> 原文地址 blog.csdn.net

卡尔曼滤波器

> 卡尔曼滤波器是一个 “optimal recursive data processing algorithm”（最优化自回归数据处理算法）。

先说一个例子：

假设我们要研究的对象是一个房间的温度。

> - 1.根据经验，温度是恒定的，即上一分钟的温度等于现在这一分钟的温度，经验即预测，但这并不是完全可信的，即存在一定的误差。我们假设成高斯白噪声。
> - 2.另外在房间里放一个温度计，实时检测房间的温度，这是观测值。同样地也存在误差，也是高斯白噪声。

我们现在的目标就是，用这两个值，结合它们各自的噪声，估算出房间的实际温度。假设要估算 k 时刻的实际温度值，首先需要知道 $k-1$ 时刻的温度值。

1. 预测值，假设 $k-1$ 时刻房间温度为23度，则 k 时刻的温度应该也为23度（恒定）。同时该值的偏差为5（5是3的平方加上4的平方再开方，其中3为 $k-1$ 时刻估算出的最优温度值的偏差3，4为对预测的偏差）。
2. 测量值，假设为25度，同时偏差为4度。

那么实际值为多少？相信谁多一点，可用它们的协方差来判断：

$$Kg^2 = \frac{5^2}{5^2+4^2},\quad Kg = 0.78$$

则实际的温度值是：23+0.78*(25-23) = 24.56度。可以看出温度计测量的 covariance 较小，所以实际值比较偏向温度计。

在进入 $k+1$ 时刻估算前，需要算出 k 时刻最优值（24.56）的偏差。算法$\sqrt{(1-Kg)\times 5^2}=2.35$，5即 k 时刻估算所用 $k-1$ 时刻23度温度值的偏差。得出的2.35即 k 时刻最优值的偏差（对应上面 $k-1$ 时刻的3）。

这样卡尔曼滤波器就不断得把 covariance 递归，从而估算出最优值。而且它只保留上一刻的 covariance。上面的 Kg 就是卡尔曼增益（***Kalman Gain***）。

1. 真正工程系统上的卡尔曼滤波器算法

首先引入一个离散控制过程的系统，改系统可用一个线性随机微分方程（linear stochastic difference equation）来描述：

$$X(k)=A\cdot X(k-1)+B\cdot U(K)+W(k)$$

再加上系统的测量值：

$$Z(k)=H\cdot X(k)+V(k)$$

其中：$X(k)$ 是 k 时刻的系统状态，$U(k)$ 是 k 时刻对系统的控制量。A 和 B 是系统参数，对于多模系统，它们为矩阵。

$Z(k)$ 是 k 时刻的测量值，H 是测量参数，对于多模系统，H 为矩阵。$W(k)$ 和 $V(k)$ 分别表示过程和测量的噪声，它们被假设成高斯白噪声，它们的 covariance 分别是 Q，R。

对于满足上面条件（线性随机微分系统，过程和测量都是高斯白噪声），卡尔曼滤波器是最优的信息处理器。

2. 卡尔曼滤波器算法流程及核心公式

首先利用系统过程模型，来预测下一个状态的系统：

$$X(k|k-1)=A\cdot X(k-1|k-1)+B\cdot U(k) \tag{1}$$

$X(k|k-1)$ 是利用上一状态预测的结果，$X(k-1|k-1)$ 是上一状态最优结果。$U(k)$ 为现在状态的控制量，若没有，可为0.

现在更新 $X(k|k-1)$ 的 covariance，用 P 表示：

$$P(k|k-1)=A\cdot P(k-1|k-1)\cdot A'+Q \tag{2}$$

其中 $P(k|k-1)$ 是 $X(k|k-1)$ 对应的 covariance；$P(k-1|k-1)$ 是 $X(k-1|k-1)$ 对应的 covariance。A' 表示A 的转置矩阵，Q 是系统的 covariance。
式(1)(2)是5各公式的前两个，用来对系统进行预测。

然后再收集系统的观测值 $Z(k)$，则最优值：

$$X(k|k) = X(k|k-1) + Kg(k) \cdot (Z(k) - H \cdot X(k|k-1)) \tag{3}$$

其中 Kg 为卡尔曼增益（***Kalman Gain***）：

$$Kg(k) = \frac{P(k|k-1) \cdot H'}{H \cdot P(k|k-1) \cdot H' + R} \tag{4}$$

到目前为止，已经得到 k 状态下最优的估算值 $X(k|k-1)$。

但是为了要卡尔曼滤波器不断得运行下去直到系统过程结束，需要更新k状态下 $X(k|k-1)$ 的 covariance：

$$P(k|k) = (I - Kg(k) \cdot H)P(k|k-1) \tag{5}$$

其中 I 为1的矩阵，对于单模系统 $I = 1$。

当系统进入 $k+1$ 状态时，$P(k|k)$ 就是式(2)中的 $P(k-1|k-1)$。这样，算法就可以自回归地运算下去。

3. 扩展卡尔曼滤波器

由上述的介绍可以得知，卡尔曼滤波器只适用于线性系统模型，然而实际中的系统往往都是非线性模型。所示必须对卡尔曼滤波器进行修改。

首先要了解一下**线性化卡尔曼滤波**它和**线性卡尔曼滤波器**在滤波器的算法方面有同样的结构，不一样的地方在于这两者的**系统模型不同**。

线性卡尔曼滤波器的系统本身就是线性系统，而线性化卡尔曼滤波器的系统本身是非线性系统，但是机智的大神们将非线性的系统进行了线性化，于是卡尔曼滤波就可以用在非线性系统中了。对于一个卡尔曼滤波器的设计者，就可以不管模型到底是一开始就是线性系统还是非线性系统线性化得到的线性系统，反正只要是线性系统就好了。

但是**线性化卡尔曼滤波器会发散**。为什么会发散呢？是这样，我们在对非线性系统进行线性化的过程中，只有被线性化的那个点附近的线性化模型和真实的模型相近，远的误差就大了，那么这个时候卡尔曼滤波器的效果就不好。所以线性化的这个限制要时刻考虑，这也就是为什么要把线性卡尔曼滤波器和

线性化卡尔曼滤波器区分开的理由。而**决定一个线性化滤波器成功与否的关键就在于这个滤波器系统模型线性化得好不好**。

EKF基于线性化系统的思想，将系统函数的非线性函数作一阶Taylor展开，得到线性化系统方程。扩展的卡尔曼滤波器算法就是适用于非线性系统的卡尔曼滤波器。它与经典的线性卡尔曼滤波器很相似，算法步骤和结构都相同。不同在于系统模型和矩阵A和H。

4. MATLAB实现卡尔曼滤波器（房间温度例子）

```
% Kalman filter example of temperature measurement in Matlab implementation of Kalman f
% 房间当前温度真实值为24度，认为下一时刻与当前时刻温度相同，误差为0.02度（即认为连续的两个时刻最多变化0
% 温度计的测量误差为0.5度。
% 开始时，房间温度的估计为23.5度，误差为1度。
close all;

% intial parameters
% 计算连续n_iter个时刻
n_iter = 100;
% size of array. n_iter行，1列
sz = [n_iter, 1];
% 温度的真实值
x = 24;
% 过程方差，反应连续两个时刻温度方差。更改查看效果
Q = 4e-4;
% 测量方差，反应温度计的测量精度。更改查看效果
R = 0.25;
% z是温度计的测量结果，在真实值的基础上加上了方差为0.25的高斯噪声。
z = x + sqrt(R)*randn(sz);
% 对数组进行初始化
% 对温度的后验估计。即在k时刻，结合温度计当前测量值与k-1时刻先验估计，得到的最终估计值
xhat = zeros(sz);
% 后验估计的方差
P = zeros(sz);
% 温度的先验估计。即在k-1时刻，对k时刻温度做出的估计
xhatminus = zeros(sz);
% 先验估计的方差
Pminus = zeros(sz);
% 卡尔曼增益，反应了温度计测量结果与过程模型（即当前时刻与下一时刻温度相同这一模型）的可信程度
K = zeros(sz);
% intial guesses
xhat(1) = 23.5; %温度初始估计值为23.5度
P(1) =1; % 误差方差为1

for k = 2:n_iter
    % 时间更新（预测）
    % 用上一时刻的最优估计值来作为对当前时刻的温度的预测
    xhatminus(k) = xhat(k-1);
    % 预测的方差为上一时刻温度最优估计值的方差与过程方差之和
    Pminus(k) = P(k-1)+Q;
    % 测量更新（校正）
    % 计算卡尔曼增益
    K(k) = Pminus(k)/( Pminus(k)+R );
    % 结合当前时刻温度计的测量值，对上一时刻的预测进行校正，得到校正后的最优估计。该估计具有最小均方差
    xhat(k) = xhatminus(k)+K(k)*(z(k)-xhatminus(k));
    % 计算最终估计值的方差
    P(k) = (1-K(k))*Pminus(k);
end

FontSize = 14;
LineWidth = 3;
```

```
figure();
plot(z,'k+'); %画出温度计的测量值
hold on;
plot(xhat,'b-','LineWidth',LineWidth) %画出最优估计值
hold on;
plot(x*ones(sz),'g-','LineWidth',LineWidth); %画出真实值
legend('温度计的测量结果', '后验估计', '真实值');
xl = xlabel('时间(分钟)');
yl = ylabel('温度');
set(xl,'fontsize',FontSize);
set(yl,'fontsize',FontSize);
hold off;
set(gca,'FontSize',FontSize);

figure();
valid_iter = 2:n_iter; % Pminus not valid at step 1
% 画出最优估计值的方差
plot(valid_iter,P(valid_iter),'LineWidth',LineWidth);
legend('后验估计的误差估计');
xl = xlabel('时间(分钟)');
yl = ylabel('℃^2');
set(xl,'fontsize',FontSize);
set(yl,'fontsize',FontSize);
set(gca,'FontSize',FontSize);
```

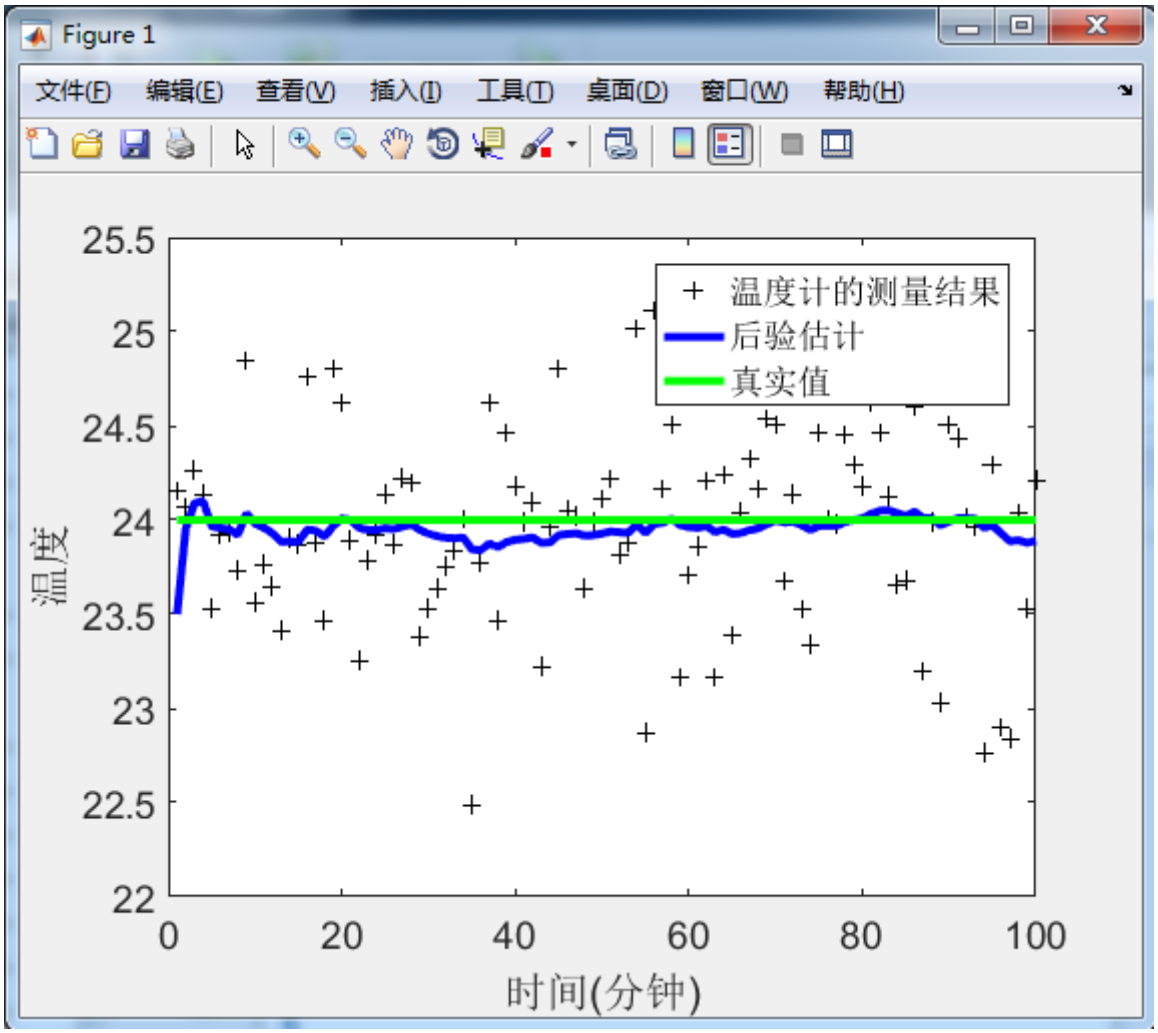
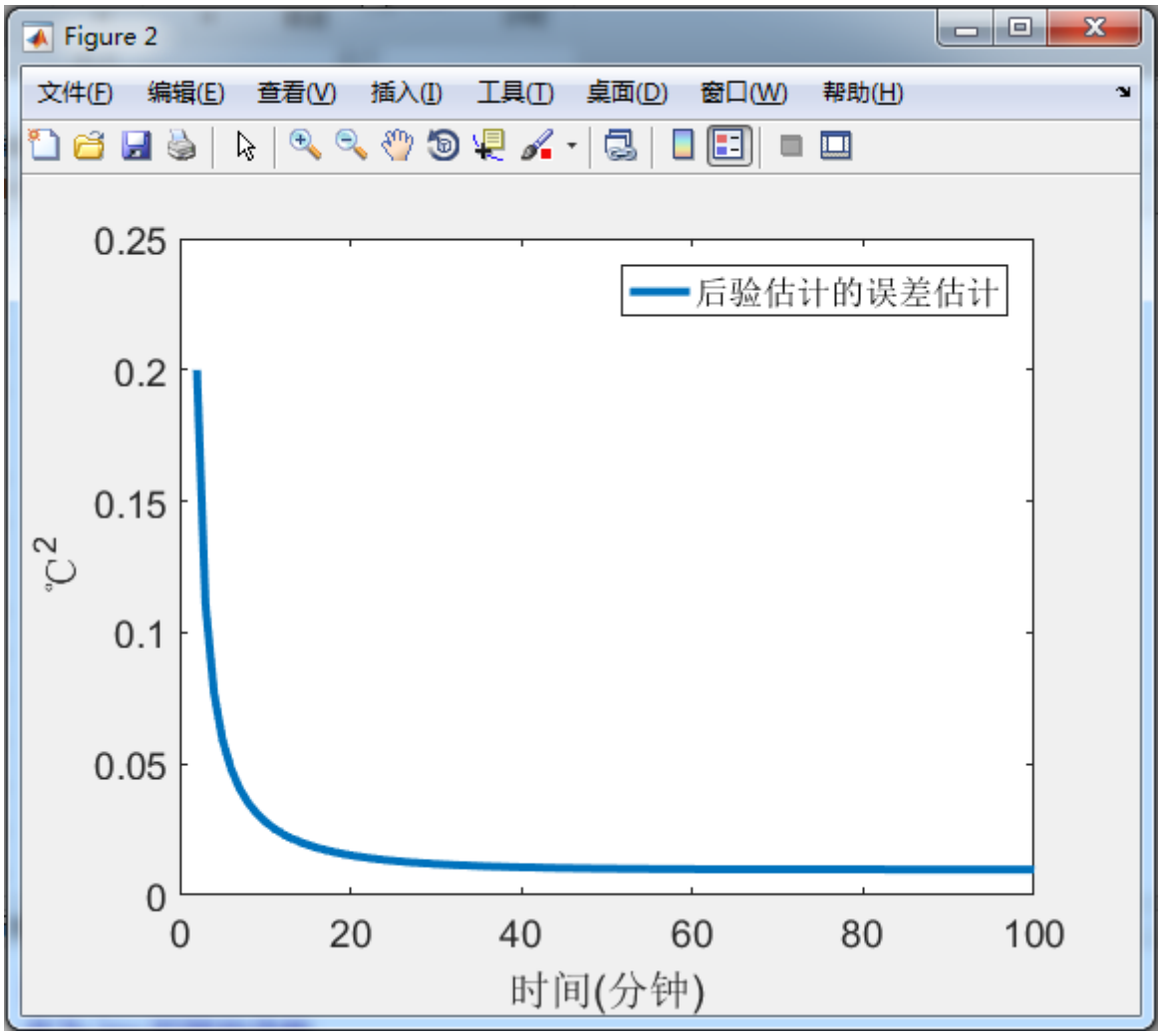


Figure 1
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
温度计的测量结果
后验估计
真实值
温度
时间(分钟)
25.5
25
24.5
24
23.5
23
22.5
22
0
20
40
60
80
100

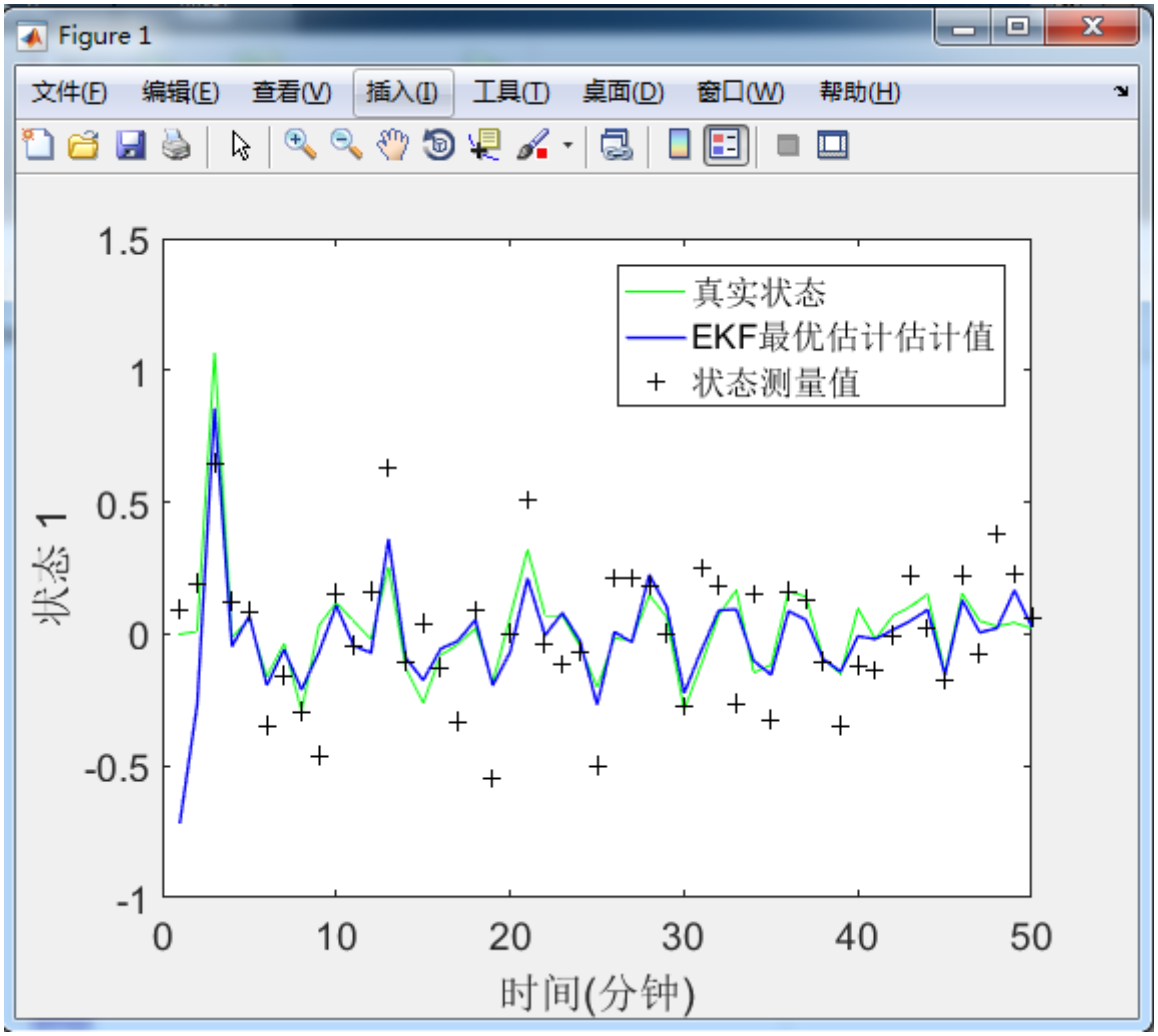


5. MATLAB实现扩展卡尔曼滤波器

```
% function: simulating the process of EKF
N = 50;            % 计算连续N个时刻
n = 3;              % 状态维度
q = 0.1;           % 过程标准差
r = 0.2;            % 测量标准差
% eye函数产生单位矩阵
Q = q^2*eye(n);    % 过程方差
R = r^2;                % 测量值的方差

%{
     FUNHANDLE = @FUNCTION_NAME returns a handle to the named function,
     FUNCTION_NAME. A function handle is a MATLAB value that provides a
     means of calling a function indirectly.
%}
f = @(x)[x(2);x(3);0.05*x(1)*(x(2)+x(3))];  % 状态方程
h = @(x)[x(1);x(2);x(3)];                              % 测量方程
s = [0;0;1];                                                  % 初始状态

% 初始化状态
x = s+q*randn(3,1);
% eye返回单位矩阵
P = eye(n);
% 3行50列，一列代表一个数据
% 最优估计值
xV = zeros(n,N);
% 真实值
sV = zeros(n,N);
% 状态测量值
zV = zeros(n,N);

for k = 1:N
    z = h(s) + r * randn;
    % 实际状态
    sV(:,k) = s;
    % 状态测量值
    zV(:,k) = z;

    % 计算f的雅可比矩阵，其中x1对应黄金公式line2
    [x1,A] = jaccsd(f,x);
    % 过程方差预测，对应line3
    P = A*P*A'+Q;
    % 计算h的雅可比矩阵
    [z1,H] = jaccsd(h,x1);

    % 卡尔曼增益，对应line4
    % inv返回逆矩阵
    K = P*H'*inv(H*P*H'+R);
    % 状态EKF估计值，对应line5
    x = x1+K*(z-z1);
    % EKF方差，对应line6
    P = P-K*H*P;
```

```
    % save
    xV(:,k) = x;
    % update process
    s = f(s) + q*randn(3,1);
end

for k = 1:3
    FontSize = 14;
    LineWidth = 1;

    figure();
    % 画出真实值
    plot(sV(k,:),'g-');
    hold on;

    % 画出最优估计值
    plot(xV(k,:),'b-','LineWidth',LineWidth);
    hold on;

    % 画出状态测量值
    plot(zV(k,:),'k+');
    hold on;

    legend('真实状态', 'EKF最优估计估计值','状态测量值');
    xl = xlabel('时间(分钟)');
    % 把数值转换成字符串， 转换后可以使用fprintf或disp函数进行输出。
    t = ['状态 ',num2str(k)] ;
    yl = ylabel(t);
    set(xl,'fontsize',FontSize);
    set(yl,'fontsize',FontSize);
    hold off;
    set(gca,'FontSize',FontSize);
end

function [z, A] = jaccsd(fun, x)
% JACCSD Jacobian through complex step differentiation
% [z J] = jaccsd(f,x)
% z = f(x)
% J = f'(x)
%
z = fun(x);
% numel返回数组中元素个数。若是一幅图像，则numel(A)将给出它的像素数。
n = numel(x);
m = numel(z);
A = zeros(m,n);
h = n*eps;
for k = 1:n
    x1 = x;
    x1(k) = x1(k)+h*1i;
    % imag返回复数的虚部
```

```
    A(:,k) = imag(fun(x1))/h;
end
```



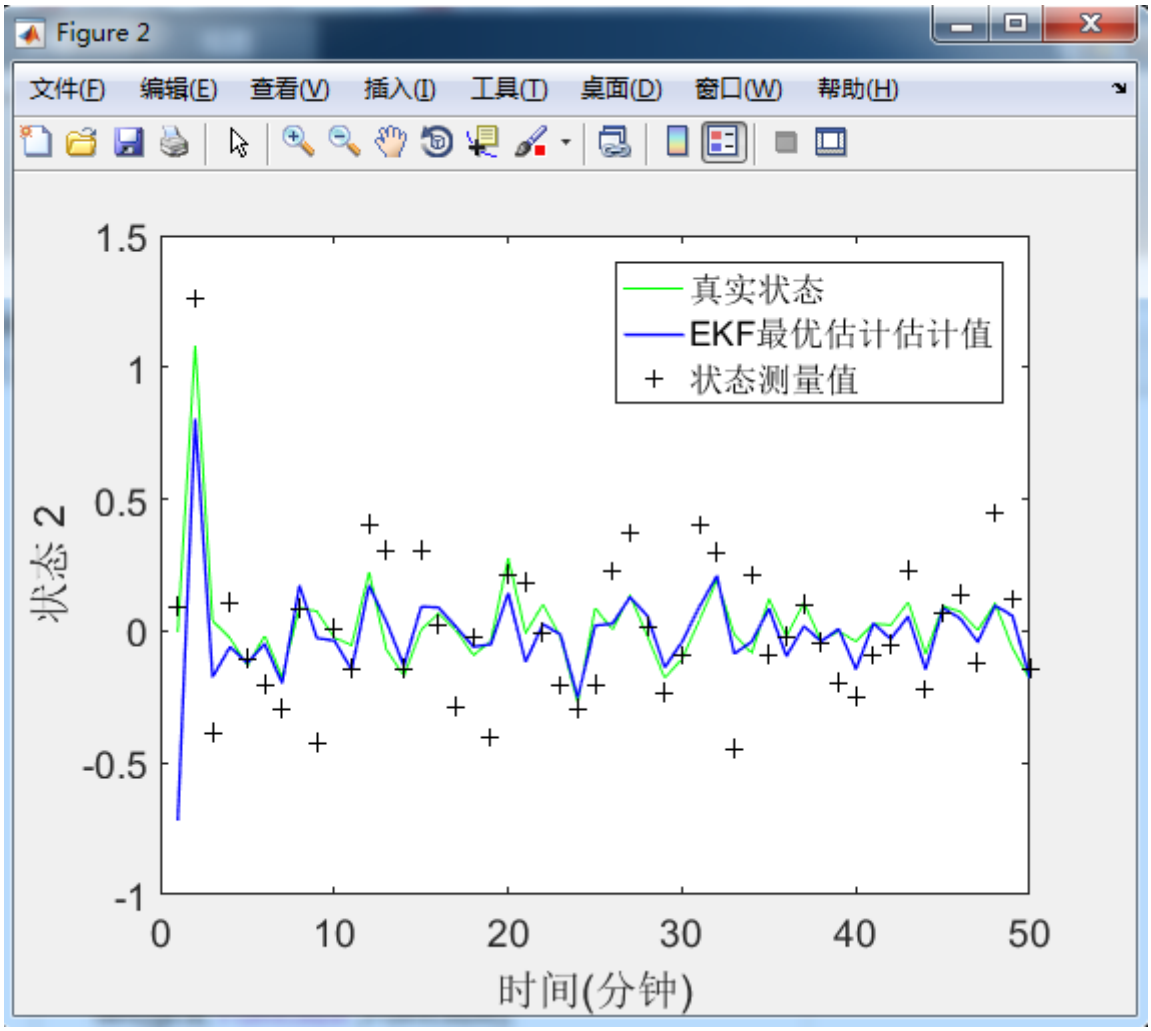


Figure 2
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
真实状态
EKF最优估计估计值
状态测量值
状态 2
时间(分钟)
1.5
1
0.5
0
-0.5
-1
0
10
20
30
40
50

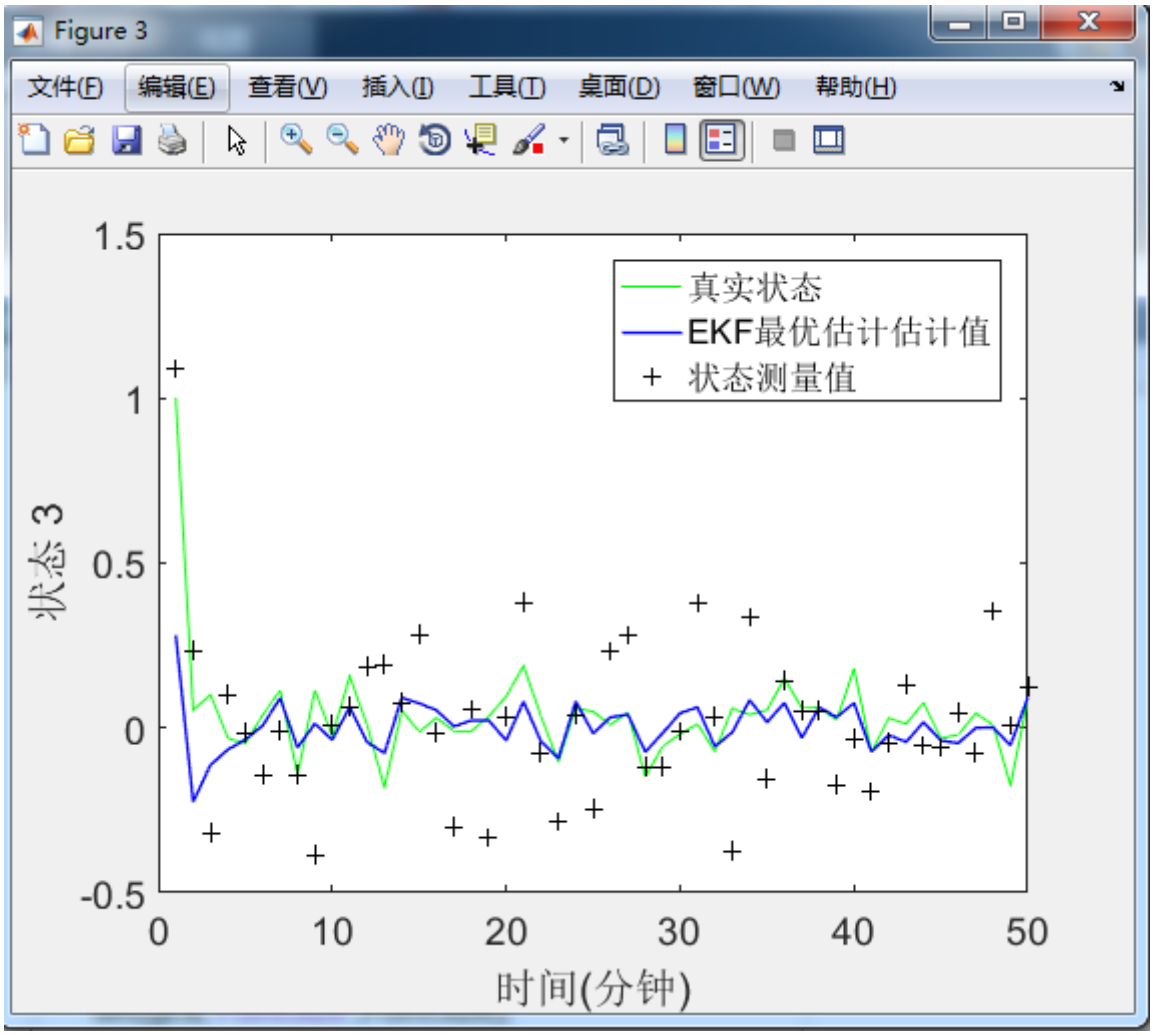


Figure 3
文件(F)
编辑(E)
查看(V)
插入(I)
工具(T)
桌面(D)
窗口(W)
帮助(H)
真实状态
EKF最优估计估计值
状态测量值
状态 3
时间(分钟)
1.5
1
0.5
0
-0.5
0
10
20
30
40
50