# Introduction to Particle Filtering

Dr. **Yuan-Li Cai**

Spring • 2024

# 0. Outline

# 1. 引言

前面介绍的扩展卡尔曼滤波方法（EKF）是非线性系统状态估计中最常用的方法，其本质是基于线性化来传递系统的均值和协方差。对于非线性十分严重的系统，实际应用 EKF 变得非常困难，有时给出的状态估计可能十分不可靠。而 UKF 可以减小线性化带来的误差，因此一般情况下可以获得比 EKF 要高的估计精度。

对比 EKF 和 UKF，我们不难发现 EKF 估计非线性系统均值的精度是 1 阶的，而 UKF 可以到达 2 阶以上。然而，当系统的状态方程或量测方程

的非线性非常严重时，两者都避免不了滤波发散问题，只不过 UKF 推迟或延缓了滤波的发散。

这里我们将介绍粒子滤波（partical filter, PF）方法，可以有效地克服 EKF 或 UKF 可能遇到的滤波发散问题。粒子滤波的最初研究可以追溯到 20 世纪 40 年代，当时 Metropolis 和 Norbert Wiener 就提出了类似的设想。但是，实现粒子滤波离不开强大的计算资源，即使是计算机技术高度发展的今天，粒子滤波所需要的计算量仍然是限制其广泛应用的瓶颈。在有的文献中，粒子滤波又称为序贯重要性采样、蒙特-卡洛滤波等。

粒子滤波是一种“暴力”滤波方法，它以计算代价换取滤波性能。对于有些问题而言，这是值得或必须的。而对另外一些问题而言，如果计算资源受到较大的限制，那么粒子滤波则是不可取的。简言之，世上没有免费的

午餐，要不要采用粒子滤波，或者如何采用粒子滤波，要在计算资源与估计性能之间寻找合理的折衷，完全取决于问题本身。

由于 PF 属于随机滤波算法，因此我们首先简要回顾一下贝叶斯递推滤波算法 (第2节)。PF 的本质是蒙特-卡洛仿真，我们将在第3节、第4节分别介绍其中的完备采样、重要性采样及重采样等技术。然后，在第5节介绍 PF 的基本原理和算法。在基本粒子滤波算法的基础上，已经发展起来许多改进算法，我们将在第6节简要讨论两个重要的改进算法，分别是正则化粒子滤波和组合粒子滤波。此外，我们还给出了几个算例（附有了 MATLAB 代码），由此可以帮助大家对基本概念的理解和掌握。

# 2. 贝叶斯递推滤波

考虑如下一般的非线性系统:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_k(\boldsymbol{x}_k, \boldsymbol{w}_k) \tag{1}$$

$$\boldsymbol{y}_k = \boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{v}_k) \tag{2}$$

其中，$\boldsymbol{x}_k \in \mathcal{R}^n$，$\boldsymbol{y}_k \in \mathcal{R}^m$; $\{\boldsymbol{w}_k\}$ 和 $\{\boldsymbol{w}_k\}$ 是相互独立的纯随机序列; $\boldsymbol{x}_0 \sim p_{\boldsymbol{x}_0}(\boldsymbol{x}_0) = p(\boldsymbol{x}_0) = p(\boldsymbol{x}_0|\boldsymbol{y}_0)$，而且与 $\{\boldsymbol{w}_k\}$ 和 $\{\boldsymbol{v}_k\}$ 相互独立。

记 $\boldsymbol{Y}_k = (\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_k)$, 那么

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{Y}_k) = \int p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k) p(\boldsymbol{x}_k|\boldsymbol{Y}_k) \mathrm{d}\boldsymbol{x}_k \tag{3}$$

$$p(\boldsymbol{y}_{k+1}|\boldsymbol{Y}_k) = \int p(\boldsymbol{y}_{k+1}|\boldsymbol{x}_{k+1}) p(\boldsymbol{x}_{k+1}|\boldsymbol{Y}_k) \mathrm{d}\boldsymbol{x}_{k+1} \tag{4}$$

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{Y}_{k+1}) = \frac{p(\boldsymbol{y}_{k+1}|\boldsymbol{x}_{k+1}) p(\boldsymbol{x}_{k+1}|\boldsymbol{Y}_k)}{p(\boldsymbol{y}_{k+1}|\boldsymbol{Y}_k)} \tag{5}$$

以上三式便构成了递推求解 $p(\boldsymbol{x}_k|\boldsymbol{Y}_k)$ 的核心公式，称为贝叶斯递推滤波算法。一般情况下，通常无法获得解析解。

在 Monte Carlo (MC) 仿真中，以若干足够多的离散样本（称为粒子）对期望的概率密度进行近似，即

$$p(\boldsymbol{x}_k|\boldsymbol{Y}_k) \approx \hat{p}(\boldsymbol{x}_k|\boldsymbol{Y}_k) = \sum_{i=1}^{N} w_k^{(i)} \delta(\boldsymbol{x} - \boldsymbol{x}_k^{(i)}) \tag{6}$$

其中，$w_k^{(i)} \geq 0$，而且 $\sum_{i=1}^{N} w_k^{(i)} = 1$。

**问题：**如果生成粒子 $\{\boldsymbol{x}_k^{(i)}\}_{i=1,2,\cdots,N}$ 并确定 $\{w_k^{(i)}\}_{i=1,2,\cdots,N}$ ？

# 3. 完备采样（perfect sampling）

对于目标密度函数 $p(\boldsymbol{x})$，设 $\{\boldsymbol{x}^{(i)}, i = 1, 2, \cdots, N\}$ 是根据 $p(\boldsymbol{x})$ 采样到的独立同分布粒子，那么

$$p(\boldsymbol{x}) \approx \hat{p}(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} \delta(\boldsymbol{x} - \boldsymbol{x}^{(i)}) \tag{7}$$

例如:

$$\bar{\boldsymbol{x}} = \int \boldsymbol{x} p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \approx \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}^{(i)}$$

$$\mathrm{var}(\boldsymbol{x}) = \int (\boldsymbol{x} - \bar{\boldsymbol{x}})(\boldsymbol{x} - \bar{\boldsymbol{x}})^{\mathrm{T}} p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \approx \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}})(\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}})^{\mathrm{T}}$$

更加一般地, 对于比较任意的函数 $\boldsymbol{g}(\cdot)$ 有

$$E\boldsymbol{g}(\boldsymbol{x}) \approx \hat{E}_N \boldsymbol{g}(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{g}(\boldsymbol{x}^{(i)}) \tag{8}$$

理论上

$$\lim_{N \to +\infty} \hat{E}_N \boldsymbol{g}(\boldsymbol{x}) \xrightarrow{a.s.} E\boldsymbol{g}(\boldsymbol{x}) \tag{9}$$

对于非线性滤波问题，几乎不可能直接从 $p(\boldsymbol{x}_k|\boldsymbol{Y}_k)$ 进行采样，需要下述重要性采样方法。

# 4. 重要性采样与重采样

## 4.1 重要性采样

通常，目标密度函数 $p(\boldsymbol{x})$ 非常复杂，不容易直接产生符合密度函数 $p(\boldsymbol{x})$ 的粒子。设 $q(\boldsymbol{x})$ 是较之 $p(\boldsymbol{x})$ 容易实现采样的概率分布密度函数，在支撑覆盖条件下，如果 $\{\boldsymbol{x}^{(i)}, i = 1, 2, \cdots, N\} \sim q(\boldsymbol{x})$，那么粒子 $\boldsymbol{x}^{(i)}$ 属于 $p(\boldsymbol{x})$ 的概率为 $w^{(i)}$，称为**接受概率**，有

$$p(\boldsymbol{x}^{(i)}) \propto w^{(i)} q(\boldsymbol{x}^{(i)}) \tag{10}$$

上式可以简单论证如下。对于任意的函数 $g(\boldsymbol{x})$，有

$$\begin{aligned} Eg(\boldsymbol{x}) &= \int g(\boldsymbol{x})p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\ &= \int [g(\boldsymbol{x})\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}]q(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \\ &\approx \frac{1}{N}\sum_{i=1}^{N} g(\boldsymbol{x}^{(i)})\frac{p(\boldsymbol{x}^{(i)})}{q(\boldsymbol{x}^{(i)})} \\ &= \sum_{i=1}^{N} w^{(i)} g(\boldsymbol{x}^{(i)}) \end{aligned}$$

上式定义了 $w^{(i)}$，并验证了 (10) 式。换言之，我们有

$$p(\boldsymbol{x}) \approx \sum_{i=1}^{N} w^{(i)}\delta(\boldsymbol{x} - \boldsymbol{x}^{(i)}) \tag{11}$$

上式说明了 $w^{(i)}$ 为**接受概率**的含义，同时暗喻着 $\sum_{i=1}^{N} w^{(i)} = 1$。

[例] 定积分的 MC 求解。

## 4.2 重采样

重采样（resampling）是指根据（11）式进行重新采样，获取若干个近似服从 $p(\boldsymbol{x})$ 分布的粒子。有许多重采样方法，它决定了粒子滤波算法的计算复杂度。

对于目标密度函数的估计

$$\hat{p}(\boldsymbol{x}) = \sum_{i=1}^{N} \bar{w}^{(i)} \delta(\boldsymbol{x} - \boldsymbol{x}^{(i)}) \tag{12}$$

以 $\bar{w}^{(i)}$ 为接受概率重新产生 $N$ 个粒子，即（离散概率密度）

$$\Pr(\hat{\boldsymbol{x}}^{(i)} = \boldsymbol{x}^{(j)}) = \bar{w}^{(j)}, i = 1, 2, \cdots, N \tag{13}$$

重采样后，目标密度函数近似为

$$\tilde{p}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\boldsymbol{x} - \hat{\boldsymbol{x}}^{(i)}) \tag{14}$$

显然，重采样可能带来粒子贫化的问题。

常用的重采样方法:

- 简单随机采样（轮盘赌算法）

  取 $i = 1, 2, \cdots, N$，执行如下两步:

  a) 生成随机数 $r \sim U[0, 1]$（均匀分布）;

  b) 逐个累加 $w^{(i)}$，直到大于 $r$，即 $\sum_{i=1}^{j} w^{(i)} \geq r$ 且 $\sum_{i=1}^{j-1} < r$，置旧粒子 $\boldsymbol{x}^{(j)}$ 为重采样粒子 $\hat{\boldsymbol{x}}^{(j)}$。

♦ 系统采样法（systematic sampling）

a) 根据下式生成 $N$ 个随机数:

$$u_i = \frac{(i-1)+r}{N}, r \sim U[0,1]$$

b) 如果 $\sum_{j=1}^{m-1} \bar{w}^{(j)} < u_i \leq \sum_{j=1}^{m} \bar{w}^{(j)}$，直接拷贝 $m$ 个粒子 $\{\boldsymbol{x}^{(i)}\}$ 为重采样粒子 $\{\hat{\boldsymbol{x}}^{(i)}\}$。

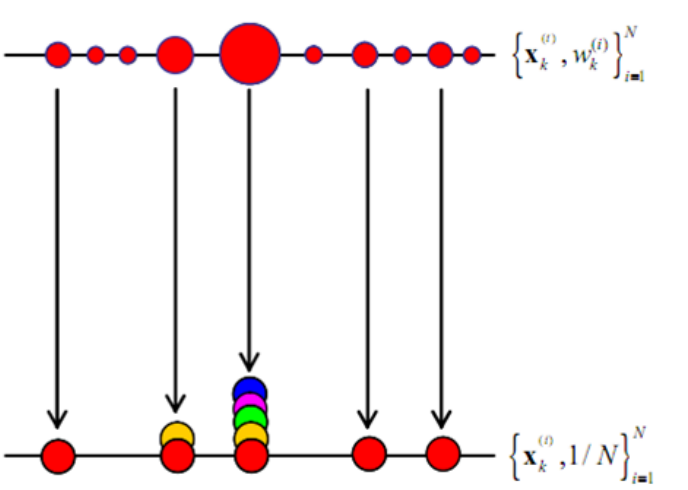


Figure 1: 重采样示意图

# 5. 粒子滤波算法

## 5.1 基本思想

粒子滤波是 Beyes 递推滤波算法的数值实现。首先根据系统状态的初始概率密度函数 $p(\boldsymbol{x}_0)$（假设是已知的）产生 $N$ 个状态矢量（称为粒子），即 $\boldsymbol{x}_{0|0}^{(i)} \sim p(\boldsymbol{x}_0)(i = 1, 2, \cdots, N)$。

在每一时刻 $k = 1, 2, \cdots$，按照状态方程传播粒子，即

$$\boldsymbol{x}_{k|k-1}^{(i)} = \boldsymbol{f}_k(\boldsymbol{x}_{k-1|k-1}^{(i)}, \boldsymbol{w}_{k-1}^{(i)}), i = 1, 2, \cdots, N \tag{15}$$

式中，$\boldsymbol{w}_{k-1}^{(i)}$ 是根据过程噪声统计特性生成的噪声矢量。上式表明，我们获得了服从 $p(\boldsymbol{x}_{k|k-1}) = p(\boldsymbol{x}_k|\boldsymbol{y}_{k-1})$ 分布的若干粒子，由此可以计算一步预测的均值和方差等。换句话说，只要 $N$ 足够大，由完全采样可知

$$p(\boldsymbol{x}_{k|k-1}) \approx \hat{p}(\boldsymbol{x}_{k|k-1}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\boldsymbol{x}_{k|k-1} - \boldsymbol{x}_{k|k-1}^{(i)}) \tag{16}$$

当获得新的量测值 $\boldsymbol{y}_k$ 后，$p(\boldsymbol{x}_{k|k})$ 的近似估计可以表达为

$$\hat{p}(\boldsymbol{x}_{k|k}) = \sum_{i=1}^{N} w^{(i)}\delta(\boldsymbol{x}_{k|k} - \boldsymbol{x}_{k|k-1}^{(i)}) \tag{17}$$

其中 $w^{(i)} = \Pr(\boldsymbol{x}_k = \boldsymbol{x}_{k|k-1}^{(i)}|\boldsymbol{y}_k)$，即对 (16) 中 $1/N$ 的修正。

由于

$$\Pr(\boldsymbol{x}_k = \boldsymbol{x}_{k|k-1}^{(i)}|\boldsymbol{y}_k) = \frac{\Pr(\boldsymbol{y}_k|\boldsymbol{x}_k = \boldsymbol{x}_{k|k-1}^{(i)})\Pr(\boldsymbol{x}_k = \boldsymbol{x}_{k|k-1}^{(i)})}{\Pr(\boldsymbol{y}_k)} \tag{18}$$

因此

$$w^{(i)} \propto \Pr(\boldsymbol{y}_k|\boldsymbol{x}_k = \boldsymbol{x}_{k|k-1}^{(i)}) \tag{19}$$

总结上面的讨论，根据量测方程计算每个粒子 $\boldsymbol{x}_{k|k-1}^{(i)}$ 的条件似然值，并归一化，即

$$w_k^{(i)} = p(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(i)}) \tag{20}$$

$$\bar{w}_k^{(i)} = \frac{w_k^{(i)}}{\sum_{i=1}^{N} w_k^{(i)}}, i = 1, 2, \cdots, N \tag{21}$$

那么

$$\hat{p}(\boldsymbol{x}_{k|k}) = \sum_{i=1}^{N} \bar{w}_k^{(i)} \delta(\boldsymbol{x}_{k|k} - \boldsymbol{x}_{k|k-1}^{(i)}) \tag{22}$$

为了下一步递推计算，需要进行重采样。重采样可以认为按下式获得 $\{\boldsymbol{x}_{k|k}^{(i)}\}_{i=1}^{N}$.

$$\Pr(\boldsymbol{x}_{k|k}^{(i)} = \boldsymbol{x}_{k|k-1}^{(j)}) = \bar{w}_k^{(j)}, \quad i = 1, 2, \cdots, N \tag{23}$$

重采样后粒子权重均为 $1/N$。

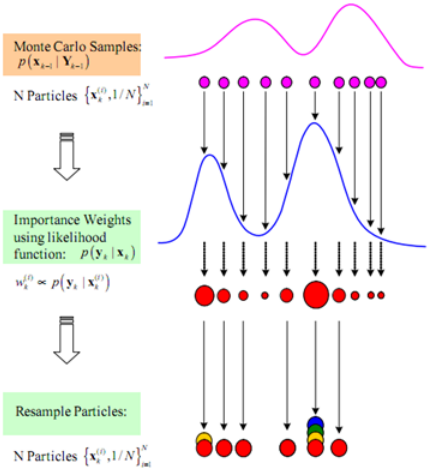


Figure 2: 粒子滤波示意图

## 5.2 PF 算法 1

(1) 初始化：$\{\boldsymbol{x}_{0|0}^{(i)}\}_{i=1}^{N} \sim p_{\boldsymbol{x}_0}(\boldsymbol{x}_0)$，置 $k=1$；

(2) 预测：$\boldsymbol{x}_{k|k-1}^{(i)} \sim p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1|k-1}^{(i)})$，$i=1,2,\cdots,N$；

(3) 滤波：

(a) 计算重要性权重：$w_k^{(i)} = p(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(i)})$，$i=1,2,\cdots,N$；

(b) 归 1 化得到接受概率：$\bar{w}_k^{(i)} = \dfrac{w_k^{(i)}}{\sum\limits_{i=1}^{N} w_k^{(i)}}$，$i=1,2,\cdots,N$；

(c) 输出（根据需要）：

$$\hat{\boldsymbol{x}}_{k|k} = \sum_{i=1}^{N} \bar{w}_k^{(i)} \boldsymbol{x}_{k|k-1}^{(i)},$$

$$\boldsymbol{P}_{k|k} = \sum_{i=1}^{N} \bar{w}_k^{(i)} (\boldsymbol{x}_{k|k-1}^{(i)} - \hat{\boldsymbol{x}}_{k|k})(\boldsymbol{x}_{k|k-1}^{(i)} - \hat{\boldsymbol{x}}_{k|k})^{\mathrm{T}}.$$

(d) 根据 $\Pr(\boldsymbol{x}_{k|k}^{(i)} = \boldsymbol{x}_{k|k-1}^{(j)}) = \bar{w}_k^{(j)}$ 重采样 $N$ 个新的粒子 $\boldsymbol{x}_{k|k}^{(i)}$;

(4) 置 $k := k + 1$，转（2）循环迭代。

## 5.3 PF 算法 2

(1) 初始化：$\{\boldsymbol{x}_{0|-1}^{(i)}\}_{i=1}^{N} \sim p_{\boldsymbol{x}_0}(\boldsymbol{x}_0)$，置 $k = 0$;

(2) 滤波：

(a) 计算重要性权重：$w_k^{(i)} = p(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(i)})$，$i = 1, 2, \cdots, N$；

(b) 归 1 化得到接受概率：$\bar{w}_k^{(i)} = \frac{w_k^{(i)}}{\sum_{i=1}^{N} w_k^{(i)}}$，$i = 1, 2, \cdots, N$；

(c) 输出（根据需要）：

$$\hat{\boldsymbol{x}}_{k|k} = \sum_{i=1}^{N} \bar{w}_k^{(i)} \boldsymbol{x}_{k|k-1}^{(i)}$$

$$\boldsymbol{P}_{k|k} = \sum_{i=1}^{N} \bar{w}_k^{(i)} (\boldsymbol{x}_{k|k-1}^{(i)} - \hat{\boldsymbol{x}}_{k|k})(\boldsymbol{x}_{k|k-1}^{(i)} - \hat{\boldsymbol{x}}_{k|k})^{\mathrm{T}}$$

(3) 重采样：根据 $\Pr(\boldsymbol{x}_{k|k}^{(i)} = \boldsymbol{x}_{k|k-1}^{(j)}) = \bar{w}_k^{(j)}$ 重采样 $N$ 个新的粒子 $\boldsymbol{x}_{k|k}^{(i)}$；

(4) 预测：$\boldsymbol{x}_{k+1|k}^{(i)} \sim p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_{k|k}^{(i)})$，$i = 1, 2, \cdots, N$；

(5) 置 $k := k + 1$，转（2）循环迭代。

**[说明]**

- 在上述粒子滤波算法中，预测 $\boldsymbol{x}_{k+1|k}^{(i)} \sim p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_{k|k}^{(i)})$ 是指计算 $\boldsymbol{x}_{k|k-1}^{(i)} = \boldsymbol{f}_k(\boldsymbol{x}_{k-1|k-1}^{(i)}, \boldsymbol{w}_{k-1}^{(i)})$，其中 $\boldsymbol{w}_{k-1}^{(i)} \sim p(\boldsymbol{w}_{k-1}), i = 1, 2, \cdots, N$;
- 算法 1 与算法 2 之间的差异在于对初始粒子的假设不同，算法 2 比算法 1 提前 1 步进行量测修正;
- 上述滤波算法的输出均采用了最常规的验后概率均值，也可以采用验后概率对应的中值、模值等。

## 5.4 改进算法及相关研究

1. 引入粒子退化度量，避免每一步都进行重采样;
2. 重要性函数选取：辅助粒子滤波（APF）
3. 目标密度函数近似：高斯粒子滤波（GPF，GSPF 等）
4. 系统结构信息：边缘化粒子滤波（Marginalized Particle Filter）
5. 不同的重采样算法;
6. 嵌入 EKF/UKF 等;
7. 自适应技术;

8. 粒子平滑.

## 5.5 实现问题与 MATLAB 代码

### 5.5.1 重要性权重计算

如果量测噪声是加性噪声，即 $\boldsymbol{y}_k = \boldsymbol{h}_k(\boldsymbol{x}_k) + \boldsymbol{v}_k$，那么

$$w_k^{(i)} = p(\boldsymbol{y}_k|\boldsymbol{x}_{k|k-1}^{(i)}) = p_{\boldsymbol{v}_k}(\boldsymbol{y}_k - h(\boldsymbol{x}_{k|k-1}^{(i)})) \tag{24}$$

### 5.5.2 预测粒子计算

如果过程噪声是加性的，那么

$$\boldsymbol{x}_{k|k-1}^{(i)} = \boldsymbol{f}_k(\boldsymbol{x}_{k-1|k-1}^{(i)}) + \boldsymbol{w}_{k-1}^{(i)} \tag{25}$$

其中噪声粒子 $\boldsymbol{w}_{k-1}^{(i)} \sim p_{\boldsymbol{w}_{k-1}}(\boldsymbol{w}_{k-1})$。

### 5.5.3 MATLAB code

```
%PF
function [xhat] = PF(f,h,pd\bm{f}_v,Q,P0,M,y)
n = size(P0,2);
x = sqrtm(P0)*randn(n,M); % 1. Initialize particles
tf = size(y,2);

% Algrithm 2
for t = 1:tf
    e = repmat(y(t),1,M) - h(x);
```

```
    % 2. Calculate weights
    w = feval(pd\bm{f}_v,e); % The likelihood function
    w = w/sum(w); % Normalize importance weights
    xhat(t) = sum(repmat(w,n,1).*x,2);
    x = resampling(x, w);% 3. Resampling
    x = feval(f,x,t)+sqrtm(Q)*randn(n,M);
    % 4. Time update
end

% Algrithm 1
% for t = 1:tf
```

```
%       x = feval(f,x,t)+sqrtm(Q)*randn(n,M);
% 2. Time update
%       e = repmat(y(t),1,M) - h(x);
% 3. Calculate weights
%       w = feval(pd\bm{f}_v,e); % The likelihood function
%       w = w/sum(w); % Normalize importance weights
%       xhat(t) = sum(repmat(w,n,1).*x,2);
%       x = resampling(x, w); % 4. Resampling
% end


function [x] = resampling(x,w)
```

```
wc = cumsum(w); M=length(w);
u = ([0:M-1]+rand(1))/M;
ind = zeros(1,M); k = 1;
for j = 1:M
    while (wc(k)<u(j))
        k = k + 1;
    end
    ind(j) = k;
end
x=x(:, ind);
```

说明:

(1) f，h，pd\bm{f}_v: 内键（inline）函数或 M 文件，分别为状态方程及量测方程非线性函数以及似然函数（量测噪声密度函数）;

(2) Q，P0: 过程噪声与初始状态协方差矩阵;

(3) M，y: 粒子数与量测值;

(4) Xhat: 状态滤波（输出）。

### 5.5.4 仿真算例

[例 1] 许多论文采样的算例（EKF 发散）：

$$x_{k+1} = \frac{x_k}{2} + \frac{25x_k}{1+x_k^2} + 8\cos(1.2k) + w_k$$

$$y_k = \frac{x_k^2}{20} + v_k$$

其中，$x_0 \sim \mathcal{N}(0,5)$；$w_k \sim \mathcal{N}(0,10)$，$v_k \sim \mathcal{N}(0,1)$，二者均为白噪声，相互独立。

实现代码

```
clear all;
M = 1000; % Number of particles
P0 = 5; % Initial noise covariance
Q = 10; % Process noise covariance
R = 1; % Measurement noise covariance
tf = 100; % Final time

pd\bm{f}_v = inline('1/(2*pi*1)^(1/2)*exp(-(x.^2)/(2*1))');
f = inline('x./2+25*x./(1+x.^2)+8*cos(1.2*t)','x','t');
h = inline('(x.^2)/20');
```

```
x(1) = sqrtm(P0)*randn(1); % Initial state value
y(1) = feval(h,x(1)) + sqrtm(R)*randn(1);

for t = 2:tf % Simulate the system
x(t) = feval(f,x(t-1),t-1) + sqrtm(Q)*randn(1);
y(t) = feval(h,x(t)) + sqrtm(R)*randn(1);
end

xTrue = x;
xhat = PF(f,h,pd\bm{f}_v,Q,P0,M,y);
```

```
plot(1:tf,xhat,'b--',1:tf,xTrue,'r');
xlabel('Time');
legend('状态估值','状态真值');
title('Particle Filter Simulation by Yuan-Li Cai');
grid on;

rms = sum((xTrue - xhat).^2);
rms = sqrt(rms/tf)
```

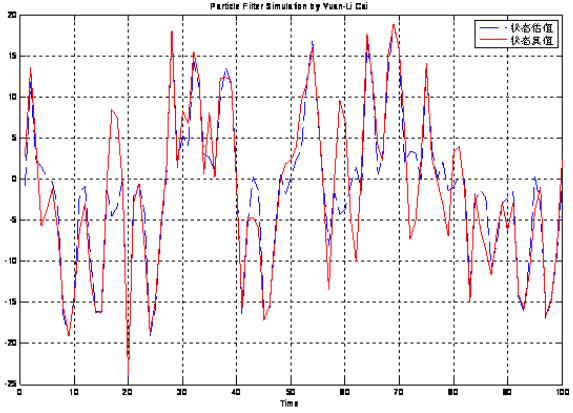


Figure 3: 粒子滤波仿真 (a)

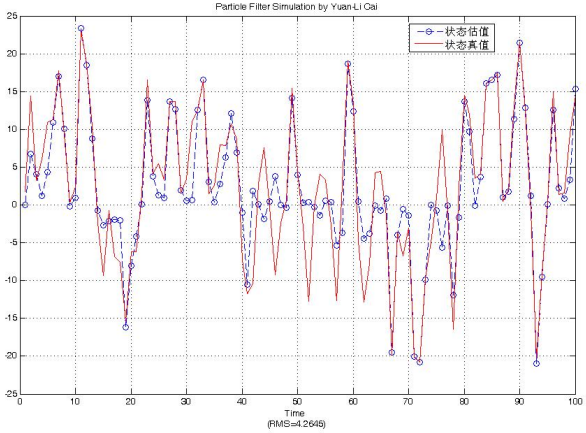


Figure 4: 粒子滤波仿真 (b)

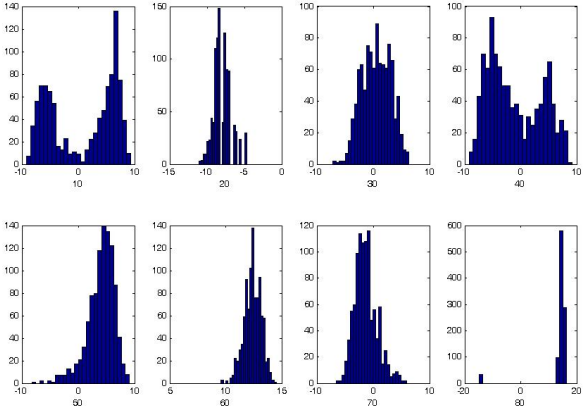


Figure 5: 粒子滤波仿真 (c)

# 6. 改进粒子滤波

## 6.1 正则化粒子滤波

粒子滤波除了计算量大外，主要还存在**粒子退化**和**粒子贫化**问题。粒子退化指滤波过程中，少量粒子的重要性权重变得很大，而若干粒子的重要性权重变得变得很小，甚至趋于 0。通过重采样，如前面给出的粒子滤波算法那样，可以克服粒子退化现象。

但在状态空间的某些区域，如果 $p(\boldsymbol{y}_k|\boldsymbol{x}_k)$ 与 $p(\boldsymbol{x}_k|\boldsymbol{y}_{k-1})$ 不能很好覆盖时，会出现粒子贫化现象。即通过 $p(\boldsymbol{y}_k|\boldsymbol{x}_k)$ 确定的接受概率进行重采样时，进入下一步滤波计算的**不同粒子数目**越来越少，使得粒子的多样性越来越

差。为此，提出了许多改进算法。这里简单介绍正则化粒子滤波的基本思想，这是一种可以有效克服粒子贫化的改进粒子滤波算法。

正则化粒子滤波（Regularied Particle Filtering, RPF）通过修改重采样依据的概率密度函数来完成，即用连续的概率密度函数代替前面 PF 算法中的离散概率密度函数，从而保证重采样后的粒子具有多样性。

回顾前面 PF 算法，重采样本质是依据如下概率密度函数:

$$\hat{p}(\boldsymbol{x}_{k|k}) = \sum_{i=1}^{N} \bar{w}_k^{(i)} \delta(\boldsymbol{x}_{k|k} - \boldsymbol{x}_{k|k-1})$$

RPF 中，将上式修改为如下连续函数:

$$\hat{p}(\boldsymbol{x}_{k|k}) = \sum_{i=1}^{N} \bar{w}_k^{(i)} K_h(\boldsymbol{x}_{k|k} - \boldsymbol{x}_{k|k-1}) \tag{26}$$

其中

$$K_h(\boldsymbol{x}) = h^n K(\boldsymbol{x}/h) \tag{27}$$

$h$ 称为带宽，是一个设计参数；$K(\boldsymbol{x})$ 为一个对称的核函数，满足如下条件:

$$\int \boldsymbol{x} K(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = 0 \tag{28}$$

$$\int ||\boldsymbol{x}||_2^2 \mathrm{d}\boldsymbol{x} < \infty \tag{29}$$

可用的核函数很多，例如高斯函数、三角函数、窗口函数等，都是可能的选择。

## 6.2 组合粒子滤波

粒子滤波可以与与其他滤波方法组合起来使用，从而可以改善滤波的效果。例如，可以将 PF 与 EKF 组合，也可以将 PF 与 UKF 组合，前者可以称为 EKPF，后者可以称为 UKPF 或 SPPF。

EKPF 基本原理如下：

1. 对每个粒子进行 EKF：

$$[\hat{\boldsymbol{x}}_{k|k}^{(i)}, \boldsymbol{P}_{k|k}^{(i)}] = \mathrm{EKF}(\hat{\boldsymbol{x}}_{k-1|k-1}^{(i)}, \boldsymbol{P}_{k-1|k-1}^{(i)}), i = 1, 2, \cdots, N$$

2. 计算重采样概率：

$$w_k^{(i)} = p(\boldsymbol{y}_k|\hat{\boldsymbol{x}}_{k|k}^{(i)}), \bar{w}_k^{(i)} = \frac{w_k^{(i)}}{\sum_{i=1}^{N} w_k^{(i)}}$$

3. 重采样:

$$[\hat{\boldsymbol{x}}_{k|k}^{(i)}, \boldsymbol{P}_{k|k}^{(i)}] = \text{Resample}[\hat{\boldsymbol{x}}_{k|k}^{(i)}, \bar{\boldsymbol{w}}_{k}^{(i)}], i = 1, 2, \cdots, N$$

只要将上面的 EKF 换为 UKF 便是 SPPF 的原理，不再赘述。

[例 2] 考虑非线性系统

$$x_{k+1} = 1 + \sin\frac{k\pi}{25} + \frac{1}{2}x_k + w_k$$

$$y_k = \begin{cases} \frac{1}{2}x_k^2 + v_k, & k \le 30 \\ \frac{1}{2}x_k + v_k, & k > 30 \end{cases}$$

其中，$w_k \sim \mathcal{N}(0, 1.e-5)$，$v_k \sim \Gamma(3, 0.5)$，$x_0 \sim \mathcal{N}(1, \frac{3}{4})$。

200; % number of particle

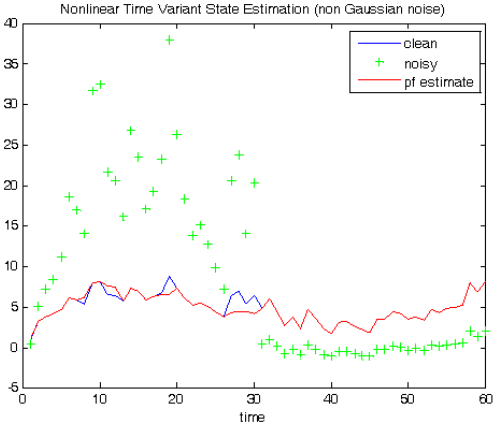


Figure 6: PF 仿真

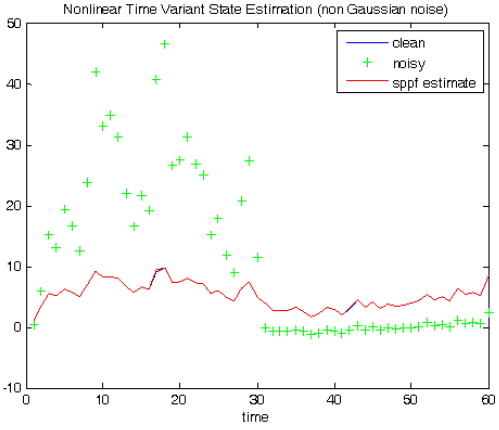


Figure 7: SPPF 仿真

# 6. 参考文献

[1] 蔡远利 (Cai, Y.L.). A Note on Partical Filtering. *Xi'an Jiaotong University*, 2011.

[2] 蔡远利 (Cai, Y.L.). Partical Filtering(PF). *Xi'an Jiaotong University*, 2010. (包括序贯重要性采样 SIS、相对原始或严格的 PF 介绍等。)

[3] Too many to be listed. I will complete this list when I am a little free in the near future.

# Questions?