一、引入

上次讲到了贝叶斯滤波，这次来讲一讲卡尔曼滤波。

贝叶斯滤波算法实际上是从一维的角度阐述“预测+更新”的基本思路，但在实际应用中，描述物体状态的量可能有多个，除了位置外，还有速度、加速度……将贝叶斯滤波算法公式中的变量 x 换成矩阵 X 的形式表示，这样我们便得到了卡尔曼滤波。

先看一个笑话：

一片绿油油的草地上有一条曲折的小径，通向一棵大树，一个要求被提出：从起点沿着小径走到树下。

“很简单。” A 说，于是他丝毫不差地沿着小径走到了树下。

现在，难度被增加了：蒙上眼。

“也不难，我当过特种兵。” B 说，于是他歪歪扭扭地走到了树旁。“唉，好久不练，生疏了。” （只凭自己的预测能力）

“看我的，我有 DIY 的 GPS！” C 说，于是他像个醉汉似地歪歪扭扭的走到了树旁。“唉，这个 GPS 没做好，漂移太大。”（测量存在噪声）

“我来试试。” 旁边一也当过特种兵的拿过 GPS，蒙上眼，居然沿着小径很顺滑的走到了树下。（自己能预测+测量结果的反馈）

“这么厉害！你是什么人?” “卡尔曼 ！”

“卡尔曼？！你就是卡尔曼？”众人大吃一惊。

“我是说这个 GPS 卡而慢。”

这当然知只是一个小故事，但却指出了卡尔曼滤波的核心，预测+测量反馈，这一思想也将贯穿整个滤波算法的学习。

二、基础知识

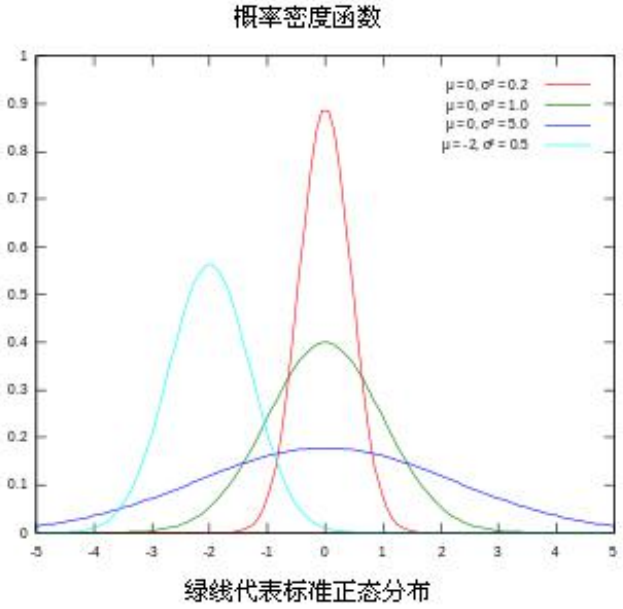
方差：方差是描述随机变量的离散程度，是变量离期望值的距离。

$$V(X)=\operatorname{cov}(X,X)=E\left((X-\mu)^2\right)$$

协方差：衡量两个变量的总体误差，表征两个变量的关系。如果两个变量的变化趋势一致，那么两个变量之间的协方差就是正值。如果两个变量的变化趋势相反，那么两个变量之间的协方差就是负值。

$$\operatorname{cov}(X,Y)=E\left((X-\mu)(Y-\nu)\right)$$

高斯分布：



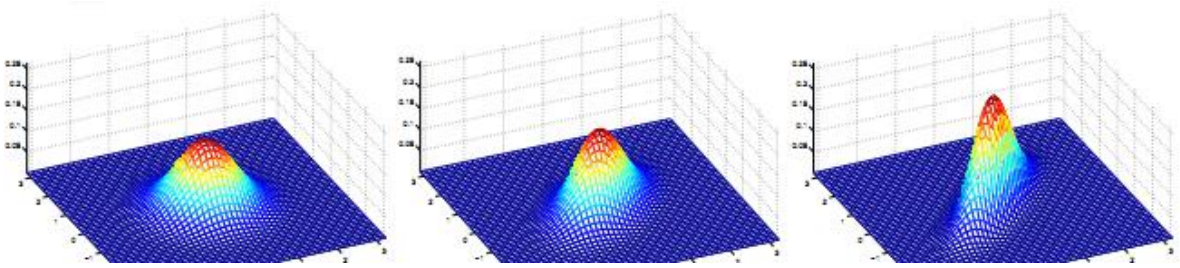
协方差矩阵：在多元高斯分布中，两变量的协方差常如下表示

$$\sum XY=\operatorname{cov}(X,Y)=E\left((X-\mu)(Y-\nu)\right)$$

则二元高斯分布的协方差矩阵可以表示成如下形式

$$\mathbf{P}_k=\begin{bmatrix}\Sigma_{pp} & \Sigma_{pv}\\ \Sigma_{vp} & \Sigma_{vv}\end{bmatrix}$$

协方差矩阵的主对角线就是方差，反对角线上的就是两个变量间的协方差。



上面几个图对应的协方差矩阵分别为：

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}; \quad .\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

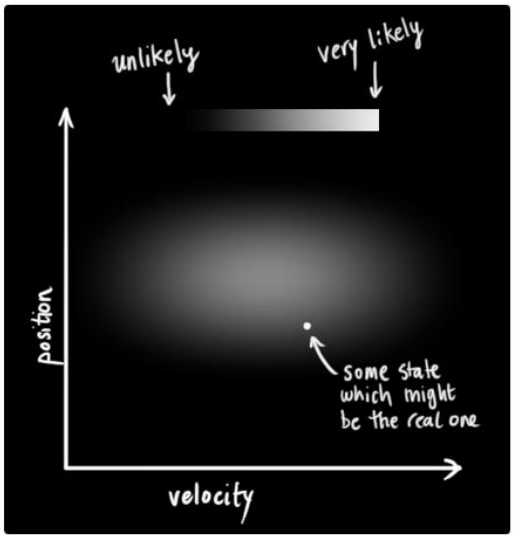
如图所示，协方差越大，图像越扁。协方差表征两个变量的线性关系，当协方差→1 时，两个变量近乎成线性关系。

三、卡尔曼滤波器

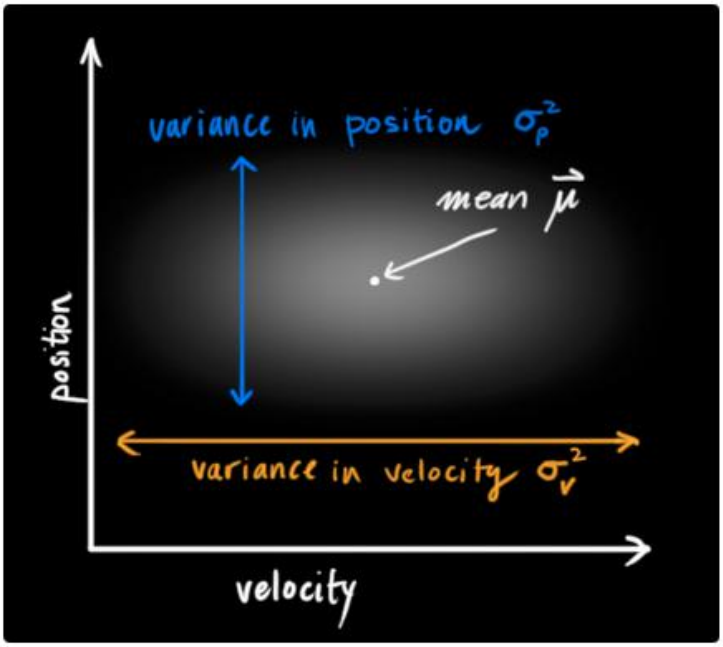
我们用位置和速度代表物体的状态：

$$\vec{x} = \begin{bmatrix} p \\ v \end{bmatrix}$$

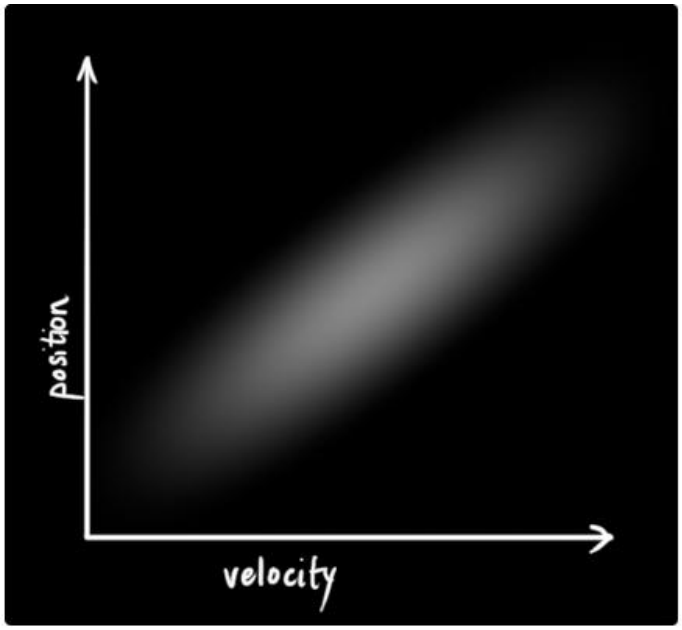
如前面所推导的一样，物体当前时刻的位置和速度值存在很多可能的可能性，我们无法知道准确的值，但是某中可能性的概率相较于其他会更大（类似电子云）。



卡尔曼滤波器假设所有变量(在此例中为位置和速度)都满足随机的高斯分布，每一个变量都有均值 u 来代表随机分布的中心点， σ^2 符号代表了不确定性。



在该示意图中，位置和速度是不具有关联性的，也就是说我们无法通过位置推断得到和与速度有关的信息，反之亦然。当位置和速度变量具有关联性时，其示意图如下所示，观察到某一个特定位置的可能性(概率)受当前速度的影响。



上图所述这种情形可能在这种情况下发生，例如，当前我们依据上一时刻的位置去预测物体下一时刻的位置，如果物体的运行速度较高，其移动距离也可能变的更大，从而导致计算的位置也发生较大的距离变化。反之亦然。

这种关系是十分重要的，因为它会告诉我们更多的有关信息：一个变量的状态将会反应出其他变量的一些状态。这也是卡尔曼滤波器的目的，即我们希望尽量的从具有不确定性的信息中获取尽可能多的有用信息。

在之后计算过程中我们用协方差矩阵来代表和描述这种关联关系。即Σij 描述了第 i 个状态变量与第 j 个状态变量之间的关联性。协方差矩阵通常用符号Σ表示，其中的元素表示为Σij。

四、用矩阵方式表述问题

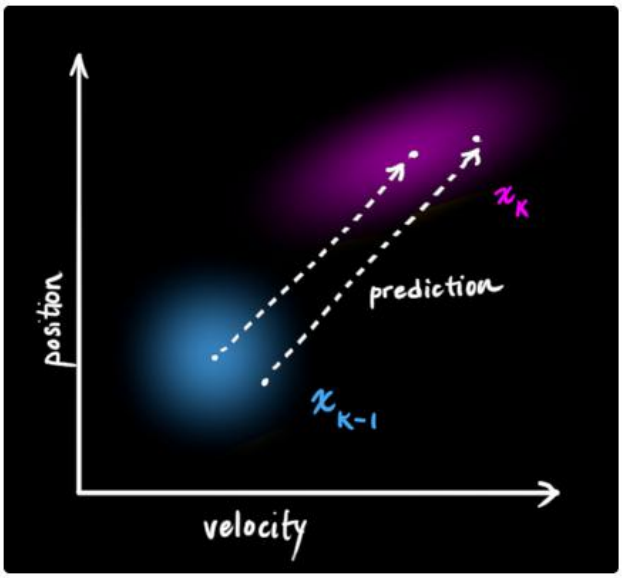
首先我们用以下两个变量定义人 k 时刻的状态（状态矩阵和协方差矩阵）：

$$\hat{\mathbf{x}}_k = \begin{bmatrix} \text{position} \\ \text{velocity} \end{bmatrix}$$

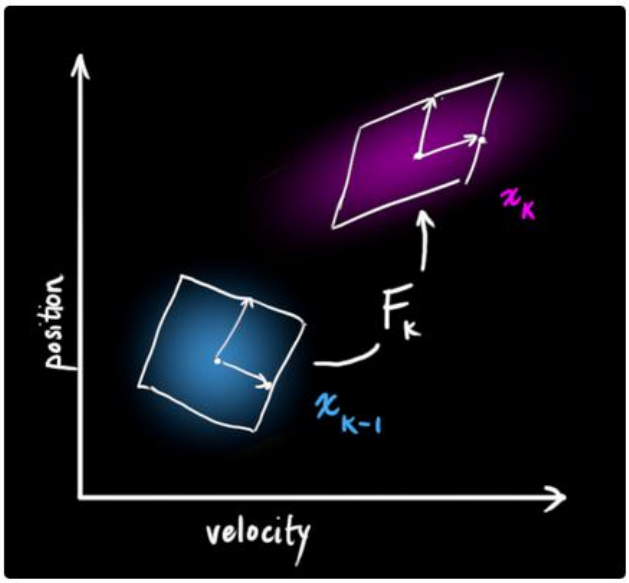
$$\mathbf{P}_k = \begin{bmatrix} \Sigma_{pp} & \Sigma_{pv} \\ \Sigma_{vp} & \Sigma_{vv} \end{bmatrix}$$

切记：k 时刻的状态由两个矩阵表示，一个表示变量值，一个表示变量间关系

现在，我们需要根据 k-1 时刻的状态预测下一时刻（k 时刻）的状态。



这里我们用一个状态转移矩阵 F_k 来表述从 k-1 时刻到 k 时刻状态量的改变



那么这个状态转移矩阵 F_k 一般怎么得到呢？我们以匀速运动为例，首先根据运动规律写出表达式，并转换成矩阵形式表示：

$$\begin{aligned} p_k &= p_{k-1} + \Delta t v_{k-1} \\ v_k &= v_{k-1} \end{aligned} \quad \rightarrow \quad \begin{aligned} \hat{\mathbf{x}}_k &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \hat{\mathbf{x}}_{k-1} \\ &= \mathbf{F}_k \hat{\mathbf{x}}_{k-1} \end{aligned}$$

这里现在我们已经求得了预测矩阵(或称为状态转移矩阵)，但是协方差矩阵还没有更新，为了更新协方差矩阵，我们需要一个新的公式：

$$Cov(x) = \Sigma$$
$$Cov(\mathbf{A}x) = \mathbf{A}\Sigma\mathbf{A}^T$$

根据上述公式即可推出更新后的协方差矩阵

$$\hat{\mathbf{x}}_k = \mathbf{F}_k\hat{\mathbf{x}}_{k-1}$$
$$\mathbf{P}_k = \mathbf{F}_\mathbf{k}\mathbf{P}_{k-1}\mathbf{F}_k^T$$

五、外部影响

到目前为止，我们考虑的都是理想情况，但实际操作中，外部世界的一些因素也能对系统产生影响。以火车运行为例，火车的操作员可能会控制运行开关使得火车加速。类似的，在此案例中，人也不可能做到完全匀速，因此我们假设在从 k-1 时刻到 k 时刻很短的一段时间内，人运动具有加速度 a，则之前所述的运行方程可更新为:

$$p_k = p_{k-1} + \Delta t v_{k-1} + \frac{1}{2}a\Delta t^2$$
$$v_k = v_{k-1} + a\Delta t$$

$$\downarrow$$

$$\hat{\mathbf{x}}_k = \mathbf{F}_k\hat{\mathbf{x}}_{k-1} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} a$$
$$= \mathbf{F}_k\hat{\mathbf{x}}_{k-1} + \mathbf{B}_k\overrightarrow{\mathbf{u}_k}$$

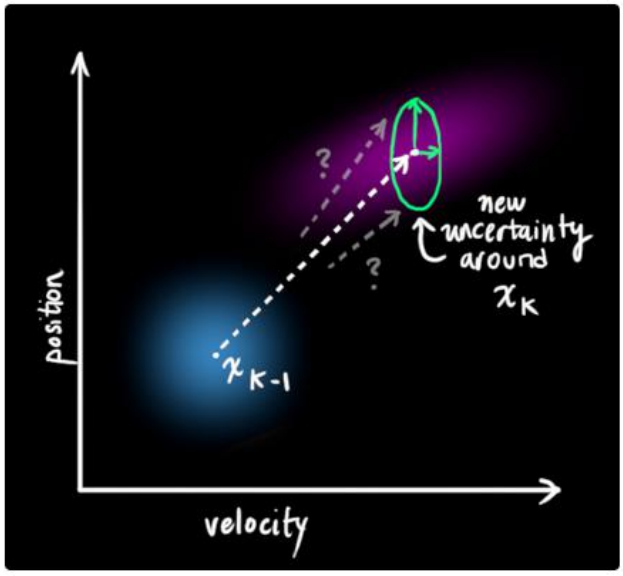
在此 $\overrightarrow{u_k}$ 处代表控制变量，即加速度 a（在实际中，可能是一个 n×1 的矩阵）

B_k 则定义为控制矩阵

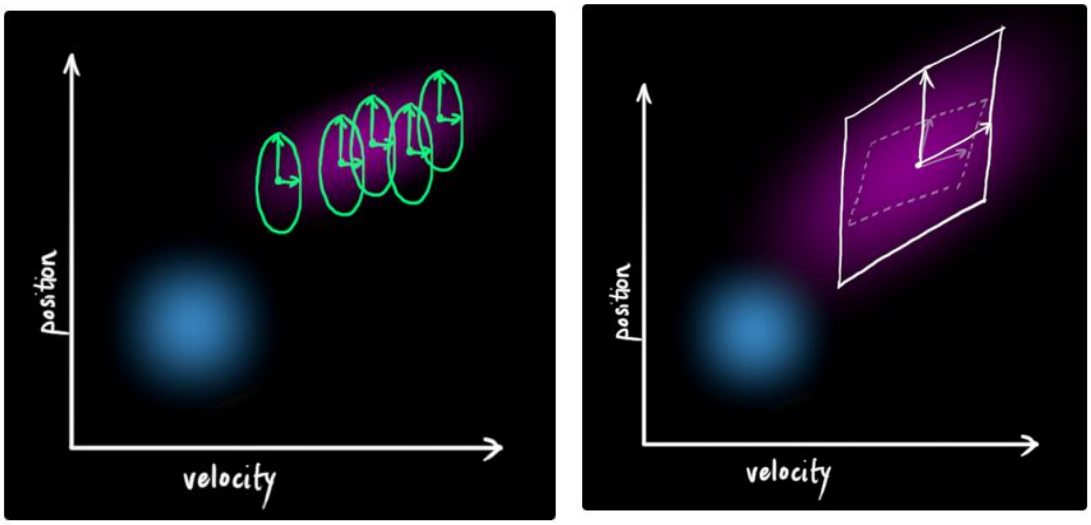
六、外部不确定性

现在在让我们思考另外一个问题，如果我们预测并不是一个 100%精准的模型,那么将会发生什么？ 仍然以人的运动为例，在运动过程中人可能撞到地面上的隆起物(地面凹凸不平)使速度变慢，或者在地面光滑处打滑，我们很难表示或追踪这些因素所产生的影响。一旦这些情况发生,我们的预测结果就很有可能与实际结果产生很大的偏差，因为我们并未在数学模型中考虑到这些因素。

在数学上为了解决上述问题，我们在每一次的预测步骤中加入新的不确定性变量来表示那些存在但是我们无法明确表示的变量所带来的影响。



因此原始估计中的每一个状态点通过变换方程都会运动到一个新的状态范围中，因此预测实际上是从范围到范围的映射。



如上图所示,上一时刻状态 $\hat{\mathbf{x}}_{k-1}$ 中的每一个状态点(蓝色区域表示)都会移动到一个新的区域范围中(无数个绿色圆的集合)。

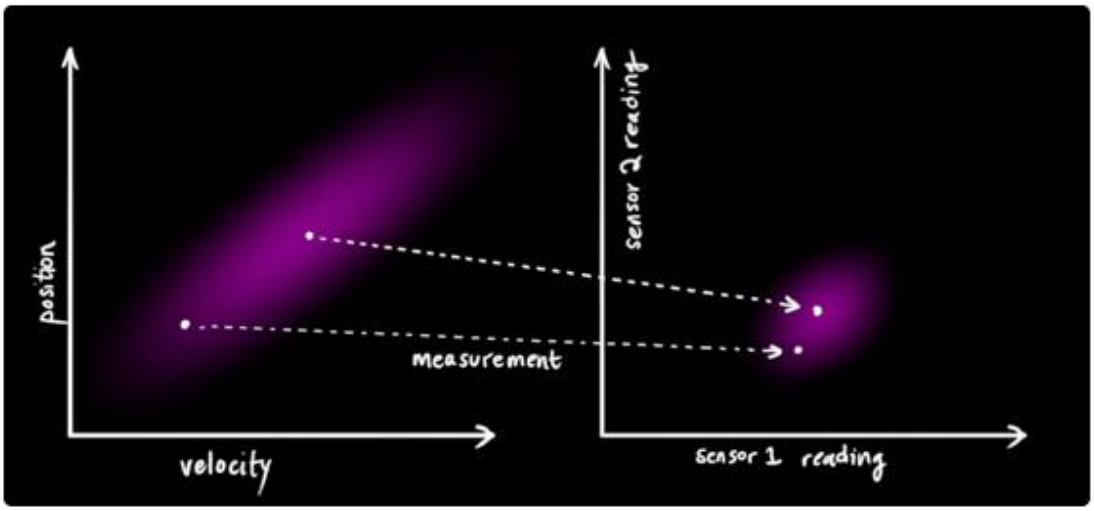
我们将那些存在但无法准确表示且会对状态变量带来不确定性的影响视作噪声，通常为白噪声，噪声的协方差用 Q_k 表示。

到这里，我们已经推得贝尔曼滤波器五个基础公式中的前两个。

现在我们已经得到了系统的预测(估计)值了。但是在真实的系统中，我们往往还能通过传感器得到一些能反映系统状态的测量值。

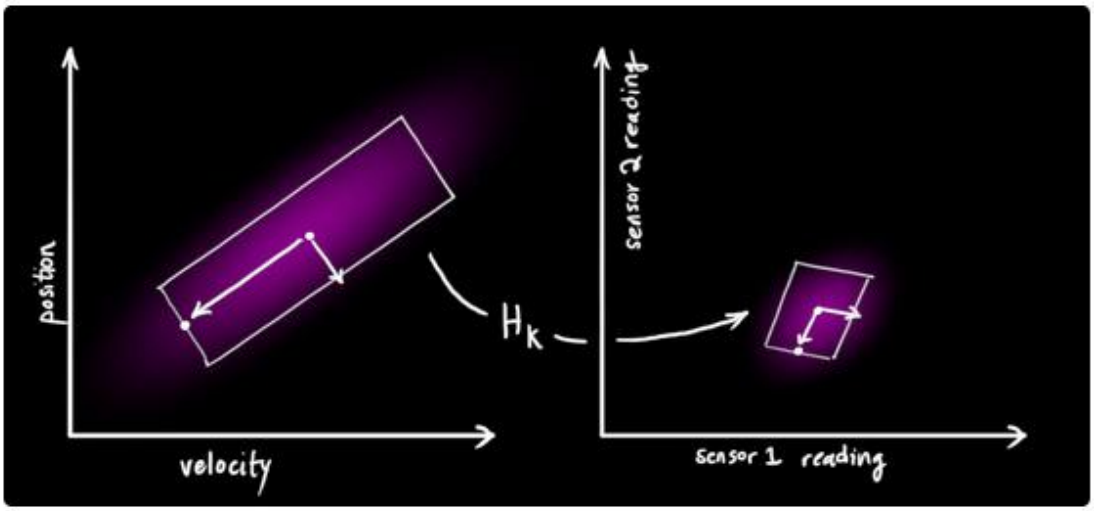
七、测量值与观测值

通常我们会在物体上安装一些传感器，这些传感器的返回值(测量量)能够让我们了解更多与物体当前状态有关的信息。例如，我们现在有两个传感器，一个返回位置信息另一个返回速度信息，这两个传感器都能够间接的提供一些物体运动状态的信息。

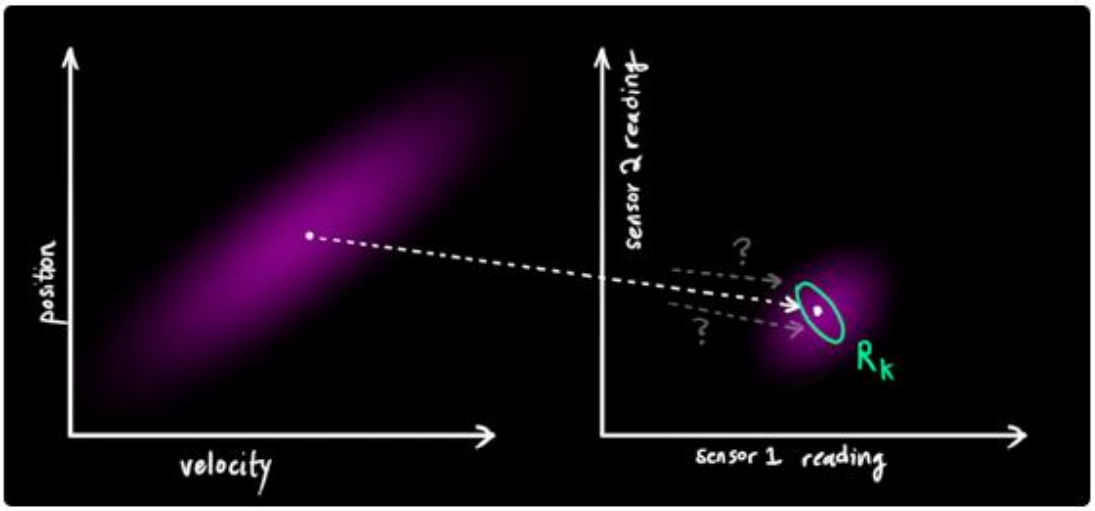


但是需要注意的是,传感器返回的信息其单位和尺度可能与我们在预测步骤中所跟踪的状态量的单位和尺度有所不同。因此通常我们使用矩阵 H_k（一般通过生产商的生产说明得到）来表示状态量与传感器所观测的测量量之间的关系，则根据 H_k，我们就能根据状态值预测可能的观测值：

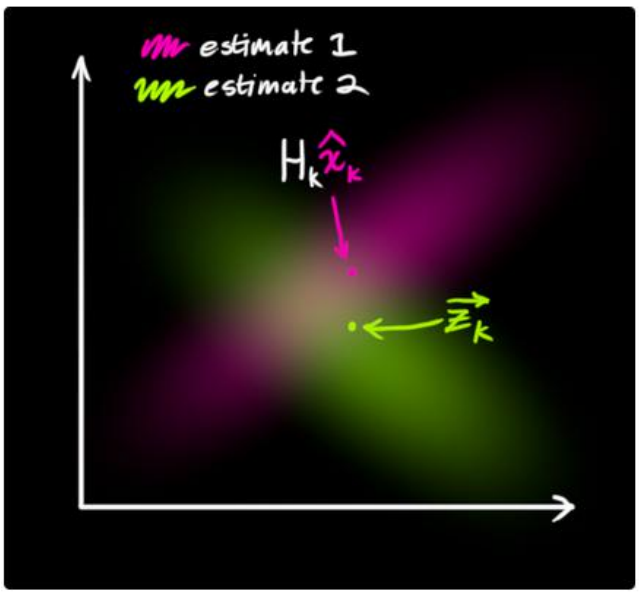
$$\vec{\mu}_{\text{expected}} = \mathbf{H}_k \hat{\mathbf{x}}_k$$
$$\mathbf{\Sigma}_{\text{expected}} = \mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^T$$



需要注意的是传感器测量值是具有一定程度的不可靠性的(由噪声引起)，因此传感器的读数是并非是一个精确值,而是一个带有不确定区域的范围。



在卡尔曼滤波器中我们用符号 R_k 来表示传感器因为噪声所带来的不确定性的协方差，而传感器观测值的均值就等于传感器读数值，通常用 $\vec{z_k}$ 来进行表示。因此，现在我们有了两个不同的高斯分布:一个是由状态值预测产生（粉色），一个由传感器观测得到（绿色）。

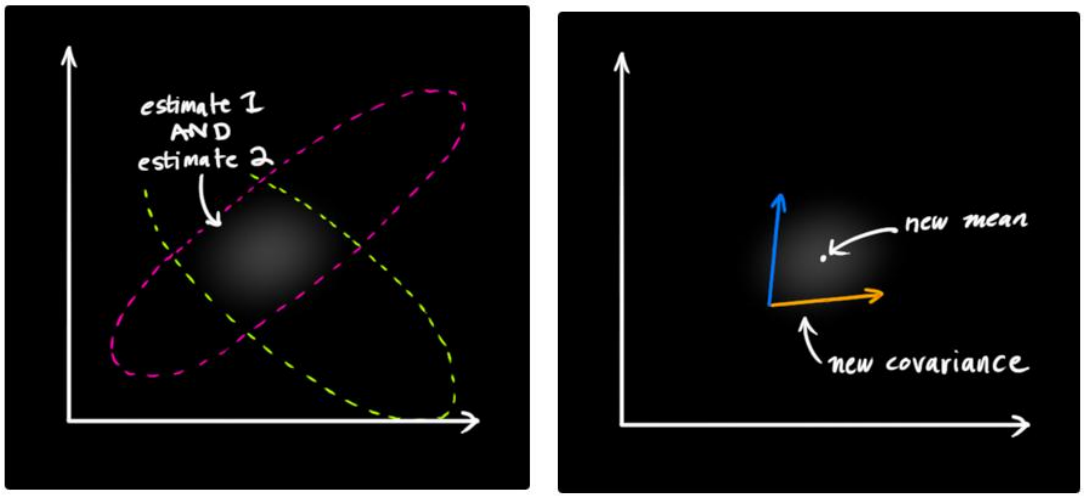


那么现在哪一种才是当前最有可能的状态情况呢？对于某一组传感器读数(Z1，Z2)，我们现在有两种可能：

1、当前的传感器测量为错误的，不可以很好的代表物体的状态(当前物体的状态应该完全与预测步骤的分布相同);

2. 当前的传感器测量为正确的，可以很好的代表当前物体的状态(当前物体的状态应该完全与传感器观测分布相同)；

如果现在我们假设这两种可能行都是正确的，我们将其分布相乘，就能得到一个新的如下所示的分布：



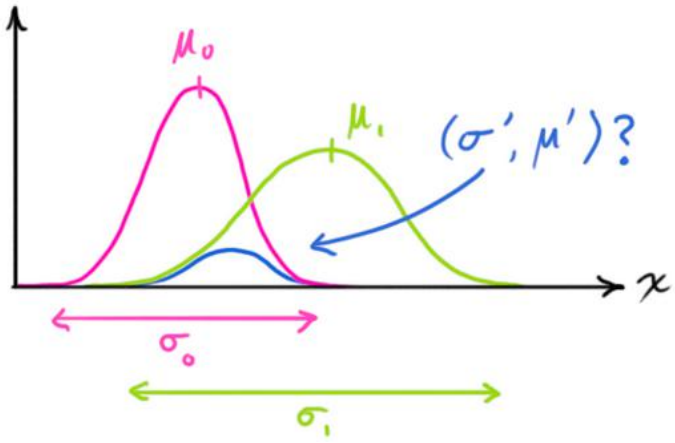
相乘后重合区域为高亮部分,在此区域中两种假设情况都是高亮，这就意味这这个新生成的分布区域比之前任一一种假设都要精确。因此该区域为对物体当前状态的最好估计，因此真正的观测值实际上是结合测量值与预测值得到的。

八、结合高斯

首先考虑一维高斯分布情况

$$\mathcal{N}(x,\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

现在我们想要知道，当我们讲两个高斯曲线(分布)相乘时会发生什么。下图中的蓝色部分表示相乘后得到的新的高斯分布的结果。



$$\mathcal{N}(x, \mu_0, \sigma_0) \cdot \mathcal{N}(x, \mu_1, \sigma_1) \stackrel{?}{=} \mathcal{N}(x, \mu', \sigma')$$

经过计算得：

$$\mathbf{k} = \frac{\sigma_0^2}{\sigma_0^2 + \sigma_1^2}$$

$$\mu' = \mu_0 + \mathbf{k}(\mu_1 - \mu_0)$$
$$\sigma'^2 = \sigma_0^2 - \mathbf{k}\sigma_0^2$$

考虑多元高斯分布：

$$\mathbf{K} = \Sigma_0(\Sigma_0 + \Sigma_1)^{-1}$$

$$\vec{\mu}' = \vec{\mu_0} + \mathbf{K}(\vec{\mu_1} - \vec{\mu_0})$$
$$\Sigma' = \Sigma_0 - \mathbf{K}\Sigma_0$$

K 为卡尔曼增益

九、最后的推导

由状态值预测的观测值分布为：

$$(\mu_0, \Sigma_0) = (\mathbf{H}_k\hat{\mathbf{x}}_k, \mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^T)$$

由传感器测量的测量值分布为：

$$(\mu_1, \Sigma_1) = (\overrightarrow{\mathbf{z}_k}, \mathbf{R}_k)$$

代入上面得到公式：

$$\mathbf{K} = \Sigma_0(\Sigma_0 + \Sigma_1)^{-1}$$

$$\begin{aligned}\vec{\mu}' &= \overrightarrow{\mu_0} + \mathbf{K}(\overrightarrow{\mu_1} - \overrightarrow{\mu_0}) \\ \mathbf{\Sigma}' &= \Sigma_0 - \mathbf{K}\Sigma_0\end{aligned}$$

解得：

$$\mathbf{K} = \mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^T + \mathbf{R}_k)^{-1}$$

$$\begin{aligned}\mathbf{H}_k\hat{\mathbf{x}}_k' &= \mathbf{H}_k\hat{\mathbf{x}}_k &&+\mathbf{K}(\overrightarrow{\mathbf{z}_k} - \mathbf{H}_k\hat{\mathbf{x}}_k) \\ \mathbf{H}_k\mathbf{P}_k'\mathbf{H}_k^T &= \mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^T &&-\mathbf{K}\mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^T\end{aligned}$$

化简，消除 H_k，得到：

$$\begin{aligned}\hat{\mathbf{x}}_k' &= \hat{\mathbf{x}}_k + \mathbf{K}'(\overrightarrow{\mathbf{z}_k} - \mathbf{H}_k\hat{\mathbf{x}}_k) \\ \mathbf{P}_k' &= \mathbf{P}_k - \mathbf{K}'\mathbf{H}_k\mathbf{P}_k\end{aligned}$$

$$\mathbf{K}' = \mathbf{P}_k\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^T + \mathbf{R}_k)^{-1}$$

十、推导总结

（1）预测：

$$\begin{aligned}\hat{\mathbf{x}}_k &= \mathbf{F}_k\hat{\mathbf{x}}_{k-1} + \mathbf{B}_k\overrightarrow{\mathbf{u}_k} \\ \mathbf{P}_k &= \mathbf{F_k}\mathbf{P}_{k-1}\mathbf{F}_k^T + \mathbf{Q}_k\end{aligned}$$

（2）更新：

$$\begin{aligned}\hat{\mathbf{x}}_k' &= \hat{\mathbf{x}}_k + \mathbf{K}'(\overrightarrow{\mathbf{z}_k} - \mathbf{H}_k\hat{\mathbf{x}}_k) \\ \mathbf{P}_k' &= \mathbf{P}_k - \mathbf{K}'\mathbf{H}_k\mathbf{P}_k\end{aligned}$$

$$\mathbf{K}' = \mathbf{P}_k\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^T + \mathbf{R}_k)^{-1}$$

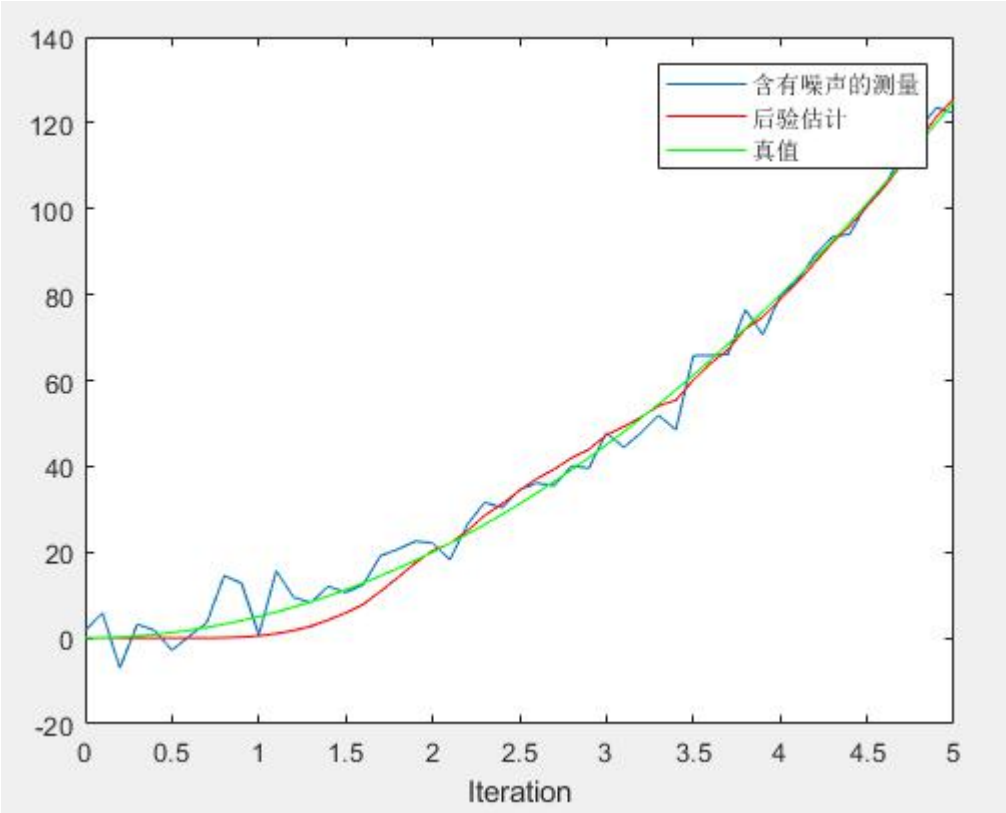
十一、代码仿真

可以跑一下代码，该表数值，看看结果变化，代码比推导好理解

也提前熟悉一下怎么把推导用到编程上

```
% 初始化参数
delta_t=0.1;   %采样时间
t=0:delta_t:5;
N = length(t); % 序列的长度
sz = [2,N];    % 信号需开辟的内存空间大小  2 行*N 列  2:为状态向量的维数 n
g=10;          %加速度值
x=1/2*g*t.^2;      %实际真实位置
z = x + sqrt(10).*randn(1,N); % 测量时加入测量白噪声

Q =[0 0;0 9e-1]; %假设建立的模型  噪声方差叠加在速度上（比如物体行进过程中磕碰引起微小的速度变化） 大小为 n*n 方阵 n=状态向量的维数
R = 10;    % 位置测量噪声方差，可以改变它来看不同效果  m*m      m=z(i)的维数

A=[1 delta_t;0 1];  % n*n
B=[1/2*delta_t^2;delta_t];
H=[1,0];           % m*n，这里表示传感器和预测的单位和尺度是相同的

n=size(Q);  %n 为一个 1*2 的向量  Q 为方阵
m=size(R);

% 分配空间
xhat=zeros(sz);       % x 的后验估计
P=zeros(n);           % 后验方差估计  n*n
xhatminus=zeros(sz);  % x 的先验估计
Pminus=zeros(n);      % n*n
K=zeros(n(1),m(1));   % Kalman 增益  n*m
I=eye(n);

% 估计的初始值都为默认的 0，即 P=[0 0;0 0],xhat=0
for k = 9:N           %假设车子已经运动 9 个 delta_T 了，我们才开始估计
    % 时间更新过程
    xhatminus(:,k) = A*xhat(:,k-1)+B*g;
    Pminus= A*P*A'+Q;

    % 测量更新过程
    K = Pminus*H'*inv( H*Pminus*H'+R );
    xhat(:,k) = xhatminus(:,k)+K*(z(k)-H*xhatminus(:,k));
    P = (I-K*H)*Pminus;
```

```
end

figure
plot(t,z);
hold on
plot(t,xhat(1,:),'r-')
plot(t,x(1,:),'g-');
legend('含有噪声的测量', '后验估计', '真值');
xlabel('Iteration');
```



刚开始时效果不怎么好，后半段基本吻合实际值。