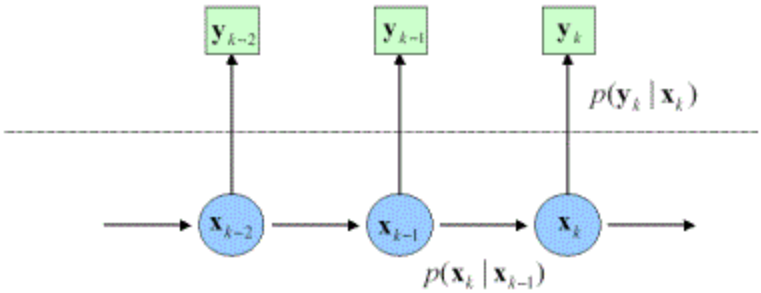
粒子滤波及 matlab 实现

粒子滤波就是指：通过寻找一组在状态空间中传播的随机样本来近似的表示概率密度函数，用样本均值代替积分运算，进而获得系统状态的最小方差估计的过程，这些样本被形象的称为“粒子”，故而叫粒子滤波。粒子滤波通过非参数化的蒙特卡洛(Monte Carlo)模拟方法来实现递推贝叶斯滤波，适用于任何能用状态空间模型描述的非线性系统，精度可以逼近最优估计。粒子滤波器具有简单、易于实现等特点，它为分析非线性动态系统提供了一种有效的解决方法，从而引起目标跟踪、信号处理以及自动控制等领域的广泛关注。

贝叶斯滤波

动态系统的目标跟踪问题可以通过下图所示的状态空间模型来描述。



在目标跟踪问题中，动态系统的状态空间模型可描述为

$$x_k = f(x_{k-1}) + u_{k-1}$$
$$y_k = h(x_k) + v_k$$

其中 $f(\cdot), h(\cdot)$ 分别为状态转移方程与观测方程，x_k 为系统状态，y_k 为观测值，u_k 为过程噪声，v_k 为观测噪声。为了描述方便，用 $X_k = x_{0:k} = \{x_0, x_1, \cdots, x_k\}$ 与 $Y_k = y_{1:k} = \{y_1, \cdots, y_k\}$ 分别表示 0 到 k 时刻所有的状态与观测值。在处理目标跟踪问题时，通常假设目标的状态转移过程服从一阶马尔可夫模型，即当前时刻的状态 x_k 只与上一时刻的状态 x_{k-1} 有关。另外一个假设为观测值相互独立，即观测值 y_k 只与 k 时刻的状态 x_k 有关。

贝叶斯滤波为非线性系统的状态估计问题提供了一种基于概率分布形式的解决方案。贝叶斯滤波将状态估计视为一个概率推理过程，即将目标状态的估计问题转换为利用贝叶斯公式求解后验概率密度 $p(X_k \mid Y_k)$ 或滤波概率密度 $p(x_k \mid Y_k)$，进而获得目标状态的最优

估计。贝叶斯滤波包含预测和更新两个阶段，预测过程利用系统模型预测状态的先验概率密度，更新过程则利用最新的测量值对先验概率密度进行修正，得到后验概率密度。

假设已知$k-1$时刻的概率密度函数为$p(x_{k-1}|Y_{k-1})$，贝叶斯滤波的具体过程如下：

(1) 预测过程，由$p(x_{k-1}|Y_{k-1})$得到$p(x_k|Y_{k-1})$：

$$p(x_k,x_{k-1}|Y_{k-1})=p(x_k|x_{k-1},Y_{k-1})p(x_{k-1}|Y_{k-1})$$

当给定x_{k-1}时，状态x_k与Y_{k-1}相互独立，因此

$$p(x_k,x_{k-1}|Y_{k-1})=p(x_k|x_{k-1})p(x_{k-1}|Y_{k-1})$$

上式两端对x_{k-1}积分，可得 Chapman-Komolgorov 方程

$$p(x_k|Y_{k-1})=\int p(x_k|x_{k-1})p(x_{k-1}|Y_{k-1})\mathrm{d}x_{k-1}$$

(2) 更新过程，由$p(x_k|Y_{k-1})$得到$p(x_k|Y_k)$：

获取k时刻的测量y_k后，利用贝叶斯公式对先验概率密度进行更新，得到后验概率

$$p(x_k|Y_k)=\frac{p(y_k|x_k,Y_{k-1})p(x_k|Y_{k-1})}{p(y_k|Y_{k-1})}$$

假设y_k只由x_k决定，即

$$p(y_k|x_k,Y_{k-1})=p(y_k|x_k)$$

因此

$$p(x_k|Y_k)=\frac{p(y_k|x_k)p(x_k|Y_{k-1})}{p(y_k|Y_{k-1})}$$

其中，$p(y_k|Y_{k-1})$为归一化常数

$$p(y_k|Y_{k-1})=\int p(y_k|x_k)p(x_k|Y_{k-1})\mathrm{d}x_k$$

贝叶斯滤波以递推的形式给出后验(或滤波)概率密度函数的最优解。目标状态的最优估计值可由后验(或滤波)概率密度函数进行计算。通常根据极大后验(MAP)准则或最小均方误差(MMSE)准则，将具有极大后验概率密度的状态或条件均值作为系统状态的估计值，即

$$\hat{x}_k^{MAP}=\arg\min_{x_k} p(x_k|Y_k)$$

$$\hat{x}_k^{MMSE}=\mathrm{E}[f(x_k)|Y_k]=\int f(x_k)p(x_k|Y_k)\mathrm{d}x_k$$

贝叶斯滤波需要进行积分运算，除了一些特殊的系统模型（如线性高斯系统，有限状态的离散系统）之外，对于一般的非线性、非高斯系统，贝叶斯滤波很难得到后验概率的封闭解析式。因此，现有的非线性滤波器多采用近似的计算方法解决积分问题，以此来获

取估计的次优解。在系统的非线性模型可由在当前状态展开的线性模型有限近似的前提下，基于一阶或二阶Taylor级数展开的扩展Kalman滤波得到广泛应用。在一般情况下，逼近概率密度函数比逼近非线性函数容易实现。据此，Julier与Uhlmann提出一种Unscented Kalman滤波器，通过选定的sigma点来精确估计随机变量经非线性变换后的均值和方差，从而更好的近似状态的概率密度函数，其理论估计精度优于扩展Kalman滤波。获取次优解的另外一中方案便是基于蒙特卡洛模拟的粒子滤波器。

粒子滤波

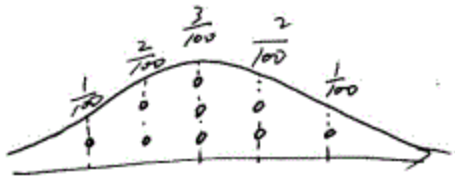
早在 20 世纪 50 年代，Hammersley 便采用基于序贯重要性采样(Sequential importance sampling,SIS)的蒙特卡洛方法解决统计学问题[121]。20 世纪 60 年代后期，Handschin 与 Mayne 使用序贯蒙特卡洛方法解决自动控制领域的相关问题。20 世纪 70 年代，Handschin、Akashi 以及 Zaritskii 等学者的一系列研究工作使得序贯蒙特卡洛方法得到进一步发展。限于当时的计算能力以及算法本身存在的权值退化问题，序贯重要性采样算法没有受到足够重视，在随后较长一段时间内进展较为缓慢。直到 20 世纪 80 年代末，计算机处理能力的巨大进展使得序贯蒙特卡洛方法重新受到关注。Tanizaki、Geweke 等采用基于重要性采样的蒙特卡洛方法成功解决了一系列高维积分问题。Smith 与 Gelfand 提出的采样-重采样思想为 Bayesian 推理提供了一种易于实现的计算策略。随后，Smith 与 Gordon 等人合作，于 20 世纪 90 年代初将重采样(Resampling)步骤引入到粒子滤波中，在一定程度上解决了序贯重要性采样的权值退化问题，并由此产生了第一个可实现的 SIR(Sampling importance resampling)粒子滤波算法(Bootstrap 滤波)，从而掀起粒子滤波的研究热潮。美国海军集成水下监控系统中的 Nodestar 便是粒子滤波应用的一个实例。进入 21 世纪，粒子滤波器成为一个非常活跃的研究领域，Doucet、Liu、Arulampalam 等对粒子滤波的研究作了精彩的总结，IEEE 出版的论文集“Sequential Monte Carlo Methods in Practice”对粒子滤波器进行了详细介绍。

贝叶斯重要性采样

蒙特卡洛模拟是一种利用随机数求解物理和数学问题的计算方法，又称为计算机随机模拟方法。该方法源于第一次世界大战期间美国研制原子弹的曼哈顿计划，著名数学家冯诺伊曼作为该计划的主持人之一，用驰名世界的赌城，摩纳哥的蒙特卡洛来命名这种方法。蒙特卡洛模拟方法利用所求状态空间中大量的样本点来近似逼近待估计变量的后验概率分布，如图 2.2 所示，从而将积分问题转换为有限样本点的求和问题。粒子滤波算法的核心思想便是利用一系列随机样本的加权和表示后验概率密度，通过求和来近似积分操作。假

设可以从后验概率密度 $p(x_k \mid Y_k)$ 中抽取 N 个独立同分布的随机样本 $x_k^{(i)}$，$i=1,\cdots,N$，则有

$$p(x_k \mid Y_k) \approx \frac{1}{N}\sum_{i=1}^{N}\delta(x_k - x_k^{(i)}) \qquad \text{* MERGEFORMAT (0.1)}$$



这里 x_k 为连续变量，$\delta(x\text{-}x_k)$ 为单位冲激函数(狄拉克函数)，即 $\delta(x\text{-}x_k)=0, x \neq x_k$，且 $\int \delta(x)\mathrm{d}x = 1$。当 x_k 为离散变量时，后验概率分布 $P(x_k \mid Y_k)$ 可近似逼近为

$$P(x_k \mid Y_k) \approx \frac{1}{N}\sum_{i=1}^{N}\delta(x_k - x_k^{(i)}) \qquad \text{* MERGEFORMAT (0.2)}$$

其中，$\delta(x_k - x_k^{(i)}) = 1, x_k = x_k^{(i)}$；$\delta(x_k - x_k^{(i)}) = 0, x_k \neq x_k^{(i)}$。

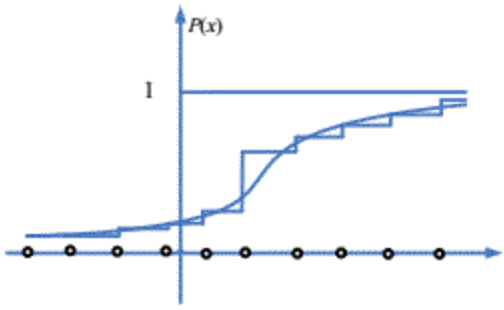


图 2.1　经验概率分布函数

Fig. 2.1　Empirical probability distribution function

设 $x_k^{(i)}$ 为从后验概率密度函数 $p(x_k \mid Y_k)$ 中获取的采样粒子，则任意函数 $f(x_k)$ 的期望估计可以用求和方式逼近，即

$$\mathrm{E}[f(x_k)|Y_k] = \int f(x_k)p(x_k \mid Y_k)\mathrm{d}x_k = \frac{1}{N}\sum_{i=1}^{N} f(x_k^{(i)}) \qquad \text{* MERGEFORMAT (0.3)}$$

蒙特卡洛方法一般可以归纳为以下三个步骤：

(1)构造概率模型。对于本身具有随机性质的问题，主要工作是正确地描述和模拟这个概率过程。对于确定性问题，比如计算定积分、求解线性方程组、偏微分方程等问题，采用蒙特卡洛方法求解需要事先构造一个人为的概率过程，将它的某些参量视为问题的解。

(2)从指定概率分布中采样。产生服从已知概率分布的随机变量是实现蒙特卡洛方法模拟试验的关键步骤。

(3)建立各种估计量的估计。一般说来，构造出概率模型并能从中抽样后，便可进行现模拟试验。随后，就要确定一个随机变量，将其作为待求解问题的解进行估计。

在实际计算中，通常无法直接从后验概率分布中采样，如何得到服从后验概率分布的随机样本是蒙特卡洛方法中基本的问题之一。重要性采样法引入一个已知的、容易采样的重要性概率密度函数$q(x_k \mid Y_k)$，从中生成采样粒子，利用这些随机样本的加权和来逼近后验滤波概率密度$p(x_k \mid Y_k)$，如图2.3所示。令$\{x_k^{(i)}, w_k^{(i)}, i=1,\cdots.N\}$表示一支撑点集，其中$x_k^{(i)}$为是$k$时刻第$i$个粒子的状态，其相应的权值为$w_k^{(i)}$，则后验滤波概率密度可以表示为

$$p(x_k \mid Y_k) = \sum_{i=1}^{N} w_k^{(i)} \delta(x_k - x_k^{(i)}) \qquad \text{* MERGEFORMAT (0.4)}$$

其中，

$$w_k^{(i)} \propto \frac{p(x_k^{(i)} \mid Y_k)}{q(x_k^{(i)} \mid Y_k)} \qquad \text{* MERGEFORMAT (0.5)}$$

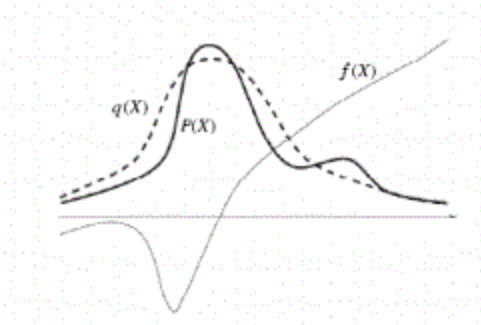


图 2.2　重要性采样

Fig. 2.2　Importance sampling

当采样粒子的数目很大时，式(2.14)便可近似逼近真实的后验概率密度函数。任意函数$f(x_k)$的期望估计为

$$\mathrm{E}[f(x_k)|Y_k] = \frac{1}{N}\sum_{i=1}^{N} f(x_k^{(i)}) \frac{p(x_k^{(i)} \mid Y_k)}{q(x_k^{(i)} \mid Y_k)} = \frac{1}{N}\sum_{i=1}^{N} f(x_k^{(i)}) w_k^{(i)} \qquad \text{* MERGEFORM}$$

AT (0.6)

序贯重要性采样算法

在基于重要性采样的蒙特卡洛模拟方法中，估计后验滤波概率需要利用所有的观测数据，每次新的观测数据来到都需要重新计算整个状态序列的重要性权值。序贯重要性采样作为粒子滤波的基础，它将统计学中的序贯分析方法应用到的蒙特卡洛方法中，从而实现后验滤波概率密度的递推估计。假设重要性概率密度函数 $q(x_{0:k} \mid y_{1:k})$ 可以分解为

$$q(x_{0:k} \mid y_{1:k}) = q(x_{0:k-1} \mid y_{1:k-1}) q(x_k \mid x_{0:k-1}, y_{1:k}) \quad \text{* MERGEFORMAT (0.7)}$$

设系统状态是一个马尔可夫过程，且给定系统状态下各次观测独立，则有

$$p(x_{0:k}) = p(x_0) \prod_{i=1}^{k} p(x_i \mid x_{i-1}) \quad \text{* MERGEFORMAT (0.8)}$$

$$p(y_{1:k} \mid x_{1:k}) = \prod_{i=1}^{k} p(y_i \mid x_i) \quad \text{* MERGEFORMAT (0.9)}$$

后验概率密度函数的递归形式可以表示为

$$p(x_{0:k} \mid Y_k) = \frac{p(y_k \mid x_{0:k}, Y_{k-1}) p(x_{0:k} \mid Y_{k-1})}{p(y_k \mid Y_{k-1})}$$

$$= \frac{p(y_k \mid x_{0:k}, Y_{k-1}) p(x_k \mid x_{0:k-1}, Y_{k-1}) p(x_{0:k-1} \mid Y_{k-1})}{p(y_k \mid Y_{k-1})}$$

$$= \frac{p(y_k \mid x_k) p(x_k \mid x_{k-1}) p(x_{0:k-1} \mid Y_{k-1})}{p(y_k \mid Y_{k-1})} \quad \text{* MERGEFORMAT (0.10)}$$

粒子权值 $w_k^{(i)}$ 的递归形式可以表示为

$$w_k^{(i)} \propto \frac{p(x_{0:k}^{(i)} \mid Y_k)}{q(x_{0:k}^{(i)} \mid Y_k)}$$

$$= \frac{p(y_k \mid x_k^{(i)}) p(x_k^{(i)} \mid x_{k-1}^{(i)}) p(x_{0:k-1}^{(i)} \mid Y_{k-1})}{q(x_k^{(i)} \mid x_{0:k-1}^{(i)}, Y_k) q(x_{0:k-1}^{(i)} \mid Y_{k-1})}$$

$$= w_{k-1}^{(i)} \frac{p(y_k \mid x_k^{(i)}) p(x_k^{(i)} \mid x_{k-1}^{(i)})}{q(x_k^{(i)} \mid x_{0:k-1}^{(i)}, Y_k)} \quad \text{* MERGEFORMAT (0.11)}$$

通常，需要对粒子权值进行归一化处理，即

$$\tilde{w}_k^{(i)} = \frac{w_k^{(i)}}{\sum_{i=1}^{N} w_k^{(i)}} \quad \text{* MERG}$$

EFORMAT (0.12)

序贯重要性采样算法从重要性概率密度函数中生成采样粒子，并随着测量值的依次到来递推求得相应的权值，最终以粒子加权和的形式来描述后验滤波概率密度，进而得到状态估计。序贯重要性采样算法的流程可以用如下伪代码描述：

$[\{x_k^{(i)}, w_k^{(i)}\}_{i=1}^{N}] = SIS(\{x_{k-1}^{(i)}, w_{k-1}^{(i)}\}_{i=1}^{N}, Y_k)$

For i=1:N

(1)时间更新，根据重要性参考函数 $q(x_k^{(i)} \mid x_{0:k-1}^{(i)}, Y_k)$ 生成采样粒子 $x_k^{(i)}$；

(2)量测更新，根据最新观测值计算粒子权值 $w_k^{(i)}$；

End For

粒子权值归一化，并计算目标状态。

为了得到正确的状态估计，通常希望粒子权值的方差尽可能趋近于零。然而，序贯蒙特卡洛模拟方法一般都存在权值退化问题。在实际计算中，经过数次迭代，只有少数粒子的权值较大，其余粒子的权值可忽略不计。粒子权值的方差随着时间增大，状态空间中的有效粒子数较少。随着无效采样粒子数目的增加，使得大量的计算浪费在对估计后验滤波概率分布几乎不起作用的粒子更新上，使得估计性能下降。通常采用有效粒子数 N_{eff} 来衡量粒子权值的退化程度，即

$$N_{eff} = N/(1+\mathrm{var}(w_k^{*(i)})) \quad \text{* MERG}$$

EFORMAT (0.13)

$$w_k^{*(i)} = \frac{p(x_k^{(i)} \mid y_{1:k})}{q(x_k^{(i)} \mid x_{k-1}^{(i)}, y_{1:k})} \quad \text{* MERGEFORMAT (0.14)}$$

有效粒子数越小，表明权值退化越严重。在实际计算中，有效粒子数 N_{eff} 可以近似为

$$\hat{N}_{eff} \approx \frac{1}{\sum_{i=1}^{N} (w_k^{(i)})^2} \quad \text{* MERGEFORMAT (0.15)}$$

在进行序贯重要性采样时，若 $\hat{N}_{eff}$ 小于事先设定的某一阈值，则应当采取一些措施加以控制。克服序贯重要性采样算法权值退化现象最直接的方法是增加粒子数，而这会造成计算量的相应增加，影响计算的实时性。因此，一般采用以下两种途径：(1)选择合适的重

要性概率密度函数；(2)在序贯重要性采样之后，采用重采样方法。

重要密度函数的选择

重要性概率密度函数的选择对粒子滤波的性能有很大影响，在设计与实现粒子滤波器的过程中十分重要。在工程应用中，通常选取状态变量的转移概率密度函数 $p(x_k \mid x_{k-1})$ 作为重要性概率密度函数。此时，粒子的权值为

$$w_k^{(i)} = w_{k-1}^{(i)} p(y_k \mid x_k^{(i)}) \qquad \text{* MERGEFORMAT (0.16)}$$

转移概率的形式简单且易于实现，在观测精度不高的场合，将其作为重要性概率密度函数可以取得较好的滤波效果。然而，采用转移概率密度函数作为重要性概率密度函数没有考虑最新观测数据所提供的信息，从中抽取的样本与真实后验分布产生的样本存在一定的偏差，特别是当观测模型具有较高的精度或预测先验与似然函数之间重叠部分较少时，这种偏差尤为明显。

选择重要性概率密度函数的一个标准是使得粒子权值 $\{w_k^{(i)}\}_{i=1}^N$ 的方差最小。Doucet 等给出的最优重要性概率密度函数为

$$\begin{aligned} q(x_k^{(i)} \mid x_{k-1}^{(i)}, y_k) &= p(x_k^{(i)} \mid x_{k-1}^{(i)}, y_k) \\ &= \frac{p(y_k \mid x_k^{(i)}, x_{k-1}^{(i)}) p(x_k^{(i)} \mid x_{k-1}^{(i)})}{p(y_k \mid x_{k-1}^{(i)})} \end{aligned}$$

$$= \frac{p(y_k \mid x_k^{(i)}) p(x_k^{(i)} \mid x_{k-1}^{(i)})}{p(y_k \mid x_{k-1}^{(i)})} \qquad \text{* MERGEFORMAT (0.17)}$$

此时，粒子的权值为

$$w_k^{(i)} = w_{k-1}^{(i)} p(y_k \mid x_{k-1}^{(i)}) \qquad \text{* MERGEFORMAT (0.18)}$$

以 $p(x_k^{(i)} \mid x_{k-1}^{(i)}, y_k)$ 作为重要性概率密度函数需要对其直接采样。此外，只有在 x_k 为有限离散状态或 $p(x_k^{(i)} \mid x_{k-1}^{(i)}, y_k)$ 为高斯函数时， $p(y_k \mid x_{k-1}^{(i)})$ 才存在解析解。在实际情况中，构造最优重要性概率密度函数的困难程度与直接从后验概率分布中抽取样本的困难程度等同。从最优重要性概率密度函数的表达形式来看，产生下一个预测粒子依赖于已有的粒子和最新的观测数据，这对于设计重要性概率密度函数具有重要的指导作用，即应该有效利用最新的观测信息，在易于采样实现的基础上，将更多的粒子移动到似然函数值较高的区域，如图 2.4 所示。

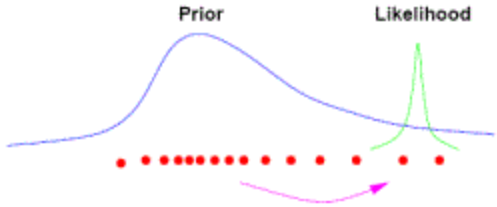


图 2.3　移动粒子至高似然区域

Fig. 2.3　Move the samples in the prior to regions of high likelihood

辅助粒子滤波算法利用 k 时刻的信息，将 $k-1$ 时刻最有前途(预测似然度大)的粒子扩展到 k 时刻[137]，从而生成采样粒子。与 SIR 滤波器相比，当粒子的似然函数位于先验分布的尾部或似然函数形状比较狭窄时，辅助粒子滤波能够得到更精确的估计结果。辅助粒子滤波引入辅助变量 m 来表示 $k-1$ 时刻的粒子列表，应用贝叶斯定理，联合概率密度函数 $p(x_k, m \mid y_{1:k})$ 可以描述为

$$\begin{aligned} p(x_k, m \mid y_{1:k}) &\propto p(y_k \mid x_k) p(x_k, m \mid y_{1:k-1}) \\ &= p(y_k \mid x_k) p(x_t \mid m, y_{1:k-1}) p(m \mid y_{1:k-1}) \\ &= p(y_k \mid x_k^m) p(x_k \mid x_{k-1}^m) w_{k-1}^m \end{aligned} \qquad \text{* MERGEFORMAT (0.19)}$$

生成 $\{x_k^{(i)}, m^{(i)}\}_{i=1}^N$ 的重要性概率密度函数 $q(x_k, m \mid x_{0:k-1}, y_{1:k})$ 为

$$q(x_k, m \mid x_{0:k-1}, y_{1:k}) \propto p(y_k \mid \mu_k^m) p(x_k \mid x_{k-1}^m) w_{k-1}^m \qquad \text{* MERGEFORMAT (0.20)}$$

其中 μ_k^m 为由 $\{x_{k-1}^{(i)}\}_{i=1}^N$ 预测出的与 x_k 相关的特征，可以是采样值 $\mu_k^m \sim p(x_k \mid x_{k-1}^m)$ 或预测均值 $\mu_k^m = E\{x_k \mid x_{k-1}^m\}$。

定义 $q(x_k \mid m, y_{1:k}) = p(x_k \mid x_{k-1}^m)$，由于

$$q(x_k, m \mid y_{1:k}) = q(x_k \mid m, y_{1:k}) q(m \mid y_{1:k}) \qquad \text{* MERGEFORMAT (0.21)}$$

则有

$$q(m \mid y_{1:k}) = p(y_k \mid \mu_k^m) w_{k-1}^m \qquad \text{* MERGEFORMAT (0.22)}$$

此时，粒子权值 $w_k^{(i)}$ 为

$$w_k^{(i)} \propto w_k^{m^{(i)}} \frac{p(y_k \mid x_k^{(i)}) p(x_k^i \mid x_k^{m^{(i)}})}{q(x_k, m \mid x_{0:k-1}^m, y_k)} = \frac{p(y_k \mid x_k^{(i)})}{p(y_k \mid \mu_k^{m^{(i)}})} \qquad \text{* MERGEFORMAT (0.}$$

采用局部线性化的方法来逼近 $p(x_k \mid x_{k-1}, y_k)$ 是另一种提高粒子采样效率的有效方法。扩展Kalman粒子滤波与Uncented粒子滤波算法在滤波的每一步迭代过程中，首先利用最新观测值，采用UKF或者EKF对各个粒子进行更新，得到随机变量经非线性变换后的均值和方差，并将它作为重要性概率密度函数[138]。另外，利用似然函数的梯度信息，采用牛顿迭代[139]或均值漂移[140]等方法移动粒子至高似然区域，也是一种可行的方案，如图2.5所示。以上这些方法的共同特点是将最新的观测数据融入到系统状态的转移过程中，引导粒子到高似然区域，由此产生的预测粒子可较好地服从状态的后验概率分布，从而有效地减少描述后验概率密度函数所需的粒子数。

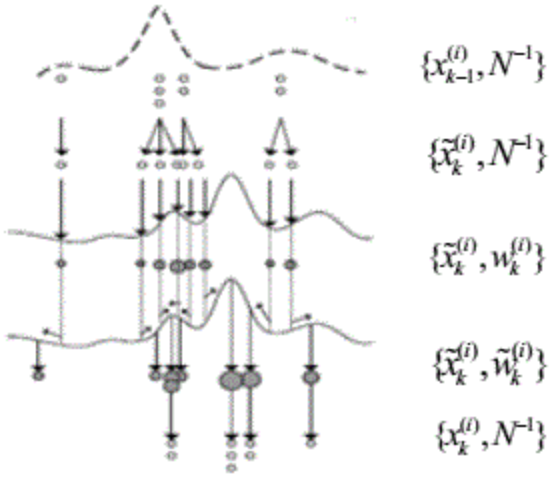


图 2.4　结合均值漂移的粒子滤波算法

Fig. 2.4　Particle filter combined with mean shift

重采样方法

针对序贯重要性采样算法存在的权值退化现象，Gordon 等提出了一种名为 Bootstrap 的粒子滤波算法。该算法在每步迭代过程中，根据粒子权值对离散粒子进行重采样，在一定程度上克服了这个问题。重采样方法舍弃权值较小的粒子，代之以权值较大的粒子。重采样过程在满足 $p(\tilde{x}_k^{(i)} = x_k^{(i)}) = \tilde{w}_k^{(i)}$ 条件下，将粒子集合 $\{\tilde{x}_k^{(i)}, \tilde{w}_k^{(i)}\}_1^N$ 更新为 $\{x_k^{(i)}, 1/N\}_1^N$。重采样策略包括固定时间间隔重采样与根据粒子权值进行的动态重采样。动态重采样通常根据当前的有效粒子数或最大与最小权值比来判断是否需要进行重采样。常用的重采样方法包括多项式(Multinomial resampling)重采样、残差重采样(Residual resampling)、分层重采样(Stratified resampling)与系统重采样(Systematic resampling)等。残余重采样法具有效率高、实现方便的特点。设 $N^i = \lfloor N\tilde{w}_k^{(i)} \rfloor$，其中 $\lfloor \bullet \rfloor$ 为取整操作。残余重采样采用新的权值

$\tilde{w}_k^{*(i)} = \bar{N}_k^{-1}\left(N\tilde{w}_k^{(i)} - N^i\right)$选择余下的$\bar{N}_k = N - \sum_{i=1}^{N} N^i$个粒子，如图 2.6 所示。残余重采样的主要过程为

(1) 计算剩余粒子的权值累计量$\lambda_j, j = 1, \cdots, \bar{N}_k$ 。

(2) 生成$\bar{N}_k$在个[0，1]区间均匀分布的随机数$\{\mu^l\}_{l=1}^{\bar{N}_k}$ ；

(3) 对于每个μ^i，寻找归一化权值累计量大于或等于μ^i的最小标号m，即

$\lambda_{m-1} < \mu^l < \lambda_m$。当$\mu^i$落在区间$[\lambda_{m-1}, \lambda_m]$时，$x_k^m$被复制一次。

这样，每个粒子$x_k^{(i)}$经重采样后的个数为步骤(3)中被选择的若干粒子数目与N^i之和。

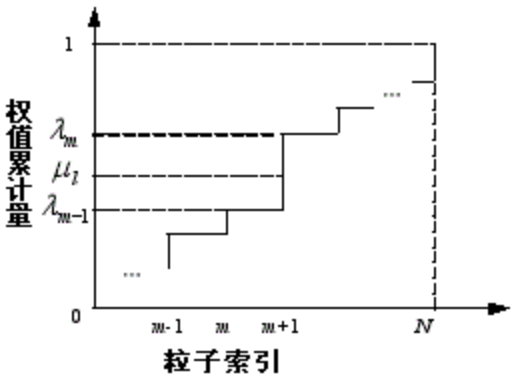


图 2.5　残差重采样

Fig. 2.5　Residual Resampling

重采样并没有从根本上解决权值退化问题。重采样后的粒子之间不再是统计独立关系，给估计结果带来额外的方差。重采样破坏了序贯重要性采样算法的并行性，不利于 VLSI 硬件实现。另外，频繁的重采样会降低对测量数据中野值的鲁棒性。由于重采样后的粒子集中包含了多个重复的粒子，重采样过程可能导致粒子多样性的丧失，此类问题在噪声较小的环境下更加严重。因此，一个好的重采样算法应该在增加粒子多样性和减少权值较小的粒子数目之间进行有效折衷。

图 2.7 为粒子滤波算法的示意图，该图描述了粒子滤波算法包含的时间更新、观测更新和重采样三个步骤。$k-1$时刻的先验概率由N个权值为$1/N$的粒子$x_{k-1}^{(i)}$近似表示。在时间更新过程中，通过系统状态转移方程预测每个粒子在k时刻的状态$\tilde{x}_k^{(i)}$。经过观测值后，更新粒子权值$\tilde{w}_k^{(i)}$。重采样过程舍弃权值较小的粒子，代之以权值较大的粒子，粒子的权值被重新设置为$1/N$。

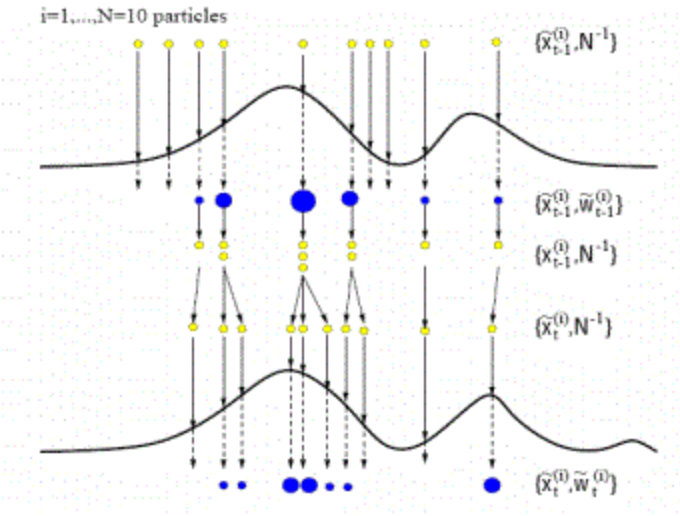


图 2.6　SIR 算法示意图
Fig. 2.6　SIR algorithm

标准的粒子滤波算法流程为：

(1) 粒子集初始化，$k=0$：

对于 $i=1,2,\cdots,N$，由先验 $p(x_0)$ 生成采样粒子 $\{x_0^{(i)}\}_{i=1}^N$

(2) 对于 $k=1,2,\cdots$，循环执行以下步骤：

① 重要性采样：对于 $i=1,2,\cdots,N$，从重要性概率密度中生成采样粒子 $\{\tilde{x}_k^{(i)}\}_{i=1}^N$，计算粒子权值 $\tilde{w}_k^{(i)}$，并进行归一化；

② 重采样：对粒子集 $\{\tilde{x}_k^{(i)}, \tilde{w}_k^{(i)}\}$ 进行重采样，重采样后的粒子集为 $\{x_k^{(i)}, 1/N\}$；

③ 输出：计算 k 时刻的状态估计值：$\hat{x}_k = \sum_{i=1}^{N} \tilde{x}_k^{(i)} \tilde{w}_k^{(i)}$。

粒子滤波中的权值退化问题是不可避免的。虽然重采样方法可以在一定程度上缓解权值退化现象，但重采样方法也会带来一些其它的问题。重采样需要综合所有的粒子才能实现，限制了粒子滤波的并行计算。另外，根据重采样的原则，粒子权值较大的粒子必然会更多的被选中复制，经过若干步迭代后，必然导致相同的粒子越来越多，粒子将缺乏多样性，可能出现粒子退化现象，从而使状态估计产生较大偏差。针对粒子退化问题，一个有效的解决方法是增加马尔可夫蒙特卡洛(Markov chain monte carlo,MCMC)移动步骤[141]。马尔可夫链蒙特卡洛方法(如 Gibbs 采样、Metropolis-Hastings 采样等)利用不可约马尔可夫过程可逆平稳分布的性质，将马尔可夫过程的平稳分布视为目标分布，通过构造马尔可夫链产生来自目标分布的粒子。粒子退化问题是由于重采样使得粒子过分集中在某些状态上而导致的，对重采样后的粒子进行马尔可夫跳转可以提高粒子群的多样性，同时保证跳转后的粒子同样能够准确的描述既定的后验分布。设粒子分布服从后验概率 $p(\tilde{x}_{0:k} \mid y_{1:k})$，实施核为 $k(x_{0:k} \mid \tilde{x}_{0:k})$ 的马尔科夫链变换之后，若满足

$$\int k(x_{0:k} \mid \tilde{x}_{0:k}) p(\tilde{x}_{0:k} \mid y_{1:k}) \mathrm{d}\tilde{x}_{0:k} = p(x_{0:k} \mid y_{1:k}) \quad \text{* MERGEFORMAT (0.24)}$$

便可以得到一组满足既定后验概率分布的粒子集合，且这组新的粒子可能移动到状态空间中更为有利的位置。

采用 Metropolis-Hastings 算法，从概率密度 $q(x)$ 中生成粒子的具体步骤为：

(1) 从[0,1]之间的均匀分布中生成随机数 $v \sim U[0,1]$；

(2) 从重要性概率密度函数中生成采样粒子 $x_k^* \sim p(x_k^* \mid x_k^{(i)})$；

(3) 如果 $v \le \min\left\{1, \frac{q(x_k^*) p(x_k^{(i)} \mid x_k^*)}{q(x_k^{(i)}) p(x_k^* \mid x_k^{(i)})}\right\}$，接受 x_k^*，否则，拒绝 x_k^*

另一种解决粒子退化问题的方法是正则化粒子滤波[142]。传统的重采样方法是在离散分布中采样实现，即

$$\{x_k^{(i)}, w_k^{(i)}\} \sim p(x_{0k} \mid y_{1:k}) = \frac{1}{N} \sum_{i=1}^{N} \omega_k^{(i)} \delta(x_k - x_k^{(i)}) \quad \text{* MERGEFORMAT (0.25)}$$

在过程噪声比较小的情况下，传统的粒子滤波方法(如SIR方法)的粒子退化现象比较严重；正则化粒子滤波首先采用密度估计理论计算后验密度的连续分布，然后从连续分布中采样来生成采样粒子，以提高粒子集的多样性，即

$$\{x_k^{(i)}, w_k^{(i)}\} \sim p(x_k \mid y_{1:k}) = \frac{1}{N} \sum_{i=1}^{N} \omega_k^{(i)} K_h(x_k - x_k^{(i)}) \quad \text{* MERGEFORMAT (0.26)}$$

其中 $K_h = \frac{1}{h_x^n} K(\frac{x}{h})$，$h$ 为带宽，$K(\bullet)$ 为核函数。在实际计算中，为了减少计算量，通常采用高斯核函数。

粒子滤波程序

```
clear;
clc;
% function ParticleEx1

% Particle filter example, adapted from Gordon, Salmond, and Smith paper.

x = 0.1; % 初始状态
Q = 1; % 过程噪声协方差 process noise covariance
R = 1; % 观测噪声协方差 measurement noise covariance
```

```
tf = 50; % 模拟长度 simulation length

N = 100; % 粒子数目 number of particles in the particle filter

xhat = x;
P = 2;
xhatPart = x;

% 粒子滤波初始化 Initialize the particle filter.
for i = 1 : N
    xpart(i) = x + sqrt(P) * randn;
end

xArr = [x];
yArr = [x^2 / 20 + sqrt(R) * randn];
xhatArr = [x];
PArr = [P];
xhatPartArr = [xhatPart];

close all;

for k = 1 : tf
    % 模拟系统 System simulation
    x = 0.5 * x + 25 * x / (1 + x^2) + 8 * cos(1.2*(k-1)) + sqrt(Q) * randn;%状态方程
    y = x^2 / 20 + sqrt(R) * randn;%观测方程
    % EKF 滤波器 Extended Kalman filter
    F = 0.5 + 25 * (1 - xhat^2) / (1 + xhat^2)^2;
    P = F * P * F' + Q;
    H = xhat / 10;
    K = P * H' * (H * P * H' + R)^(-1);
    xhat = 0.5 * xhat + 25 * xhat / (1 + xhat^2) + 8 * cos(1.2*(k-1));%预测
    xhat = xhat + K * (y - xhat^2 / 20);%更新
    P = (1 - K * H) * P;
    % 粒子滤波 Particle filter
    for i = 1 : N
        xpartminus(i) = 0.5 * xpart(i) + 25 * xpart(i) / (1 + xpart(i)^2) + 8 * cos(1.2*(k-1))
+ sqrt(Q) * randn;
        ypart = xpartminus(i)^2 / 20;
        vhat = y - ypart;%观测和预测的差
        q(i) = (1 / sqrt(R) / sqrt(2*pi)) * exp(-vhat^2 / 2 / R);
    end
    % 归一化先验概率 Normalize the likelihood of each a priori estimate.
    qsum = sum(q);
    for i = 1 : N
```

```
        q(i) = q(i) / qsum;%归一化权重
    end
    % 重采样 Resample.
    for i = 1 : N
        u = rand; % uniform random number between 0 and 1
        qtempsum = 0;
        for j = 1 : N
            qtempsum = qtempsum + q(j);
            if qtempsum >= u
                xpart(i) = xpartminus(j);
                break;
            end
        end
    end
    % 粒子滤波的估计是粒子的平均 The particle filter estimate is the mean of the particles.
    xhatPart = mean(xpart);
    % 画出特定点的 pdf Plot the estimated pdf's at a specific time.
    if k == 20
        % Particle filter pdf
        pdf = zeros(81,1);
        for m = -40 : 40
            for i = 1 : N
                if (m <= xpart(i)) && (xpart(i) < m+1)
                    pdf(m+41) = pdf(m+41) + 1;
                end
            end
        end
        figure;
        m = -40 : 40;
        plot(m, pdf / N, 'r');
        hold;
        title('Estimated pdf at k=20');
        disp(['min, max xpart(i) at k = 20: ', num2str(min(xpart)), ', ', num2str(max(xpart))]);
        % Kalman filter pdf
        pdf = (1 / sqrt(P) / sqrt(2*pi)) .* exp(-(m - xhat).^2 / 2 / P);
        plot(m, pdf, 'b');
        legend('Particle filter', 'Kalman filter');
    end
    % 保存数据 Save data in arrays for later plotting
    xArr = [xArr x];
    yArr = [yArr y];
    xhatArr = [xhatArr xhat];
    PArr = [PArr P];
    xhatPartArr = [xhatPartArr xhatPart];
```

```
end

t = 0 : tf;

%figure;
%plot(t, xArr);
%ylabel('true state');

figure;
plot(t, xArr, 'b.', t, xhatArr, 'k-', t, xhatArr-2*sqrt(PArr), 'r:', t, xhatArr+2*sqrt(PArr), 'r:');
axis([0 tf -40 40]);
set(gca,'FontSize',12); set(gcf,'Color','White');
xlabel('time step'); ylabel('state');
legend('True state', 'EKF estimate', '95% confidence region');

figure;
plot(t, xArr, 'b.', t, xhatPartArr, 'k-');
set(gca,'FontSize',12); set(gcf,'Color','White');
xlabel('time step'); ylabel('state');
legend('True state', 'Particle filter estimate');

xhatRMS = sqrt((norm(xArr - xhatArr))^2 / tf);
xhatPartRMS = sqrt((norm(xArr - xhatPartArr))^2 / tf);
disp(['Kalman filter RMS error = ', num2str(xhatRMS)]);
disp(['Particle filter RMS error = ', num2str(xhatPartRMS)]);
```

粒子滤波的 matlab 实现

粒子滤波是以贝叶斯推理和重要性采样为基本框架的。因此，想要掌握粒子滤波，对于上述两个基本内容必须有一个初步的了解。贝叶斯公式非常 perfect，但是在实际问题中，由于变量维数很高，被积函数很难积分，常常会给粒子滤波带来很大的麻烦。为了克服这个问题，它引入了重要性采样。即先设计一个重要性密度，根据重要性密度与实际分布之间的关系，给采样得到的粒子分配权重。再利用时变贝叶斯公式，给出粒子权重的更新公式及重要性密度的演变形式。在实际问题中，由于直接从重要性密度中采样非常困难，因此做出了妥协，重要性密度选为状态转移分布，随之可得权值更新遵循的规律与量测方程有关。

粒子滤波算法源于 Monte carlo 思想，即以某事件出现的频率来指代该事件的概率。因此在滤波过程中，需要用到概率如 P(x)的地方，一概对变量 x 采样，以大量采样及其相应的权值来近似表示 P(x)。因此，采用此思想，在滤波过程中粒子滤波可以处理任意形式的概率，而不像 Kalman 滤波只能处理线性高斯分布的概率问题。粒子滤波的一大优势也在

于此。

下来看看对任意如下的状态方程：

$$x(t)=f(x(t-1),u(t),w(t))$$

$$y(t)=h(x(t),e(t))$$

其中的 x(t)为 t 时刻状态，u(t)为控制量，w(t)和 e(t)分别为状态噪声和观测噪声。前一个方程描述是状态转移，后一个是观测方程。对于这么一个问题粒子滤波怎么来从观测 y(t)，和 x(t-1),u(t) 滤出真实状态 x(t)呢？

预测阶段：粒子滤波首先根据 x(t-1) 的概率分布生成大量的采样，这些采样就称之为粒子。那么这些采样在状态空间中的分布实际上就是 x(t-1) 的概率分布了。好，接下来依据状态转移方程加上控制量可以对每一粒子得到一个预测粒子。

校正阶段：观测值 y 到达后，利用观测方程即条件概率 $P(y|x^i)$,对所有的粒子进行评价，直白的说，这个条件概率代表了假设真实状态 x(t)取第 i 个粒子 x^i 时获得观测 y 的概率。令这个条件概率为第 i 个粒子的权重。如此这般下来，对所有粒子都进行这么一个评价，那么越有可能获得观测 y 的粒子，当然获得的权重越高。

重采样算法：去除低权值的粒子，复制高权值的粒子。所得到的当然就是我们说需要的真实状态 x(t)了，而这些重采样后的粒子，就代表了真实状态的概率分布了。下一轮滤波，再将重采样过后的粒子集输入到状态转移方程中，直接就能够获得预测粒子了。

初始状态的问题： 由于开始对 x(0)一无所知，所有我们可以认为 x(0)在全状态空间内平均分布。于是初始的采样就平均分布在整个状态空间中。然后将所有采样输入状态转移方程，得到预测粒子。然后再评价下所有预测粒子的权重，当然我们在整个状态空间中只有部分粒子能够获的高权值。最后进行重采样，去除低权值的，将下一轮滤波的考虑重点缩小到了高权值粒子附近。

具体过程：

1）初始化阶段-提取跟踪目标特征

该阶段要人工指定跟踪目标，程序计算跟踪目标的特征，比如可以采用目标的颜色特征。具体到 Rob Hess 的代码，开始时需要人工用鼠标拖动出一个跟踪区域，然后程序自动计算该区域色调(Hue)空间的直方图，即为目标的特征。直方图可以用一个向量来表示，所以目标特征就是一个 N*1 的向量 V。

2）搜索阶段-放狗

好，我们已经掌握了目标的特征，下面放出很多条狗，去搜索目标对象，这里的狗就是粒子 particle。狗有很多种放法。比如，a)均匀的放：即在整个图像平面均匀的撒粒子(uniform distribution)；b)在上一帧得到的目标附近按照高斯分布来放，可以理解成，靠近目标的地方多放，远离目标的地方少放。Rob Hess 的代码用的是后一种方法。狗放出去后，每条狗怎么搜索目标呢？就是按照初始化阶段得到的目标特征(色调直方图，向量 V)。每

条狗计算它所处的位置处图像的颜色特征，得到一个色调直方图，向量 Vi，计算该直方图与目标直方图的相似性。相似性有多种度量，最简单的一种是计算 sum(abs(Vi-V)).每条狗算出相似度后再做一次归一化，使得所有的狗得到的相似度加起来等于 1.

3）决策阶段

我们放出去的一条条聪明的狗向我们发回报告，“一号狗处图像与目标的相似度是 0.3”，“二号狗处图像与目标的相似度是 0.02”，“三号狗处图像与目标的相似度是 0.0003”，“N 号狗处图像与目标的相似度是 0.013”...那么目标究竟最可能在哪里呢？我们做次加权平均吧。设 N 号狗的图像像素坐标是(Xn,Yn),它报告的相似度是 Wn,于是目标最可能的像素坐标 X = sum(Xn*Wn),Y = sum(Yn*Wn).

4）重采样阶段 Resampling

既然我们是在做目标跟踪，一般说来，目标是跑来跑去乱动的。在新的一帧图像里，目标可能在哪里呢？还是让我们放狗搜索吧。但现在应该怎样放狗呢？让我们重温下狗狗们的报告吧。“一号狗处图像与目标的相似度是 0.3”，“二号狗处图像与目标的相似度是 0.02”，“三号狗处图像与目标的相似度是 0.0003”，“N 号狗处图像与目标的相似度是 0.013”...综合所有狗的报告，一号狗处的相似度最高，三号狗处的相似度最低，于是我们要重新分布警力，正所谓好钢用在刀刃上，我们在相似度最高的狗那里放更多条狗，在相似度最低的狗那里少放狗，甚至把原来那条狗也撤回来。这就是 Sampling Importance Resampling，根据重要性重采样(更具重要性重新放狗)。

(2)->(3)->(4)->(2)如是反复循环，即完成了目标的动态跟踪。

程序：

```
clc;
clear all;
close all;
x = 0; %初始值
R = 1;
Q = 1;
tf = 100; %跟踪时长
N = 100; %粒子个数
P = 2;
xhatPart = x;
for i = 1 : N
    xpart(i) = x + sqrt(P) * randn;%初始状态服从 0 均值，方差为 sqrt(P)的高斯分布
end
xArr = [x];
yArr = [x^2 / 20 + sqrt(R) * randn];
xhatArr = [x];
PArr = [P];
xhatPartArr = [xhatPart];
```

```
for k = 1 : tf%g 跟踪时长
    x = 0.5 * x + 25 * x / (1 + x^2) + 8 * cos(1.2*(k-1)) + sqrt(Q) * randn;
     %k 时刻真实值
     y = x^2 / 20 + sqrt(R) * randn; %k 时刻观测值
    for i = 1 : N%粒子个数
        xpartminus(i) = 0.5 * xpart(i) + 25 * xpart(i) / (1 + xpart(i)^2) ...
        + 8 * cos(1.2*(k-1)) + sqrt(Q) * randn;%采样获得 N 个粒子
        ypart = xpartminus(i)^2 / 20;%每个粒子对应观测值
        vhat = y - ypart;%与真实观测之间的似然
        q(i) = (1 / sqrt(R) / sqrt(2*pi)) * exp(-vhat^2 / 2 / R);
    %每个粒子的似然即相似度   正太分布的概率密度
    end
qsum = sum(q);
for i = 1 : N
        q(i) = q(i) / qsum;%权值归一化
end
    for i = 1 : N %根据权值重新采样
        u = rand;
        qtempsum = 0;
        for j = 1 : N
            qtempsum = qtempsum + q(j);
            if qtempsum >= u
                xpart(i) = xpartminus(j);
                break;
            end
         end
    end
xhatPart = mean(xpart);
%最后的状态估计值即为 N 个粒子的平均值，这里经过重新采样后各个粒子的权值相
同
xArr = [xArr x]; %真实值
yArr = [yArr y]; %观测值
% xhatArr = [xhatArr xhat];?
PArr = [PArr P];%误差
xhatPartArr = [xhatPartArr xhatPart];%预测值
end
t = 0 : tf;
figure;
plot(t, xArr, 'b-.', t, xhatPartArr, 'k-');
legend('Real Value','Estimated Value');
set(gca,'FontSize',10);
xlabel('time step');
ylabel('state');
title('Particle filter')
```

```
xhatRMS = sqrt((norm(xArr - xhatArr))^2 / tf);
xhatPartRMS = sqrt((norm(xArr - xhatPartArr))^2 / tf);
figure;
plot(t,abs(xArr-xhatPartArr),'b');
title('The error of PF')
```