# EKF

## 公式





公式1.1 非线性系统状态方程和观测方程

说明：u是过程控制，w是系统过程噪声，v是观测噪声





公式1.2 非线性系统状态先验和观测方程先验估计

说明：和线性kalman滤波中类似，预测估计值的时候，令噪声为0，便可得到k时刻的先验估计。需要注意的是，非线性kalman中，xk的先验估计由k-1时刻的最优估计值得来。可以类比，但是由于线性kalman中没有对观测值进行先验估计，这里的观测值先验估计由k时刻的x先验估计值得来。





公式1.3 非线性kalman公式1.1经过泰勒展开得来的近似估计

说明：第一个公式是在k-1时刻的最优估计点进行展开，第二个公式是在k时刻的先验估计点进行展开，其中A，W，H，V见公式1.4









公式1.4 公式1.3的各项系数的表达式

说明：对于A的求导，其括号中的值是求完偏导之后需要带入的值，因为噪声点的均值为0，所以带入噪声的值为0。



公式 1.5 公式1.3中误差的协方差矩阵

说明：=，=





公式1.6 EKF中的预测公式







公式1.7 EKF中的更新公式

说明：其中P的预测和Kk的更新公式中，相当于更换了噪声的协方差矩阵，对于公式1.7中的Xk估计公式，可以由公式1.8中的目标函数对Xk求导得来。



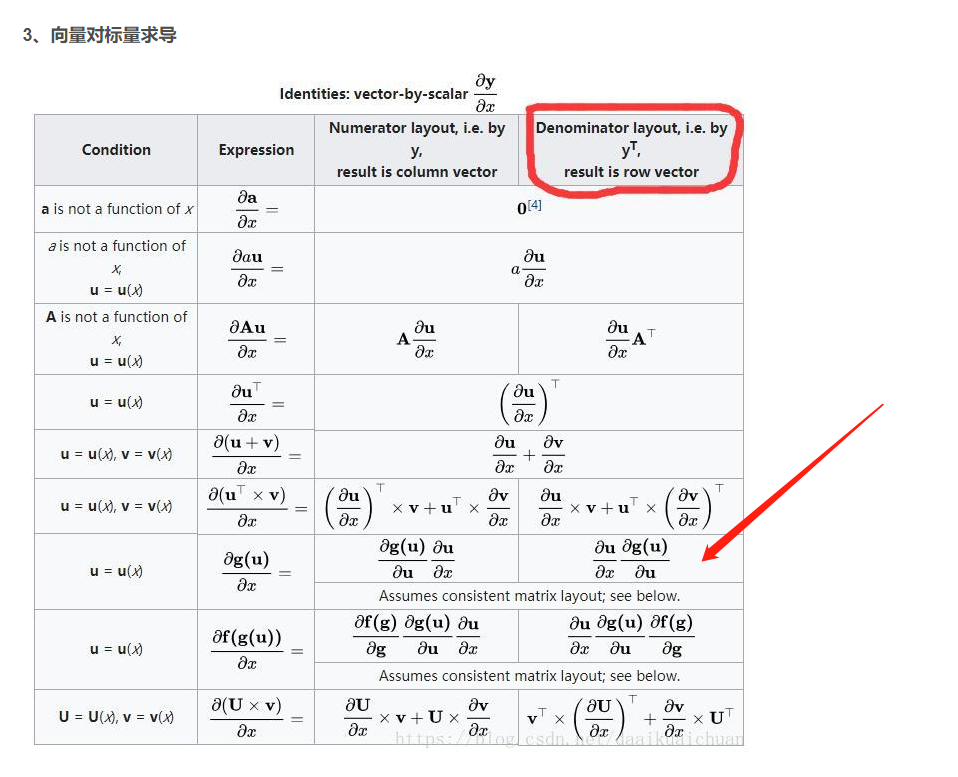
公式1.8 EKF目标函数

推导过程：

解释：对于第一项的求偏导过程如下，令，u=u(x)，u是关于x的向量，，，

对于第二项的求偏导过程如下，令，u=u(x)，u是关于x的向量，可以理解为yk是一个向量，所以xk是一个向量

，，因为如果A不是一个关于X的函数的话，有如下向量对向量求导的公式，故



令，有



令，有

该形式其实又回到了KF目标函数对xk求导的过程。过程如下。



可以看出，xk最有估计值就是由目标函数对xk求导得来，接下来的线性回归也可以证明其最小二乘估计就是最优解（就等于上式的xk\_hat）,

### 线性回归

与线性kalman类似，首先根据目标函数写出线性回归方程，只不过这里需要用泰勒展开之后的式子进行线性回归。



公式1.8 回归模型



公式1.9 线性回归模型

说明：假设噪声为加性噪声，在进行泰勒展开的时候，不需要进行求偏导V



公式1.10 线性回归方程

说明：其中,,,，注意方差不能为负，所以尽管ek\_tilta中含有-ita项，其算出来的方差也为正数

现在令,有



公式1.11 归一化之后的线性回归方程

令，有



公式1.12

则有，，，

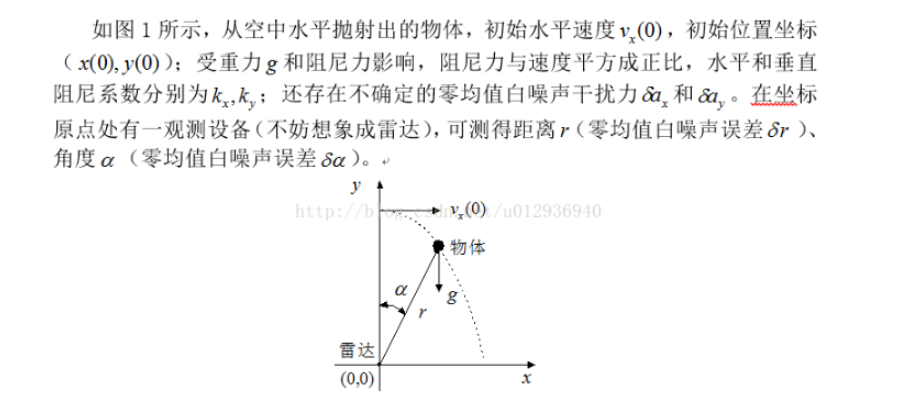
有最小二乘解为，接下来证明最小二乘解就是最优估计



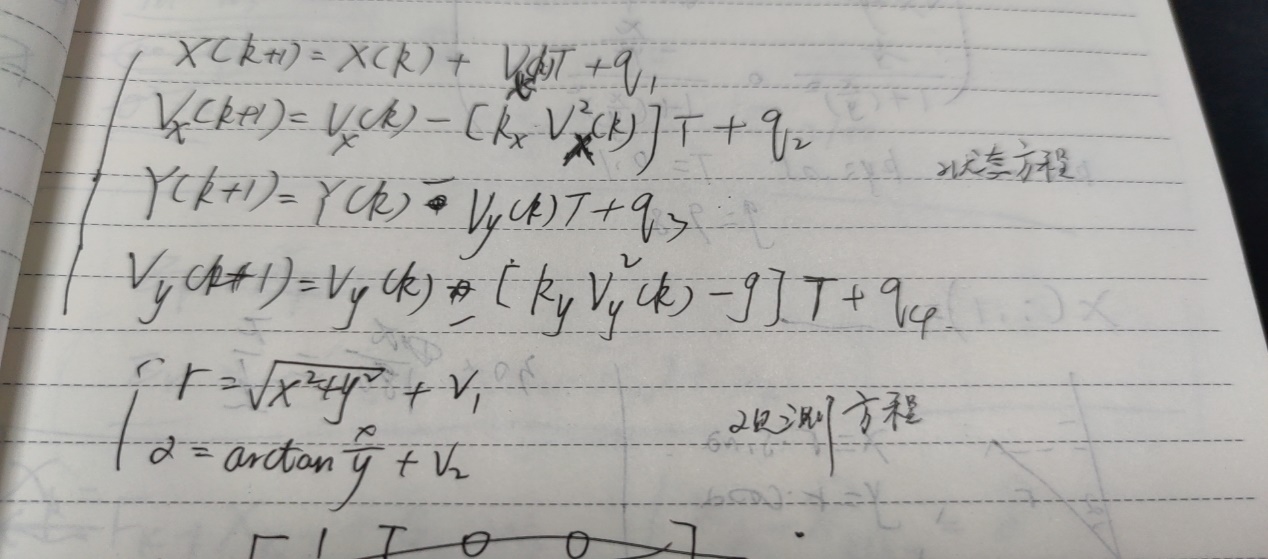
将最后一项经过matrix inverson lemma进行化简，可以看出就是误差协方差矩阵P，另外一种推导证明方法见OLS中8.12更新。

，其中，最后的化简过程就和上述的J对x求偏导一样了。

## 物理模型



下图为系统状态方程和观测方程。



泰勒展开之后的雅可比行列式



## 代码

#### 线性回归的EKF

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

%function main3\_3\_2

% non-linear model， OLS, No IRLS

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

N=100; % simulation time

T=0.1;

kx=0.01;ky=0.05;g=9.8;

for i=1:4

vector\_matrix(:,:,i)=zeros(100,N);

end

% do kk time experiments

for kk=1:100

delta\_w=1e-3;

Q=diag([0.5,1,0.5,1]); % system process noise covariance matrix

R=diag([1,1]); % observation noise variance matrix, observation states, distance, angle

% initialization

X=zeros(4,N);

% system noise and observation noise

W=sqrt(Q)\*randn(4,N);

V=sqrt(R)\*randn(2,N);

X(:,1)=[0,50,500,0]; % initial x position=0,velocity x,y position, velocity y;

P0=eye(4);

Z=zeros(2,N);

for t=2:N

x1=X(1,t-1)+X(2,t-1)\*T+W(1,t);

v1=X(2,t-1)-(kx\*X(2,t-1)^(2))\*T+W(2,t);

y1=X(3,t-1)-X(4,t-1)\*T+W(3,t);

v2=X(4,t-1)-(ky\*X(4,t-1)^(2)-g)\*T+W(4,t);

X(:,t)=[x1;v1;y1;v2];

end

% init observation

for t=1:N

x1=X(1,t);y1=X(3,t);

r1=Dist(x1,y1)+V(1,t);

alpha1=atan(x1/y1)\*180/pi+V(2,t);

Z(:,t)=[r1;alpha1];

end

X\_ekf=zeros(4,N);% init estimation

X\_ekf(:,1)=X(:,1)+sqrt(P0)\*randn(4,1); % Introducing an estimated error variance during initialization to maintain alignment

err\_P=zeros(4,N); % Each column represents the four values of the Kalman's P matrix at that moment, with each row recording each timestamp.

for k2=1:4

err\_P(k2,1)=P0(k2,k2);

end

I=eye(4); % 4 dimensions systems

% real error estimation

for j=1:4

vector=vector\_matrix(:,:,j);

vector(kk,1)=X\_ekf(j,1)-X(j,1);

vector\_matrix(:,:,j)=vector;

end

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

for k=2:N

% state prediction, apply the state prediction to the variable Xekf)

% xpre\_hat{k}=f(,,0)

x1=X\_ekf(1,k-1)+X\_ekf(2,k-1)\*T;

v1=X\_ekf(2,k-1)-kx\*X\_ekf(2,k-1)^(2)\*T;

y1=X\_ekf(3,k-1)-X\_ekf(4,k-1)\*T;

v2=X\_ekf(4,k-1)-(ky\*X\_ekf(4,k-1)^(2)-g)\*T;

Xekf\_pre=[x1;v1;y1;v2];

%Xekf\_pre=F\*X\_ekf(:,k-1);

% observation prediction

r=Dist(x1,y1);alpha2=atan(x1/y1)\*180/pi;

Zekf\_pre=[r;alpha2];

% introducing A(k) matrix: partial derivative

A=[1 T 0 0;0 1-2\*kx\*X\_ekf(2,k-1)\*T 0 0;0 0 1 -T;0 0 0 1-2\*ky\*X\_ekf(4,k-1)\*T];

dd=Dist(x1,y1); de=1+(x1/y1)^(2);

H=[x1/dd 0 y1/dd 0;(1/y1)/de 0 (-x1/y1^(2))/de 0]; % jacobi matrix

% pre of the P

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

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

% linearization

H\_tilta=[I;H];

Y\_tilta=[Xekf\_pre;Z(:,k)-Zekf\_pre+H\*Xekf\_pre];

R\_tilta=[P\_pre zeros(4,2);zeros(2,4) R];

%normalization

St=R\_tilta^(1/2);

H\_tilta1=St^(-1)\*H\_tilta;

Y\_tilta1=St^(-1)\*Y\_tilta;

X\_ekf(:,k)=(H\_tilta1'\*H\_tilta1)^(-1)\*H\_tilta1'\*Y\_tilta1;

P0=(I-K\*H)\*P\_pre;

% update error

for j=1:4

vector=vector\_matrix(:,:,j);

vector(kk,k)=X\_ekf(j,k)-X(j,k);

vector\_matrix(:,:,j)=vector;

end

for k2=1:4

err\_P(k2,k)=P0(k2,k2);

end

end

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

end

% Comparing classical EKF and EKF\_OLS, store the estimation of classical

% EKF into the variable X\_classical

P0=eye(4);

X\_classical=zeros(4,N);

X\_classical(:,1)=X(:,1)+sqrt(P0)\*randn(4,1);

Q=diag([0.5,1,0.5,1]); % system process noise covariance matrix

R=diag([1,1]); % observation noise variance matrix, observation states, distance, angle

% system noise and observation noise

W=sqrt(Q)\*randn(4,N);

V=sqrt(R)\*randn(2,N);

for k=2:N

x1=X\_classical(1,k-1)+X\_classical(2,k-1)\*T;

v1=X\_classical(2,k-1)-kx\*X\_classical(2,k-1)^(2)\*T;

y1=X\_classical(3,k-1)-X\_classical(4,k-1)\*T;

v2=X\_classical(4,k-1)-(ky\*X\_classical(4,k-1)^(2)-g)\*T;

Xclassical\_pre=[x1;v1;y1;v2];

r=Dist(x1,y1);alpha2=atan(x1/y1)\*180/pi;

Zekf\_pre=[r;alpha2];

% introducing A(k) matrix: partial derivative

A=[1 T 0 0;0 1-2\*kx\*X\_classical(2,k-1)\*T 0 0;0 0 1 -T;0 0 0 1-2\*ky\*X\_classical(4,k-1)\*T];

dd=Dist(x1,y1); de=1+(x1/y1)^(2);

H=[x1/dd 0 y1/dd 0;(1/y1)/de 0 (-x1/y1^(2))/de 0]; % jacobi matrix

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

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

X\_classical(:,k)=Xclassical\_pre+K\*(Z(:,k)-Zekf\_pre);

P0=(I-K\*H)\*P\_pre;

end

% calculate every column of the vector matrix,

% store the value in the row vector

for i=1:4

vector=vector\_matrix(:,:,i);

Difference\_matrix(:,:,i)=var(vector,0,1);

end

for i=1:4

errPx\_matrix(:,:,i)=err\_P(i,:);

end

figure(1)

hold on;box on;

plot(X(1,:),X(3,:),'-k.');

plot(X\_ekf(1,:),X\_ekf(3,:),'-r+');

plot(Z(1,:).\*sin(Z(2,:)\*pi/180),Z(1,:).\*cos(Z(2,:)\*pi/180),'+');

legend('real orbit','EKF orbit','Observation orbit');

figure(2)

plot(Difference\_matrix(:,:,1),'-bo');

hold on;

plot(errPx\_matrix(:,:,1),'-g+');

legend('real X displacement error variance','ekf estimation error variance');

xlabel('sampling time/s');

ylabel('variance');

figure(3)

plot(Difference\_matrix(:,:,3),'-bo');

hold on;

plot(errPx\_matrix(:,:,3),'-g+');

legend('real Y displacement error variance','ekf estimation error variance');

xlabel('sampling time/s');

ylabel('variance');

% omit the error variance comparision of the velocity

figure(4)

plot(X\_ekf(1,:),'-go')

hold on;

plot(X\_classical(1,:),'-bo')

legend('OLS EKF X displacement estimation','classical EKF X displacement estimation');

xlabel('sampling time/s');

ylabel('displacement/m');

figure(5)

plot(X\_ekf(2,:),'-go')

hold on;

plot(X\_classical(2,:),'-bo')

legend('OLS EKF X velocity estimation','classical EKF X velocity estimation');

xlabel('sampling time/s');

ylabel('velocity m/s');

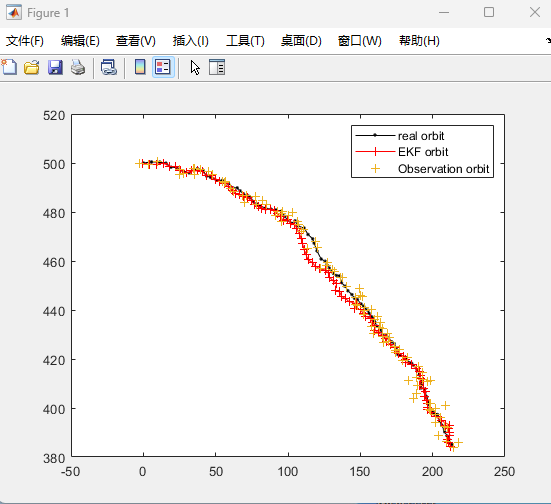
function d=Dist(X1,X2)

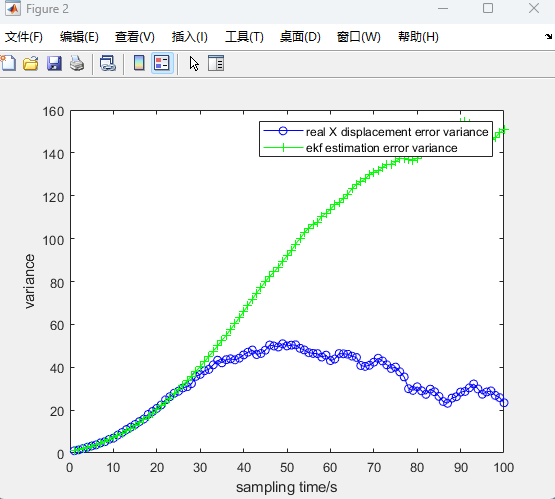
d=sqrt(X1^(2)+X2^(2));

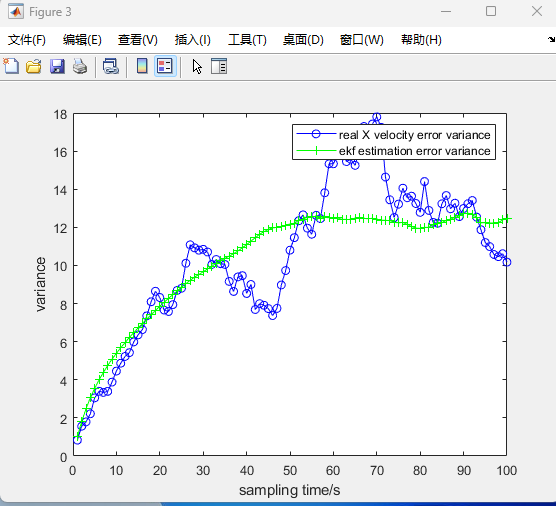
end

### 运行结果

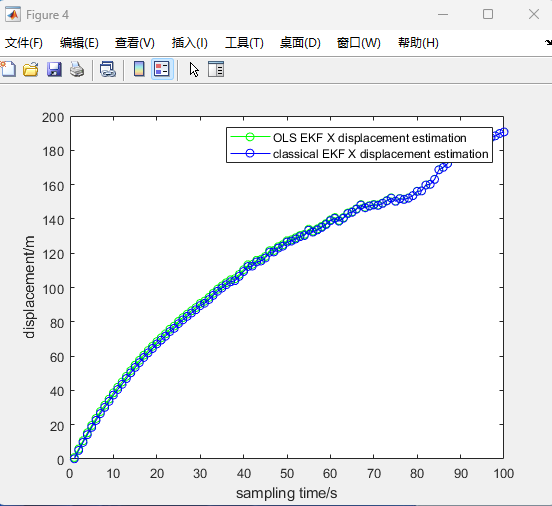
##### 线性回归的EKF运行结果







比较EKF最小二乘和最优估计



#### 