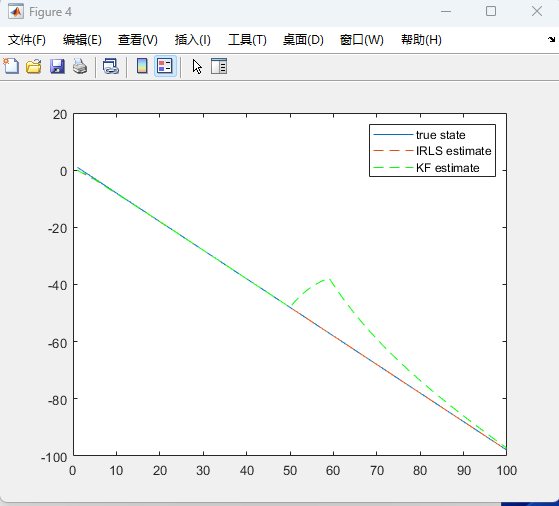
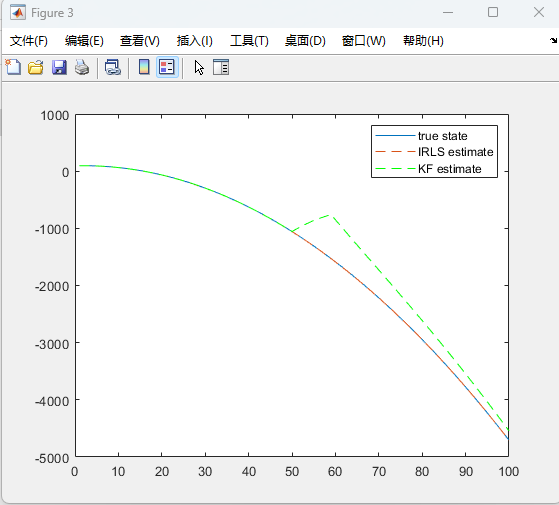
else

roul\_der=-c;

end

end

# 结果



以上是c=0.35时，且存在异常Z(k)的曲线，以下是c=1e5也存在异常Zk的曲线。

