PARTICLE FILTERS[1]

7.1: Numeric integration to solve Bayesian recursion

- Recall from Chap. 4 that the optimal Bayesian recursion is to:
  - Compute the pdf for predicting x_k given all *past* observations

$$f(x_k \mid \mathbb{Z}_{k-1}) = \int_{R_{x_{k-1}}} f(x_k \mid x_{k-1}) f(x_{k-1} \mid \mathbb{Z}_{k-1}) \, \mathrm{d}x_{k-1}.$$

  - Update the pdf for estimating x_k given *all* observations via

$$f(x_k \mid \mathbb{Z}_k) = \frac{f(z_k \mid x_k) f(x_k \mid \mathbb{Z}_{k-1})}{f(z_k \mid \mathbb{Z}_{k-1})}.$$

- If we knew how to compute these pdfs, then we could find any desired estimator; *e.g.*, mean, median, mode, whatever.
- So far, we have assumed that computing these pdfs is intractable, so have made approximations to arrive at the KF, EKF, and SPKF.
- We now revisit this assumption.

Numeric integration

- The prediction step requires evaluating an integral.
- Normalization of $f(x_k \mid \mathbb{Z}_k)$ via its denominator requires an integral

$$f(z_k \mid \mathbb{Z}_{k-1}) = \int_{R_{x_k}} f(z_k \mid x_k) f(x_k \mid \mathbb{Z}_{k-1}) \, \mathrm{d}x_k.$$

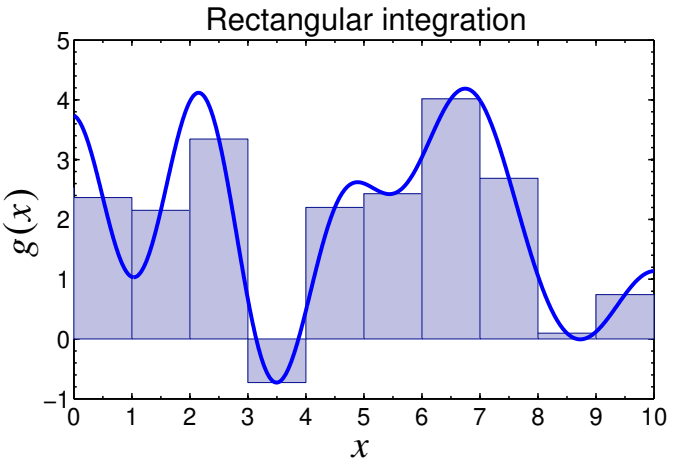
[1] The contents of this chapter of notes, including the MATLAB examples, are very closely based on the excellent course materials and lecture videos prepared by Prof. James McNames at Portland State University for "*ECE 510 State Space Tracking*," and are used with his permission and my gratitude.

- If state dimension n is large (*e.g.*, $n \geq 3$), evaluation of these multi-dimensional integrals may be impractical, even with modern CPUs.
- Computation needed for a given precision scales exponentially with n.
  - The particle filter will explore some clever ways to avoid this issue.
  - But, first, we look at numeric integration for low-order problems.

- Suppose that we wish to evaluate the definite integral

$$\text{integral} = \int_{x_{\min}}^{x_{\max}} g(x)\,\mathrm{d}x.$$

- The rectangular rule approximates $g(x)$ as piecewise constant.

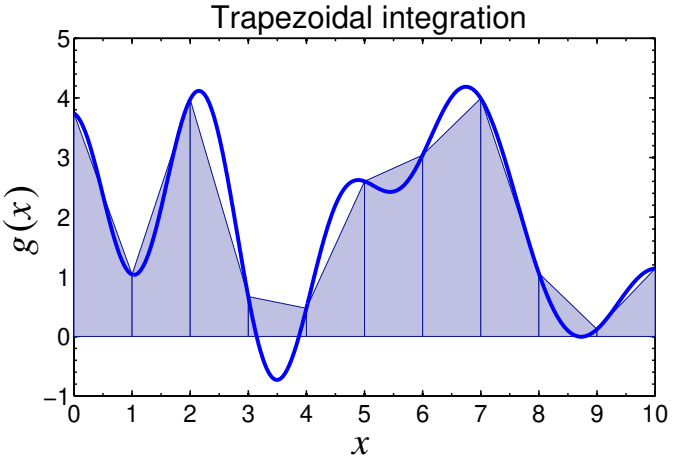


- Let N_r be the number of constant regions.
- Then, the width of each region is $w = (x_{\max} - x_{\min})/N_r$, and the integral can be approximated as

$$\text{integral} \approx \sum_{i=0}^{N_r-1} w \cdot g(x_{\min} + (i + 1/2)w).$$

- The trapezoidal rule approximates the function as piecewise linear.
- The integral approximation is

$$\text{integral} \approx w\left[\frac{g(x_{\min}) + g(x_{\max})}{2} + \sum_{i=1}^{N_r-1} g(x_{\min} + iw)\right].$$



Application to Bayesian inference

- Can write prediction step as

$$\begin{aligned} f(x_k \mid \mathbb{Z}_{k-1}) &= \int_{x_{\min}}^{x_{\max}} f(x_k \mid x_{k-1}) f(x_{k-1} \mid \mathbb{Z}_{k-1}) \, \mathrm{d}x_{k-1} \\ &= \int_{x_{\min}}^{x_{\max}} g(x; x_k, \mathbb{Z}_{k-1}) \, \mathrm{d}x \\ &\approx w \left[g(x_{\min}; x_k, \mathbb{Z}_{k-1}) + g(x_{\max}; x_k, \mathbb{Z}_{k-1}) \right] / 2 \\ &\quad + w \sum_{i=1}^{N_r - 1} g(x_{\min} + i w; x_k, \mathbb{Z}_{k-1}). \end{aligned}$$

- If we can compute this integral for every value of x_k, can directly estimate posterior distribution $f(x_k \mid \mathbb{Z}_k)$.
- Limiting factor is precision vs. speed: Each integral requires $\mathcal{O}(N_r)$ calculations at each of N_r evaluation points, or $\mathcal{O}(N_r^2)$ operations.

TRACKING EXAMPLE: We wish to track time-varying x_k for model

$$x_{k+1} = \alpha x_k + w_k$$

$$z_k = 0.1 x_k^3 + v_k$$

where $w_k \sim \mathcal{N}(0, \sigma_w^2)$ and $v_k \sim \mathcal{N}(0, \sigma_v^2)$ are mutually uncorrelated and IID.

- For the prediction step, we numerically integrate to approximate

$$f(x_k \mid \mathbb{Z}_{k-1}) = \int_{R_{x_{k-1}}} f(x_k \mid x_{k-1}) f(x_{k-1} \mid \mathbb{Z}_{k-1}) \, \mathrm{d}x_{k-1}.$$

- For the time-update step, we directly compute

$$f(x_k \mid \mathbb{Z}_k) = \frac{f(z_k \mid x_k) f(x_k \mid \mathbb{Z}_{k-1})}{f(z_k \mid \mathbb{Z}_{k-1})}.$$

- Note that the process model gives us

$$f(x_k \mid x_{k-1}) \sim \mathcal{N}(\alpha x_{k-1}, \sigma_w^2)$$

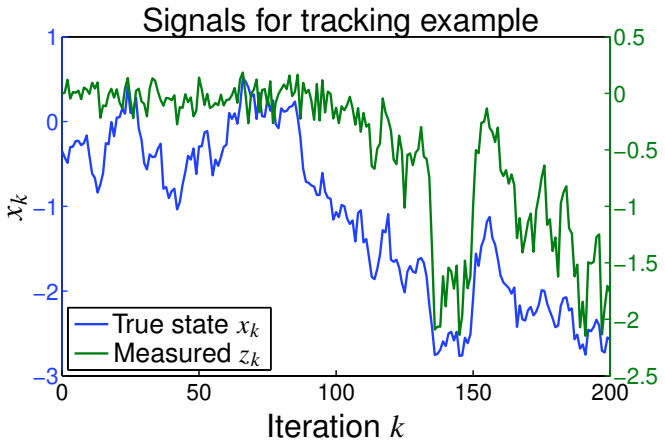
and the measurement model gives us

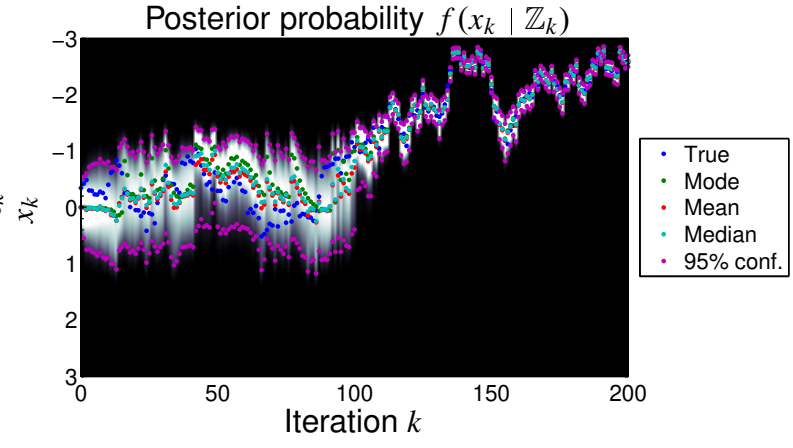
$$f(z_k \mid x_k) \sim \mathcal{N}(0.1x_k^3, \sigma_v^2).$$

- For this example, let $\sigma_w = 0.2$, $\sigma_v = 0.1$, $\alpha = 0.99$, and $x_0 \sim \mathcal{N}(0, \sigma_w^2)$.
- Note that $f(x_0 \mid \mathbb{Z}_0)$ needs to be specified, where

$$f(x_0 \mid \mathbb{Z}_0) = \frac{f(z_0 \mid x_0) f(x_0 \mid \mathbb{Z}_{-1})}{f(z_0 \mid \mathbb{Z}_{-1})}.$$

- Since $\mathbb{Z}_{-1}$ is unknown, we assume $f(x_0 \mid \mathbb{Z}_{-1}) = f(x_0) \sim \mathcal{N}(0, \sigma_w^2)$.
- Simulation run for 200 samples, range of integration from -3 to 3 with $N_r = 500$ regions.
- If we are able to find/approximate $f(x_k \mid \mathbb{Z}_k)$, we can compute mean, median, mode, whatever.
  - Can plot these point estimates as functions of time.
  - Alternately, can plot the sequence of distributions as a pseudo-color image (black is low probability; white is high probability).





```
% Code originally based on AngleTrackingExample.m by James McNames
% of Portland State University

clear; close all;

% Define User-Specified Parameters
nSamples              = 201;
xMin                  = -3;
xMax                  =  3;

measurementNoiseSigma = 0.1;
processNoiseSigma     = 0.2;
nRegions              = 500;
alpha                 = 0.99;

% Create Sequence of Observations from Model
x = zeros(nSamples+1,1); % time [0..nSamples]
z = zeros(nSamples  ,1); % time [0..nSamples-1]

x(1) = randn(1)*processNoiseSigma; % initialize x{0} ~ N(0,sigmaw^2)
for k=1:nSamples
  x(k+1) = alpha*x(k) + randn(1)*processNoiseSigma;
  z(k  ) = 0.1*x(k)^3 + randn(1)*measurementNoiseSigma;
end
x = x(1:nSamples); % constrain to [0..nSamples-1]

% Recursively Estimate the Marginal Posterior
xIntegration = linspace(xMin,xMax,nRegions).';
width = mean(diff(xIntegration));

posteriorFiltered  = zeros(nRegions,nSamples);
posteriorPredicted = zeros(nRegions,1);
xHatMedian         = zeros(nSamples,1);
xLower             = zeros(nSamples,1);
xUpper             = zeros(nSamples,1);

% initialize pdf for f(x{0})
prior = normpdf(xIntegration,0,processNoiseSigma);
% compute f(z{0}|x{0})*f(x{0}|Z{-1}) = f(z{0}|x{0})*f(x{0})
posteriorFiltered(:,1) = normpdf(z(1),0.1*xIntegration.^3,...
                                 measurementNoiseSigma).*prior;
% compute f(x{0}|Z{0})=f(z{0}|x{0})*f(x{0}|Z{-1})/[normalizing cst]
```



```
posteriorFiltered(:,1) = posteriorFiltered(:,1)/trapz(xIntegration,...
                                    posteriorFiltered(:,1));
for k=2:nSamples
  for cRegion=1:nRegions % find f(x{k}=x{i}|x{k-1}) for each x{i}
    % compute f(x{k}|x{k-1})
    prior = normpdf(xIntegration(cRegion),alpha*xIntegration,...
                    processNoiseSigma);
    % compute f(x{k}|Z{k-1})=integral[ f(x{k}|x{k-1})*f(x{k-1}|Z{k-1}) ]
    posteriorPredicted(cRegion) = trapz(xIntegration,...
                                        prior.*posteriorFiltered(:,k-1));
  end
  % compute f(z{k}|x{k})
  likelihood = normpdf(z(k),0.1*xIntegration.^3,measurementNoiseSigma);
  % compute f(x{k}|Z{k-1})*f(z{k}|x{k})
  posteriorFiltered(:,k) = likelihood.*posteriorPredicted;
  % normalize to get f(x{k}|Z{k})
  posteriorFiltered(:,k) = posteriorFiltered(:,k)/trapz(xIntegration,...
                           posteriorFiltered(:,k));

  % Find median, 95% confidence interval
  percentiles = cumtrapz(xIntegration,posteriorFiltered(:,k));
  iMedian = find(percentiles <= 0.5,1,'last');
  iLower = find(percentiles <= 0.025,1,'last');
  if isempty(iLower), iLower = 1; end;
  iUpper = find(percentiles >= 0.975,1,'first');
  if isempty(iUpper), iUpper = length(xIntegration); end
  xHatMedian(k) = xIntegration(iMedian);
  xLower(k) = xIntegration(iLower);
  xUpper(k) = xIntegration(iUpper);
end

[~,iMax] = max(posteriorFiltered);
xHatMode = xIntegration(iMax);
xHatMean = xIntegration'*posteriorFiltered.*width;

% Plot true state and observed sequence
figure(10); clf;
ax = plotyy(0:nSamples-1,x,0:nSamples-1,z);
legend('True state x_k','Measured z_k');
ylabel(ax(1),'x_k'); ylabel(ax(2),'z_k');
title('Signals for tracking example');
xlabel('Iteration k');
```

```
% Plot The Filtered Posterior
figure(3); clf; colormap('bone')
imagesc(0:nSamples-1,xIntegration,posteriorFiltered); hold on
caxis([0,prctile(posteriorFiltered(:),95)]); % more white on plot
hp = plot(0:nSamples-1,x,'.',0:nSamples-1,xHatMode,'.',...
          0:nSamples-1,xHatMean,'.',0:nSamples-1,xHatMedian,'.',...
          [0:nSamples-1,0:nSamples-1],[xUpper; xLower],'.');
set(hp,'markersize',12)

xlabel('Iteration k'); ylabel('x_k'); title('Posterior probability of x');
xlim([0 nSamples-1]); ylim([xMin xMax]);
legend(hp,{'True','Mode','Mean','Median','95% conf.'},...
       'location','eastoutside');
```

7.2: Monte-Carlo integration and the importance density

- Finding the optimal solution via numeric integration is usually not practical since it requires operation over a high-dimensional space.
- We need to find a way to beat the "curse of dimensionality" and get good approximate results using less computation.

THE PROBLEM: Consider the problem of numerically integrating $g(x)$, where we can evaluate $g(x)$ at any point x that we like:

$$\mu = \int g(x)\,\mathrm{d}x.$$

A POSSIBLE SOLUTION: Monte-Carlo methods "factor" $g(x)$ and write

$$\mu = \int \left[\frac{g(x)}{f(x)}\right] f(x)\,\mathrm{d}x,$$

where $f(x)$ is interpreted as a pdf.

- Then, the integral can be interpreted as $\mu = \mathbb{E}[g(x)/f(x)]$.
- Why does this help? The key idea is to write the integral as an expectation, then approximate the expectation with a sample mean based on sampling from the known pdf $f(x)$.
- That is, suppose we draw $N \gg 1$ random samples from pdf $f(x)$. Then,

$$\mu = \int \left[\frac{g(x)}{f(x)}\right] f(x)\,\mathrm{d}x = \mathbb{E}\left[\frac{g(x)}{f(x)}\right] \approx \frac{1}{N}\sum_{i=1}^{N} \frac{g(x^{(i)})}{f(x^{(i)})},$$

where $x^{(i)}$ is the ith sample drawn from $f(x)$, not x to the power i.

- If the samples are independent, then the estimate is unbiased and will "almost surely" converge to μ as $N \to \infty$.

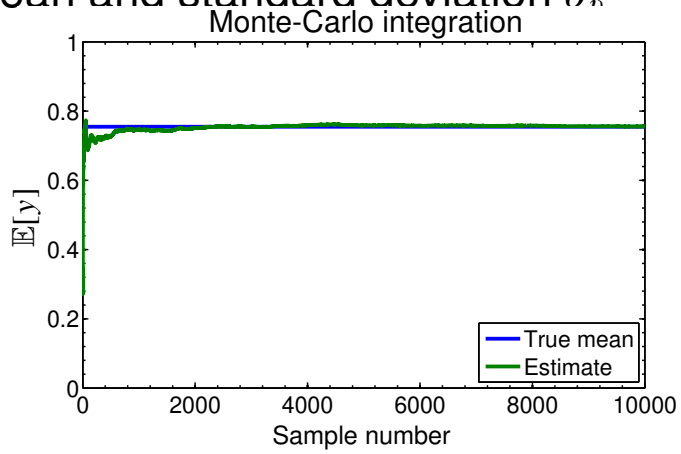
- Can choose $f(x)$ however we like, as long as its support is a superset of the support of $g(x)$.
  - Can choose $f(x)$ uniform, Gaussian, whatever;
  - Doesn't influence convergence; does affect rate of convergence.

EXAMPLE: Suppose we are given a random process $y_k = \cos(v_k)$ where $v_k \sim \mathcal{N}(0, \sigma_v^2)$ and we wish to estimate

$$\mathbb{E}[y] = \int_{-\infty}^{\infty} \cos(v) \frac{1}{\sqrt{2\pi\sigma_v^2}} \exp\left(\frac{-v^2}{2\sigma_v^2}\right) \mathrm{d}v.$$

- We draw Gaussian RVs with zero mean and standard deviation σ_v and compute

$$\frac{1}{N}\sum_{i=1}^{N} \cos(v^{(i)}).$$




```
N = 10000; sv = 0.75;
v = sv*randn(1,N); y = cos(v);
Ey = cumsum(y)./(1:N); plot(1:N,Ey);
```

- The "true mean" can be found as $\exp(-\sigma_v^2/2)$, or numerically.
- We see quite fast convergence to the neighborhood of the solution.

Reconsidering our problem

- In the prediction step, the integral we need to evaluate is already in the right form. That is,

$$f(x_k \mid \mathbb{Z}_{k-1}) = \int_{R_{x_{k-1}}} f(x_k \mid x_{k-1}) f(x_{k-1} \mid \mathbb{Z}_{k-1}) \, \mathrm{d}x_{k-1}$$

  is in the form

$$\text{something} = \int \text{fn}(x_{k-1}) f(x_{k-1}) \, \mathrm{d}x_{k-1}.$$

- To use a Monte-Carlo method to compute the integral, we need to draw random numbers from the posterior distribution $f(x_{k-1} \mid \mathbb{Z}_{k-1})$.
  - Problem: Unless this pdf is uniform or Gaussian or some other well-known distribution, we don't know how to do this.
  - That is, we may be able to evaluate the pdf at specific points (probably not, but we'll deal with that issue later) but we cannot draw IID random samples from the distribution.
  - However, there are other random number generators that we can draw samples from—how to use this?
- We define an importance density $q(x)$, which is a known pdf from which we are able to draw random samples.
- Then, if we wish to evaluate

$$\mu = \int g(x) f(x) \, \mathrm{d}x,$$

but cannot, since we cannot draw random values from $f(x)$, we could instead consider

$$\mu = \int \frac{g(x) f(x)}{q(x)} \cdot q(x) \, \mathrm{d}x,$$

where we *can* draw samples from $q(x)$.

- Then

$$\hat{\mu}_N = \frac{1}{N} \sum_{i=1}^{N} \frac{g(x^{(i)}) f(x^{(i)})}{q(x^{(i)})} = \frac{1}{N} \sum_{i=1}^{N} g(x^{(i)}) w(x^{(i)}),$$

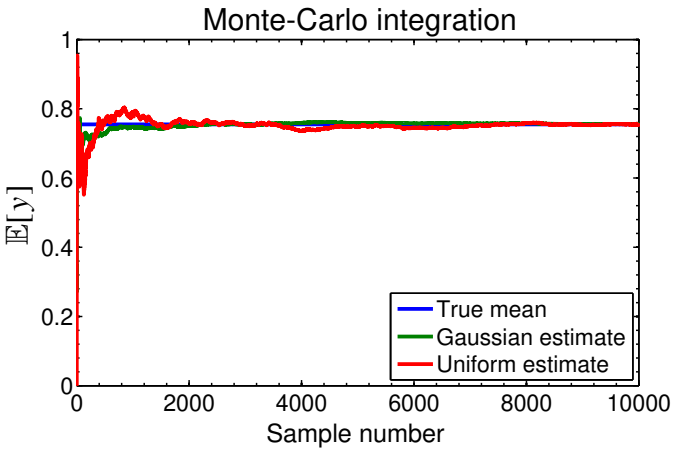
where $w(x^{(i)}) = f(x^{(i)})/q(x^{(i)})$.

- This requires that $q(x)$ have support (*i.e.*, $q(x) > 0$) everywhere $f(x)$ has support.

EXAMPLE: Same as before, want to find $\mathbb{E}[y]$, where $y = \cos(v)$ and $v \sim \mathcal{N}(0, \sigma_v^2)$. But, we now assume that we have only a uniform random number generator.

- Range of uniform RV was chosen to be $\pm 4\sigma_v$.

```
N = 10000; sv = 0.75;
q = 8*sv*(rand([1 N])-0.5);
y = cos(q).*exp(-q.^2/(2*sv^2)) ...
    /sqrt(2*pi*sv^2)/(1/(8*sv));
Ey = cumsum(y)./(1:N);
```



- Convergence is slower than before.
- Nonetheless, we get the same value in the end.

7.3: Weight normalization and impulse functions

Weight normalization

- We are getting closer to being able to evaluate the desired equations

$$f(x_k \mid \mathbb{Z}_{k-1}) = \int_{R_{x_{k-1}}} f(x_k \mid x_{k-1}) f(x_{k-1} \mid \mathbb{Z}_{k-1}) \, \mathrm{d}x_{k-1}.$$

$$f(x_k \mid \mathbb{Z}_k) = \frac{f(z_k \mid x_k) f(x_k \mid \mathbb{Z}_{k-1})}{f(z_k \mid \mathbb{Z}_{k-1})}.$$

- Typically, we do not calculate $f(z_k \mid \mathbb{Z}_{k-1})$ because
  - It involves evaluation of an integral
  - It does not depend on x_k
  - It serves only to normalize the posterior so $\int f(x_k \mid \mathbb{Z}_k) \, \mathrm{d}x_k = 1$.
- Thus, we will choose to evaluate something that is only proportional to the posterior—not the posterior itself.
- How does this affect Monte-Carlo integration?
- Assume that we know a function that is proportional to $f(x)$. Call it $\tilde{f}(x) = cf(x)$, where c is an unknown constant.
- This will bias our estimate

$$\hat{\mu}_b = \int g(x) \tilde{f}(x) \, \mathrm{d}x \approx \frac{1}{N} \sum_{i=1}^{N} g(x^{(i)}) \frac{\tilde{f}(x^{(i)})}{q(x^{(i)})} = c \frac{1}{N} \sum_{i=1}^{N} \frac{g(x^{(i)}) f(x^{(i)})}{q(x^{(i)})} = c\hat{\mu}.$$

- We can compute the bias if we (conceptually) replace the arbitrary function $g(x)$ by 1. That is, we seek to find $\mathbb{E}[1] = 1$ to expose the bias.

$$\hat{\mu}_1 = \int 1 \cdot \tilde{f}(x) \, \mathrm{d}x \approx \frac{1}{N} \sum_{i=1}^{N} 1 \cdot \frac{\tilde{f}(x^{(i)})}{q(x^{(i)})} = c \underbrace{\frac{1}{N} \sum_{i=1}^{N} \frac{f(x^{(i)})}{q(x^{(i)})}}_{1} = c.$$

- So, $c = \frac{1}{N}\sum_{i=1}^{N}\frac{\tilde{f}(x^{(i)})}{q(x^{(i)})} = \frac{1}{N}\sum_{i=1}^{N}\widetilde{w}(x^{(i)})$.
- If we divide our biased estimate by this constant, we remove the bias.
- So, in general, for any $g(x)$,

$$\begin{aligned}\hat{\mu}_N &= \frac{1}{N}\sum_{i=1}^{N} g(x^{(i)})\frac{f(x^{(i)})}{q(x^{(i)})} = \frac{1}{N}\sum_{i=1}^{N} g(x^{(i)})w(x^{(i)}) \\ &= \frac{1}{N}\sum_{i=1}^{N} g(x^{(i)})\left(\frac{\widetilde{w}(x^{(i)})}{\frac{1}{N}\sum_{i=1}^{N}\widetilde{w}(x^{(i)})}\right) \\ &= \sum_{i=1}^{N} g(x^{(i)})w^*(x^{(i)}),\end{aligned}$$

  where $w^*(x^{(i)}) = \frac{\widetilde{w}(x^{(i)})}{\sum_{i=1}^{N}\widetilde{w}(x^{(i)})}$.
- This last step is called weight normalization.

The impulse function

- We are making excellent progress in the development of background to the particle filter.
- One remaining concept relates to how we will represent the probability density functions that are recursively updated.
- It turns out that we will use the multidimensional Dirac delta (impulse) function to do so.
  - $\int \delta(x)\,\mathrm{d}x = 1$ and $\delta(x-a) = 0\,\forall\, x \neq a$.
- From these properties, can show $f(a) = \int f(x)\delta(x-a)\,\mathrm{d}x$.

- A multivariable impulse is defined as

$$\delta(x) = \prod_i \delta(x_i) = \delta(x_0)\delta(x_1)\cdots\delta(x_n),$$

  where x_i is the ith element of vector x.

- Particle filters represent the joint posterior as a sum of impulses.

- To investigate this, consider a simple example with a Gaussian pdf.

- If we model the pdf as $\hat{f}(x) = \frac{1}{N}\sum_{i=1}^{N}\delta(x - x^{(i)})$, where $x^{(i)}$ are IID samples drawn from $f(x)$, then substitution of $\hat{f}(x)$ into our integral to find a mean is

$$\mu = \int g(x) f(x)\,\mathrm{d}x$$

$$\hat{\mu} = \int g(x)\hat{f}(x)\,\mathrm{d}x = \int g(x)\frac{1}{N}\sum_{i=1}^{N}\delta(x - x^{(i)})\,\mathrm{d}x$$

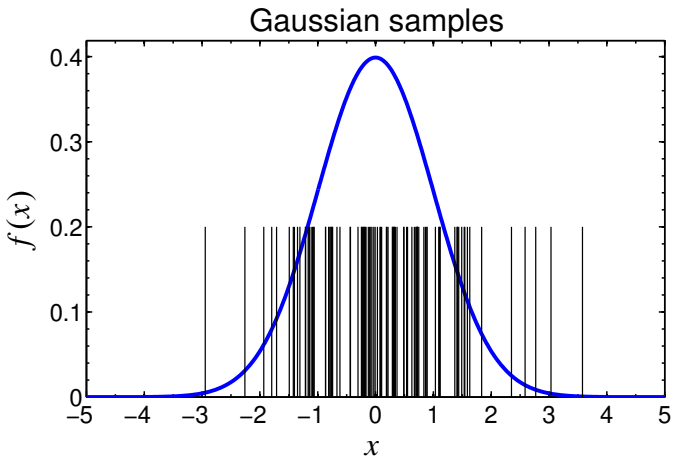
$$= \frac{1}{N}\sum_{i=1}^{N}\int g(x)\delta(x - x^{(i)})\,\mathrm{d}x = \frac{1}{N}\sum_{i=1}^{N} g(x^{(i)}).$$

- So, using our approximation $\hat{f}(x)$, we arrived at the same estimator as derived earlier based on Monte-Carlo integration.

- But, is it fair to say $\hat{f}(x) \approx f(x)$?

- The figure draws $N = 100$ impulses sampled from $f(x)$ without arrows (for clarity), superimposed on $f(x)$.

- Hard to argue any equivalence except under an integral.



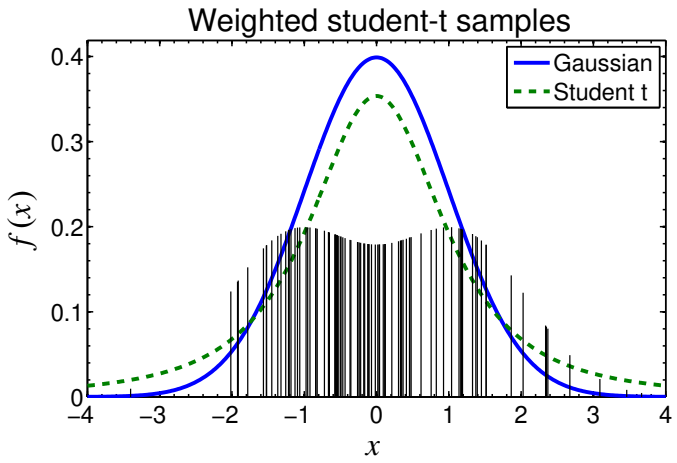
- This is a kind of scatter plot, but if you were given only the impulses it would have been tricky to guess the true shape of the distribution.
- In some sense, $f(x)$ and $\hat{f}(x)$ are nothing alike. They *are* alike in that $\int g(x) f(x)\,\mathrm{d}x \approx \int g(x)\hat{f}(x)\,\mathrm{d}x$ and

$$F(x) = \int_{-\infty}^{x} f(\alpha)\,\mathrm{d}\alpha \approx \int_{-\infty}^{x} \hat{f}(\alpha)\,\mathrm{d}\alpha.$$

- So, we can calculate percentiles (including median) from $\hat{f}(x)$.
- But, there is no obvious way to calculate mode of $f(x)$ from $\hat{f}(x)$.[2]
- Also, in our application, we won't be able to draw samples from $f(x)$, but will instead draw samples from an importance density $q(x)$.
- The equivalent effect on the estimated pdf is to weight it

$$\hat{f}(x) = \sum_i w^{(i)} \delta(x - x^{(i)}), \qquad w^{(i)} = f(x^{(i)})/q(x^{(i)}).$$

- This converges in the same manner as the unweighted approximation of the pdf.
- In example, $q(x)$ is the student-t distribution with two degrees of freedom.



- In general, $f(x^{(i)}) \neq q(x^{(i)})$, so $w^{(i)} \neq 1$.

[2] Can use Viterbi particle filters — see Prof. McNames' course notes and videos for details.

7.4: Sequential importance sampling

- At last, we get to our first particle-filter algorithm!
- Our goal is to estimate either the joint or marginal pdf

$$f(\mathbb{X}_k \mid \mathbb{Z}_k) \qquad \text{or} \qquad f(x_k \mid \mathbb{Z}_k).$$

- Note that the first is the pdf of the entire trajectory up to time k, $\mathbb{X}_k = \{x_0, x_1, \ldots, x_k\}$. Very high (and growing) dimension.

KEY IDEA: We assume that the joint pdf can be expressed as

$$\hat{f}(\mathbb{X}_k \mid \mathbb{Z}_k) = \sum_{i=1}^{N_p} w_k^{(i)} \delta(\mathbb{X}_k - \mathbb{X}_k^{(i)}), \qquad \sum_{i=1}^{N_p} w_k^{(i)} = 1.$$

- This is a discrete weighted approximation to true posterior $f(\mathbb{X}_k \mid \mathbb{Z}_k)$, parameterized and completely defined by $\{\mathbb{X}_k^{(i)}, w_k^{(i)}\}_{i=1}^{N_p}$.
- To find these parameters efficiently, we desire a recursive algorithm.
- First, we will need to select an importance density for the prediction step because we cannot sample from the pdfs.
- Any pdf (including an important density) must factor

$$q(\mathbb{X}_k \mid \mathbb{Z}_k) = q(x_k \mid \mathbb{X}_{k-1}, \mathbb{Z}_k) q(\mathbb{X}_{k-1} \mid \mathbb{Z}_k).$$

- We *choose* an importance density that factors

$$q(\mathbb{X}_k \mid \mathbb{Z}_k) = q(x_k \mid \mathbb{X}_{k-1}, \mathbb{Z}_k) q(\mathbb{X}_{k-1} \mid \mathbb{Z}_{k-1}).$$

- Not quite recursive, so again, we narrow down the kind of importance density that we will use by requiring that it satisfy

$$q(\mathbb{X}_k \mid \mathbb{Z}_k) = q(x_k \mid x_{k-1}, z_k) q(\mathbb{X}_{k-1} \mid \mathbb{Z}_{k-1}).$$

- This factoring permits samples of the trajectory $\mathbb{X}_k^{(i)}$ to be created sequentially after each observation z_k is made by appending $x_k^{(i)}$ to $\mathbb{X}_{k-1}^{(i)}$, where $x_k^{(i)}$ is made from z_k and $x_{k-1}^{(i)}$ only.
  - Note: $x_k^{(i)}$ is kth member of particle $\mathbb{X}_k^{(i)}$, where particle comprises entire trajectory and $x_k^{(i)}$ is only the most recent point in trajectory;
  - Similarly, $x_{k-1}^{(i)}$ is penultimate point $\mathbb{X}_k^{(i)}$, or final point in $\mathbb{X}_{k-1}^{(i)}$.
- We will also need a weight-update equation

$$w_k^{(i)} \propto \frac{f(\mathbb{X}_k^{(i)} \mid \mathbb{Z}_k)}{q(\mathbb{X}_k^{(i)} \mid \mathbb{Z}_k)}.$$

- The samples are chosen by importance sampling, where the samples are taken at the particle locations $\mathbb{X}_k^{(i)}$, and we get to choose the importance density $q(\mathbb{X}_k \mid \mathbb{Z}_k)$.
- We'll say more on choosing $q(\cdot)$ later, but for now we can assume we've chosen one (*e.g.*, Gaussian).
- To compute the weights, we'll also need to be able to recursively calculate $f(\mathbb{X}_k^{(i)} \mid \mathbb{Z}_k)$.
  - Note that we can write:

$$\begin{aligned} f(\mathbb{X}_k^{(i)}, z_k \mid \mathbb{Z}_{k-1}) &= f(\mathbb{X}_k^{(i)} \mid z_k, \mathbb{Z}_{k-1}) f(z_k \mid \mathbb{Z}_{k-1}) \\ &= f(\mathbb{X}_k^{(i)} \mid \mathbb{Z}_k) f(z_k \mid \mathbb{Z}_{k-1}). \end{aligned}$$

  - We can also write:

$$\begin{aligned} f(\mathbb{X}_k^{(i)}, z_k \mid \mathbb{Z}_{k-1}) &= f(z_k \mid \mathbb{Z}_{k-1}, \mathbb{X}_k^{(i)}) f(\mathbb{X}_k^{(i)} \mid \mathbb{Z}_{k-1}) \\ &= f(z_k \mid x_k^{(i)}) f(\mathbb{X}_k^{(i)} \mid \mathbb{Z}_{k-1}) \\ &= f(z_k \mid x_k^{(i)}) f(x_k^{(i)}, \mathbb{X}_{k-1}^{(i)} \mid \mathbb{Z}_{k-1}) \end{aligned}$$

$$= f(z_k \mid x_k^{(i)}) f(x_k^{(i)} \mid \mathbb{Z}_{k-1}, \mathbb{X}_{k-1}^{(i)}) f(\mathbb{X}_{k-1}^{(i)} \mid \mathbb{Z}_{k-1})$$
$$= f(z_k \mid x_k^{(i)}) f(x_k^{(i)} \mid x_{k-1}^{(i)}) f(\mathbb{X}_{k-1}^{(i)} \mid \mathbb{Z}_{k-1}).$$

- Equating these two forms gives

$$f(\mathbb{X}_k^{(i)} \mid \mathbb{Z}_k) = \frac{f(z_k \mid x_k^{(i)}) f(x_k^{(i)} \mid x_{k-1}^{(i)})}{f(z_k \mid \mathbb{Z}_{k-1})} f(\mathbb{X}_{k-1}^{(i)} \mid \mathbb{Z}_{k-1})$$
$$\propto f(z_k \mid x_k^{(i)}) f(x_k^{(i)} \mid x_{k-1}^{(i)}) f(\mathbb{X}_{k-1}^{(i)} \mid \mathbb{Z}_{k-1})$$
$$q(\mathbb{X}_k^{(i)} \mid \mathbb{Z}_k) = q(x_k^{(i)} \mid x_{k-1}^{(i)}, z_k) q(\mathbb{X}_{k-1}^{(i)} \mid \mathbb{Z}_{k-1})$$
$$w_k^{(i)} \propto \frac{f(z_k \mid x_k^{(i)}) f(x_k^{(i)} \mid x_{k-1}^{(i)}) f(\mathbb{X}_{k-1}^{(i)} \mid \mathbb{Z}_{k-1})}{q(x_k^{(i)} \mid x_{k-1}^{(i)}, z_k) q(\mathbb{X}_{k-1}^{(i)} \mid \mathbb{Z}_{k-1})}$$
$$= \frac{f(z_k \mid x_k^{(i)}) f(x_k^{(i)} \mid x_{k-1}^{(i)})}{q(x_k^{(i)} \mid x_{k-1}^{(i)}, z_k)} w_{k-1}^{(i)}.$$

- So, we end up with a beautiful recursion for the weights.
  - We get to choose $q(x_k^{(i)} \mid x_{k-1}^{(i)}, z_k)$.
  - $f(z_k \mid x_k^{(i)})$ is obtained from measurement model $z_k = h_k(x_k, u_k, v_k)$.
  - $f(x_k^{(i)} \mid x_{k-1}^{(i)})$ is from process model $x_k = f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1})$.
  - $w_{k-1}^{(i)}$ is known from the previous time step for every particle i.
- So, we now have a clever way to estimate the joint posterior

$$\hat{f}(\mathbb{X}_k^{(i)} \mid \mathbb{Z}_k) = \sum_{i=1}^{N_p} w_k^{(i)} \delta(\mathbb{X}_k - \mathbb{X}_k^{(i)}).$$

- But, what we often care about is the marginal posterior $f(x_k^{(i)} \mid \mathbb{Z}_k)$, which we can get by integrating the joint.

- Not as scary as it might seem.

$$\hat{f}(x_k \mid \mathbb{Z}_k) = \int \hat{f}(\mathbb{X}_k \mid \mathbb{Z}_k)\, \mathrm{d}\mathbb{X}_{k-1} = \int \sum_{i=1}^{N_p} w_k^{(i)} \delta(\mathbb{X}_k - \mathbb{X}_k^{(i)})\, \mathrm{d}\mathbb{X}_{k-1},$$

where

$$\delta(\mathbb{X}_k) = \prod_{i=0}^{n} \delta(x_i) = \delta(x_0)\delta(x_1)\cdots\delta(x_n).$$

- So,

$$\begin{aligned}\hat{f}(x_k \mid \mathbb{Z}_k) &= \sum_{i=1}^{N_p} w_k^{(i)} \int \delta(x_0 - x_0^{(i)}) \cdots \delta(x_{k-1} - x_{k-1}^{(i)}) \delta(x_k - x_k^{(i)})\, \mathrm{d}\mathbb{X}_{k-1} \\ &= \sum_{i=1}^{N_p} w_k^{(i)} \delta(x_k - x_k^{(i)}).\end{aligned}$$

- The impulse notation makes the algorithm completely recursive, if only the marginal is needed.

Sequential importance sampling (SIS) algorithm

- So, the most basic general particle-filter algorithm, known as sequential importance sampling, can be written as:

$$\{x_k^{(i)}, w_k^{(i)}\}_{i=1}^{N_p} = \mathrm{SIS}\left(\{x_{k-1}^{(i)}, w_{k-1}^{(i)}\}_{i=1}^{N_p}, z_k\right)$$

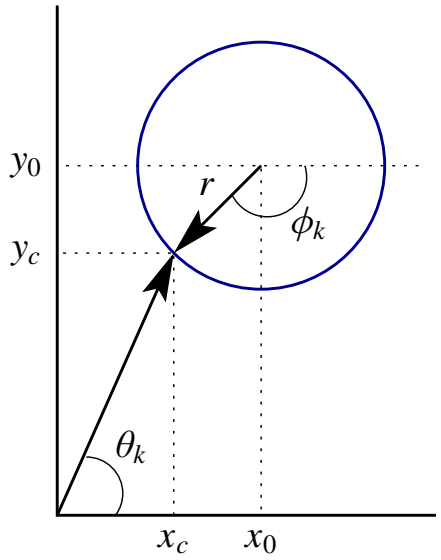
  - For $i = 1 : N_p$,
    - Draw RV $x_k^{(i)} \sim q(x_k \mid x_{k-1}^{(i)}, z_k)$ — randomly!
    - Calculate unnormalized weights:

$$\widetilde{w}_k^{(i)} = \frac{f(z_k \mid x_k^{(i)})\, f(x_k^{(i)} \mid x_{k-1}^{(i)})}{q(x_k^{(i)} \mid x_{k-1}^{(i)}, z_k)} w_{k-1}^{(i)}.$$

    - Normalize the weights: $w_k^{(i)} = \dfrac{\widetilde{w}_k^{(i)}}{\sum_{i=1}^{N_p} \widetilde{w}_k^{(i)}}$.

7.5 Example

- Suppose we have a car driving around a circular track with radius 200 m.
- Velocity is nearly constant at 2° per sample, but may be perturbed by additive Gaussian noise modeling bumps, maneuvers, etc.
- We have a sensor located $500\sqrt{2}$ m from the center of the track that measures angle to the car, θ_k with noise $\pm 5^\circ$ typical.



- We wish to estimate ϕ_k (*i.e.*, $x_k = \phi_k$).
- Note that $x_{c,k} = r\cos(\phi_k) + x_0$ and $y_{c,k} = r\sin(\phi_k) + y_0$.
- Model sensor noise $v_k \sim \mathcal{N}(0, (5^\circ)^2)$ so (where $z_k = \theta_k$)

$$\theta_k = \tan^{-1}\left(\frac{y_{c,k}}{x_{c,k}}\right) + v_k.$$

- For the process model, we write

$$\phi_{k+1} = \phi_k + \omega + w_k,$$

where $w_k \sim \mathcal{N}(0, \sigma_w^2)$ and $\omega = 2$ degrees/sample.

- Let $y_0 = x_0 = 500$ m, $r = 200$ m,

$$w_k \sim \mathcal{N}\left(0, \left(\frac{10\pi}{180}\right)^2\right)$$

$$v_k \sim \mathcal{N}\left(0, \left(\frac{5\pi}{180}\right)^2\right)$$

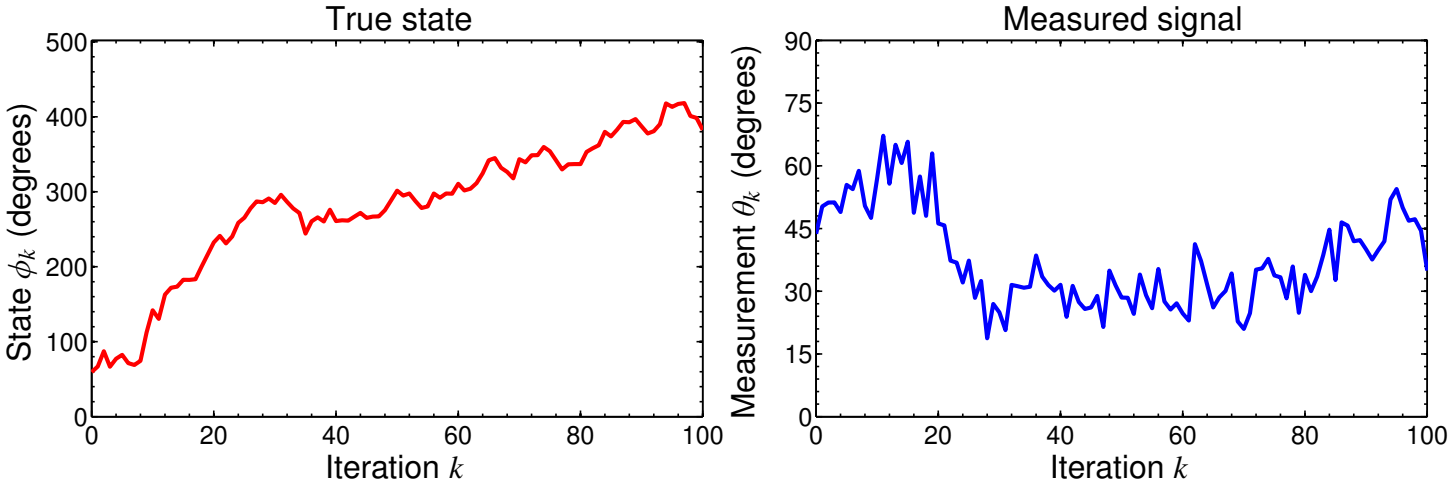
$$\phi_0 \sim \mathcal{N}\left(\left(\frac{50\pi}{180}\right), \left(\frac{10\pi}{180}\right)^2\right).$$

- We have several design choices before we can employ the filter.
  - Importance density $q(\phi_k \mid \phi_{k-1}, \theta_k)$:
    - Easiest choice is
      $q(\phi_k \mid \phi_{k-1}, \theta_k) = f(\phi_k \mid \phi_{k-1}) \sim \mathcal{N}(\phi_{k-1} + 2, \sigma_w^2)$.
  - Number of particles:
    - Performance versus computation tradeoff.
    - Useful approach—as many as you have patience for.
    - Here, we use $N_p = 500$.
  - Estimator of state:
    - We'll use the mean, as it is easiest.
- Also, the likelihood $f(\theta_k \mid \phi_k) \sim \mathcal{N}\left(\tan^{-1}\left(\frac{r\sin(\phi_k) + y_0}{r\cos(\phi_k) + x_0}\right), \sigma_v^2\right)$.

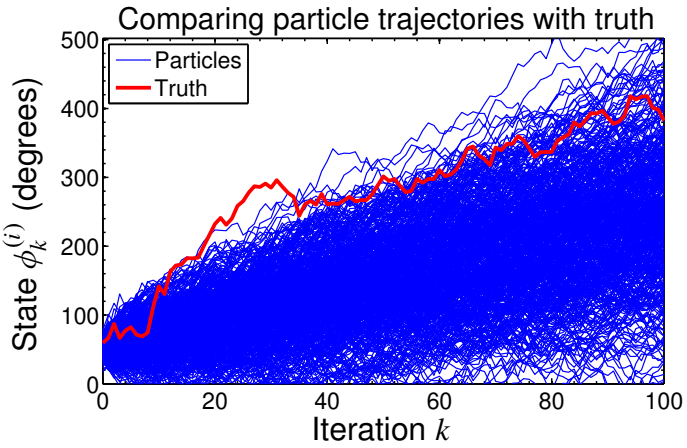
```
% Define constants describing problem
% -----------------------------------
initialStateSigma     = 10*pi/180;
measurementNoiseSigma = 5.0*pi/180;
processNoiseSigma     = 10*pi/180;
stateInitial          = 50*pi/180;
xo = 500; yo = 500; r = 200; omega = 2*pi/180; nSamples = 100;

% Create sequence of observations from model
% ------------------------------------------
x = zeros(nSamples+1,1); z = zeros(nSamples,1);
processNoise     = randn(nSamples,1)*processNoiseSigma;
measurementNoise = randn(nSamples,1)*measurementNoiseSigma;

x(1) = stateInitial + randn(1)*initialStateSigma;
for k=1:nSamples
  x(k+1) = x(k) + omega + processNoise(k);
    z(k) = atan((r*sin(x(k))+yo)./(r*cos(x(k))+xo)) + ...
           measurementNoise(k);
end
x = x(1:nSamples);
```

```
% Create the Particles
% --------------------
nParticles = 500;
xp = zeros(nParticles,nSamples); wp = zeros(nParticles,nSamples);
xPHatMean = zeros(nSamples,1); % mean-particle estimate of state

for k=1:nSamples
  processNoise = randn(nParticles,1)*processNoiseSigma;
  for cp=1:nParticles
    if k==1
      xp(cp,k)    = stateInitial + randn(1)*initialStateSigma;
      wp(cp,k)    = 1/nParticles;
    else
      xp(cp,k)    = xp(cp,k-1) + omega + processNoise(cp);
      zp          = atan((r*sin(xp(cp,k))+yo)./(r*cos(xp(cp,k))+xo));
      likelihood = normpdf(z(k),zp,measurementNoiseSigma);
      wp(cp,k)    = wp(cp,k-1)*likelihood;
    end
  end
  wp(:,k) = wp(:,k)/sum(wp(:,k));
  xPHatMean(k) = sum(wp(:,k).*xp(:,k));
end
```

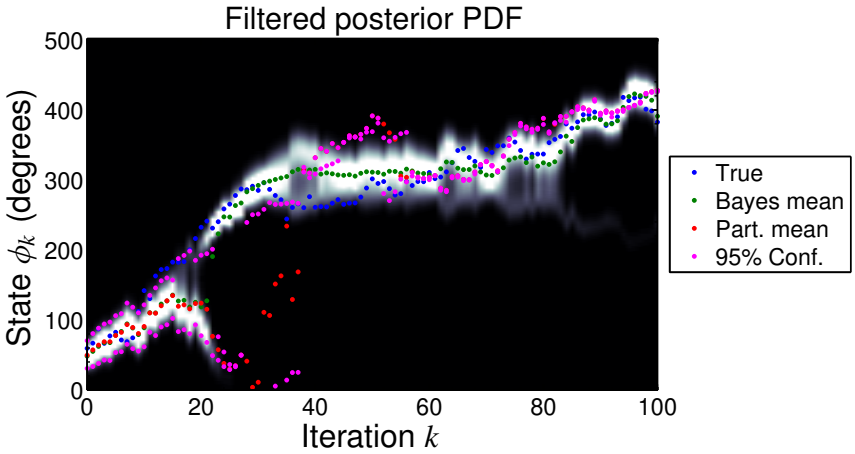
- In the time domain, the true state and measured output are



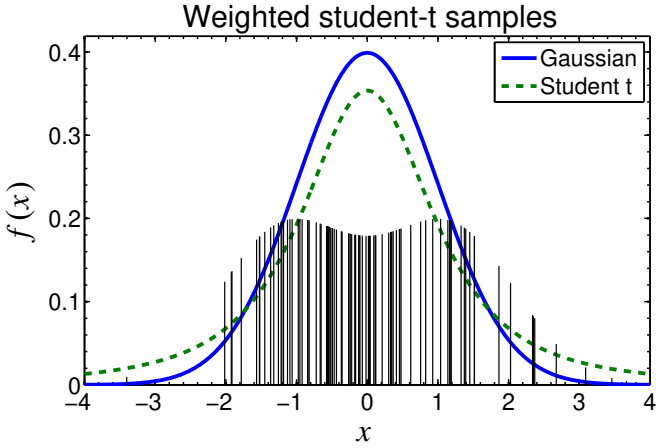
- The particles, with true state trajectory overlaid, are shown.
- Particles represent many possible trajectories to choose from.



- Particle-filter estimate is pretty poor, compared to Bayes optimal.
- Need to dig deeper to figure out what's going wrong with our estimator.



7.6: How to display a marginal posterior



- Would like to visualize particle estimate of marginal posterior $\hat{f}(x_k \mid \mathbb{Z}_k)$.
- But, consists of weighted impulses, as figure from prior example shows.
- Difficult to grasp intuitively.
- A popular method of estimating density is to add a series of "bumps" together, where each bump is scaled by the impulse weight and centered at the impulse location.
  - This is merely popular; it is not necessarily "best."
- The bumps are called kernels and should have the following properties

$$b(u) \geq 0, \qquad \int_{-\infty}^{\infty} b(u)\,\mathrm{d}u = 1.$$

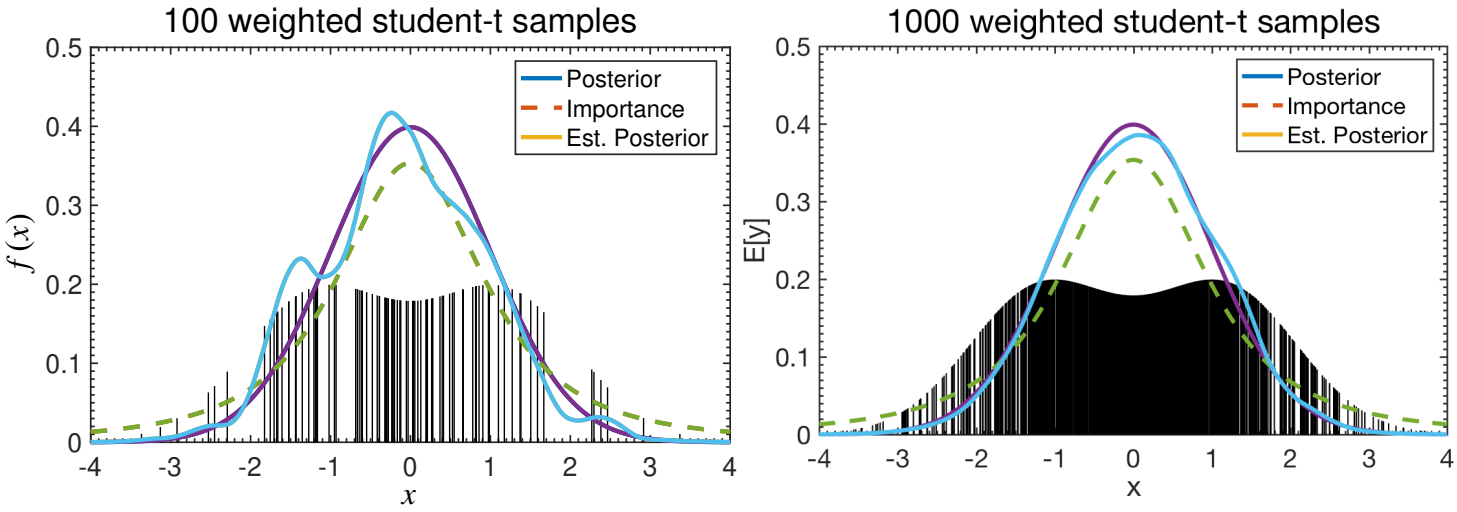
- The width of the kernel is specified by σ. Typically

$$b_\sigma(u) = \frac{1}{\sigma} b\left(\frac{u}{\sigma}\right),$$

  where it is easy to show that $\int b_\sigma(u)\,\mathrm{d}u = \int b(u)\,\mathrm{d}u = 1.$
- Then, a kernel density estimator is simply expressed as

$$\hat{f}(x_k) = \sum_{i=1}^{N_p} w_k^{(i)} b_\sigma(x_k - x_k^{(i)}).$$

- Kernels usually have even symmetry; Gaussian-shapes are popular.
- For the prior example, we can estimate the posterior as

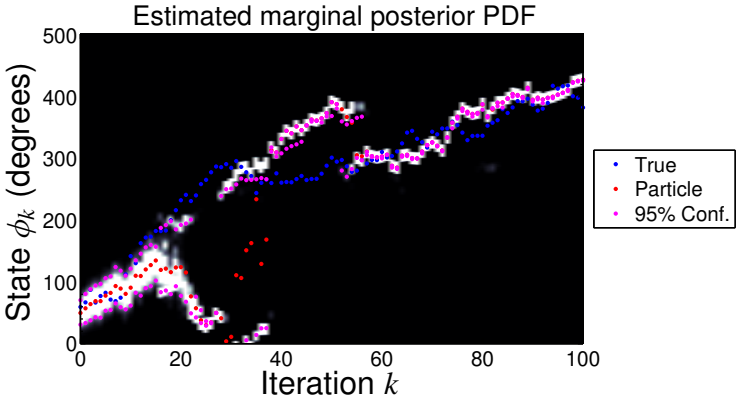


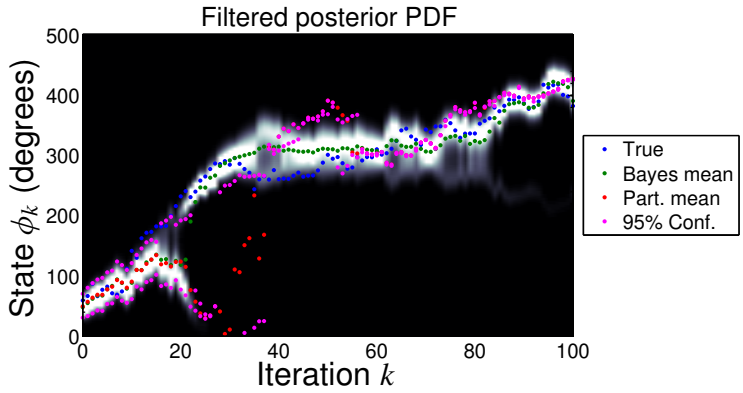
- Popular kernels include

$$\begin{aligned}
\text{Epanechnikov:} \quad & b(u) = c(1-u^2)\Pi(u/2) \\
\text{Bi-weight:} \quad & b(u) = c(1-u^2)^2\Pi(u/2) \\
\text{Tri-weight:} \quad & b(u) = c(1-u^2)^3\Pi(u/2) \\
\text{Triangular:} \quad & b(u) = c(1-|u|)\Pi(u/2) \\
\text{Gaussian:} \quad & b(u) = ce^{-u^2}
\end{aligned}$$

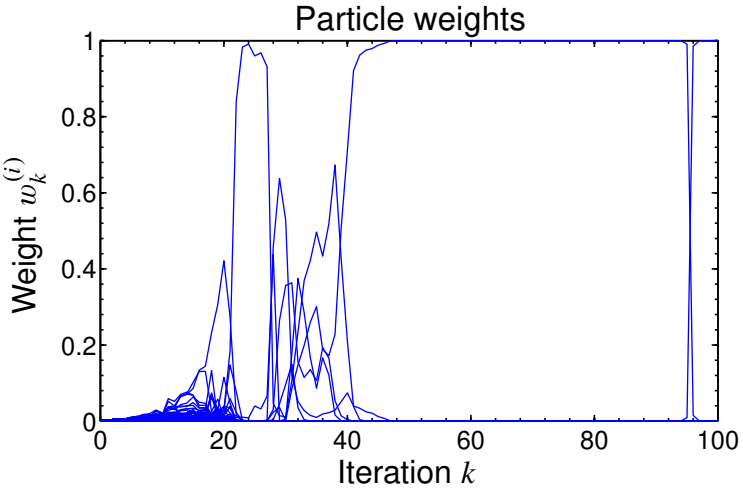
  where c is a constant chosen to make the kernel integrate to 1, and $\Pi(u)$ is the unit-pulse function.

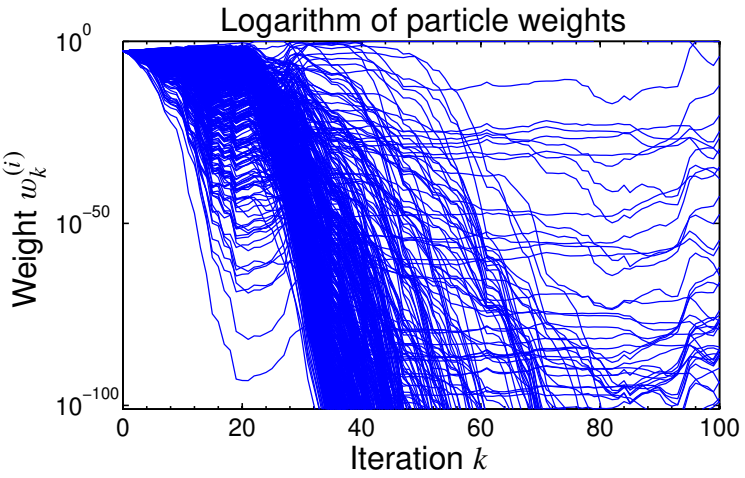
- The bump's width is the critical parameter. Much more important than the choice of kernel.
  - σ large: Smoother pdf estimate, more biased, less variable;
  - σ small: Coarser pdf estimate, less biased, more variable.
- Comparing estimated marginal posterior pdf using kernel functions to the true posterior, we get a hint of the problem.
- The posterior starts out okay, but quickly collapses to a single likely trajectory.



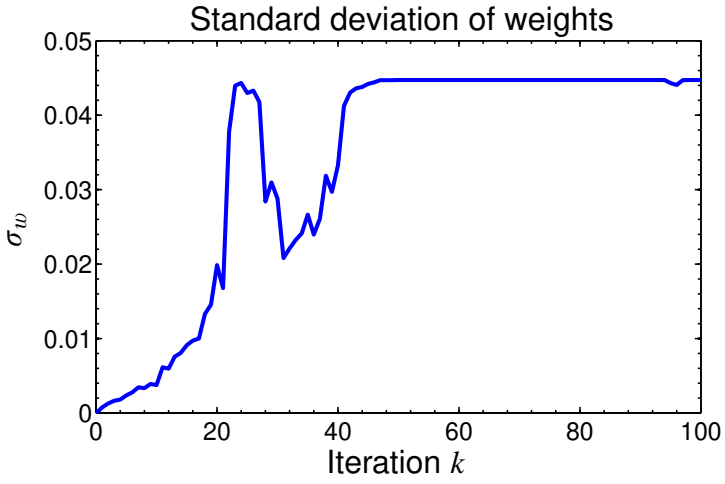


- This leads us to look at particle weights (on natural and log scales):





- The weights start at the same value, $1/N_p$, but very quickly one particle "takes over." The weights of the other particles decay to around zero, and they have no influence on the results.
- A lot of computation for no good reason.
- We might also look at the standard deviation of the weights.
- The increasing value shows that weights are becoming very unequal, and some particles are being ignored.



7.7: Reducing particle degeneracy

- In the example, we found that
  - The particle filter starts off tracking well,
  - Performance rapidly deteriorates,
  - Most of the weights become really small,
  - Increasing the number of particles helps only a little.
- If only a few of the weights are large, only a few trajectories contribute to the state estimate

$$\hat{x}_k = \sum_{i=1}^{N_p} w_k^{(i)} x_k^{(i)}.$$

- So, most of the particles are wasted clock cycles. What happened?
  - Keep in mind that the particles are entire state trajectories $\mathbb{X}_k^{(i)}$.
  - Drawing samples from a high-dimensional space.
  - Difficult to draw probable state trajectories.
  - Dimension grows linearly with time.
  - Becomes exponentially difficult to sample efficiently.
  - Known as the degeneracy problem.
- Is manifest as $w_k^{(i)} \to 0$ as k increases.
- To help visualize, we introduce the effective sample size, defined as

$$N_{p_e,k} = \frac{1}{\sum_{i=1}^{N_p} (w_k^{(i)})^2}.$$

- This is not a derivation, but a useful definition. Note that it makes sense for the boundary cases:

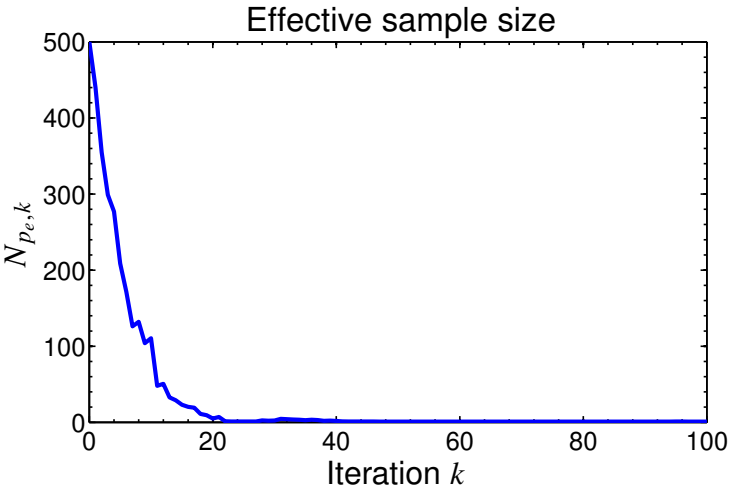
- If all weights are equal, $w_k^{(i)} = 1/N_p$ and

$$N_{p_e,k} = \frac{1}{\sum_{i=1}^{N_p}(1/N_p)^2} = \frac{1}{N_p(1/N_p)^2} = N_p.$$

- If all weights but one are zero, we have

$$N_{p_e,k} = \frac{1}{1} = 1.$$

- Thus, $N_{p_e,k}$ is bounded and $1 \leq N_{p_e,k} \leq N_p$.
- For our prior example, we see that $N_{p_e,k}$ quickly decreases.
- We are essentially calculating the state from a single trajectory after 20 or so samples.



Resampling

- Idea is to eliminate samples with low importance weights and replicate samples with high importance weights.
- We resample from our set $\{x_k^{(i)}, w_k^{(i)}\}$, with replacement, whenever $N_{p_e,k}$ drops below some threshold (perhaps some percentage of N_p).
- Can resample as many times as we like, but typically we draw as many samples as we started with.
- The new particles are drawn with a probability given by the importance weights, so we can assign uniform weights $1/N_p$ to the resulting IID samples.
- A systematic resampling method is as follows:

- Let c be the cumulative summation of weights

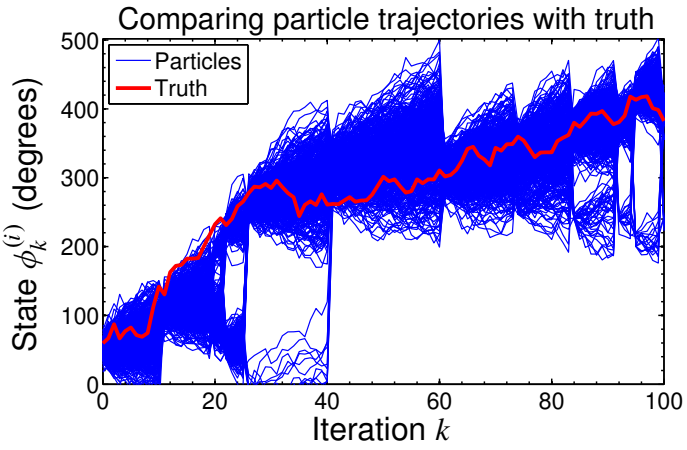
$$c_k^{(i)} = \sum_{j=1}^{i} w_k^{(j)}.$$

- Make a single draw $u_1 \sim \mathcal{U}(0, N_p^{-1})$ and set $i = 1$.
- For each $j = 1 : N_p$,
  - $u_j = u_1 + (j-1)/N_p$.
  - While $u_j > c_k^{(i)}$, $i = i + 1$.
  - Assign new $x_k^{(j)} =$ old $x_k^{(i)}$, $w_k^{(j)} = 1/N_p$.

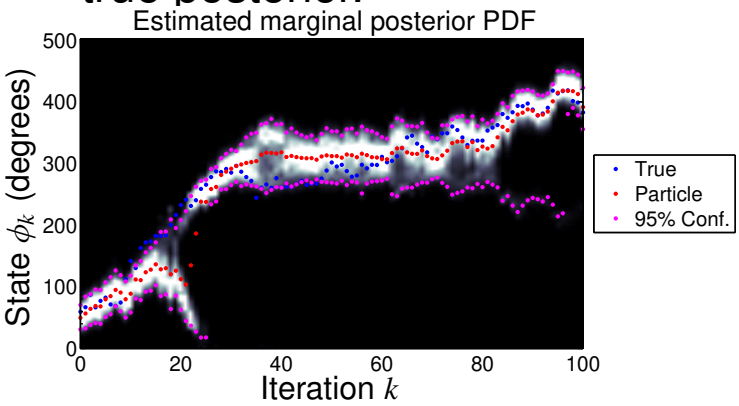
- In MATLAB code, this looks like:

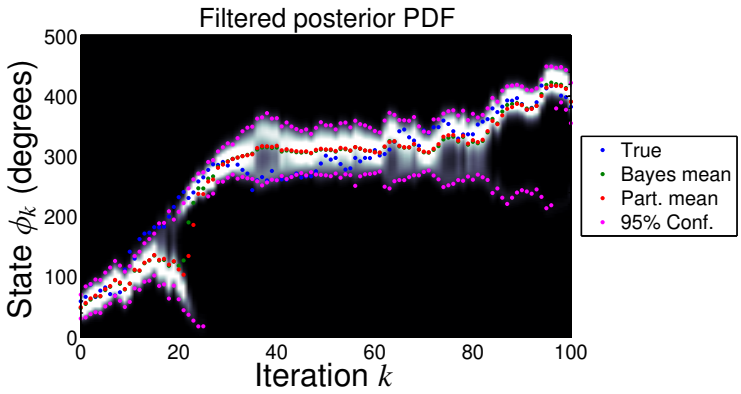
```
nEffective = 1/sum(wp(:,k).^2);
if nEffective<nEffectiveThreshold
  iPick = 1; xpNew = zeros(nParticles,1); c = cumsum(wp(:,k));
  uj = rand(1)/nParticles;
  for cp=1:nParticles
    while uj>c(iPick) && iPick<nParticles,
      iPick = iPick + 1;
    end
    xpNew(cp) = xp(iPick,k);
    uj = uj + 1/nParticles;
  end
  xp(:,k) = xpNew;
  wp(:,k) = ones(nParticles,1)/nParticles;
end
```

- Repeating the previous example (exactly same system data), with resampling threshold constraint $N_{p_e,k} \geq 100$, we get the following results.
- The particle trajectories have improved immensely.

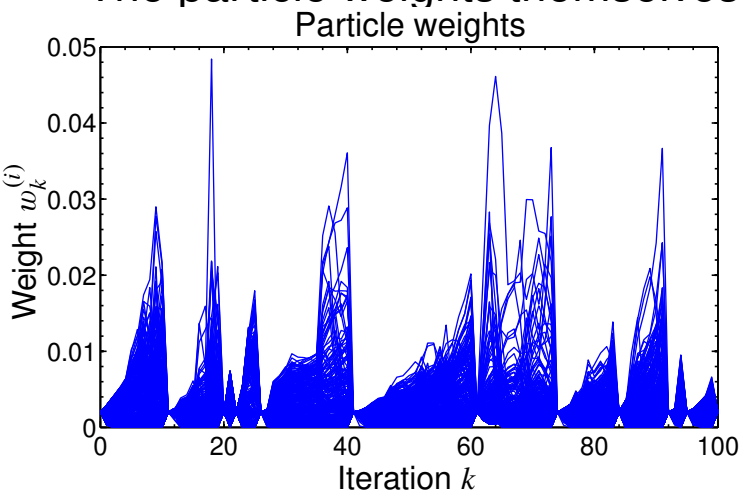


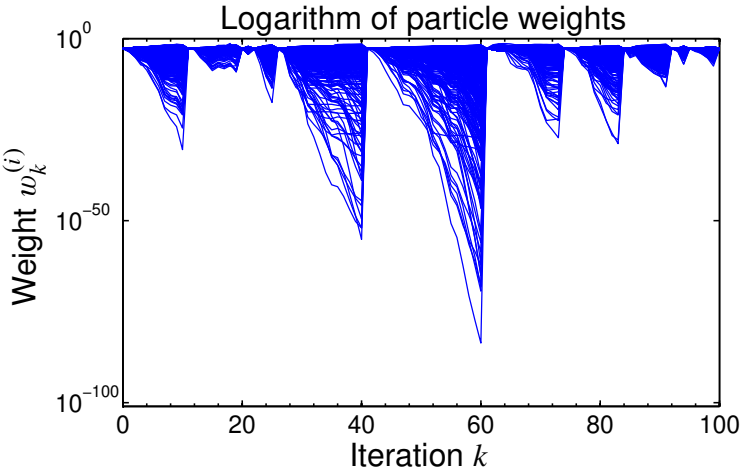
- The estimated marginal posterior is now a much closer match to the true posterior.



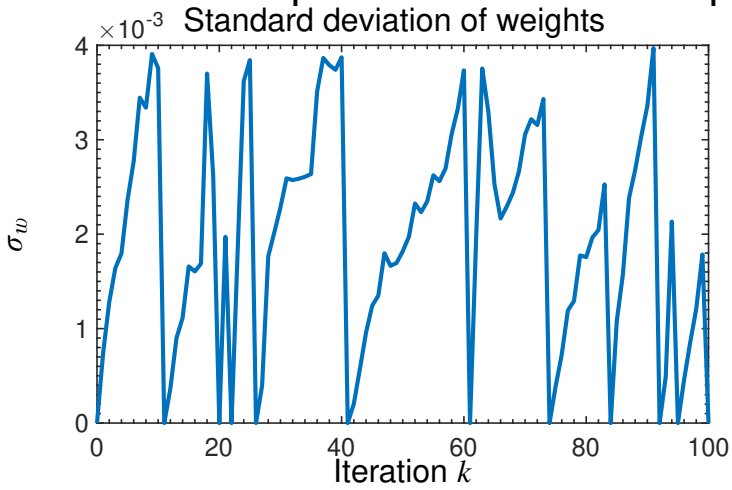


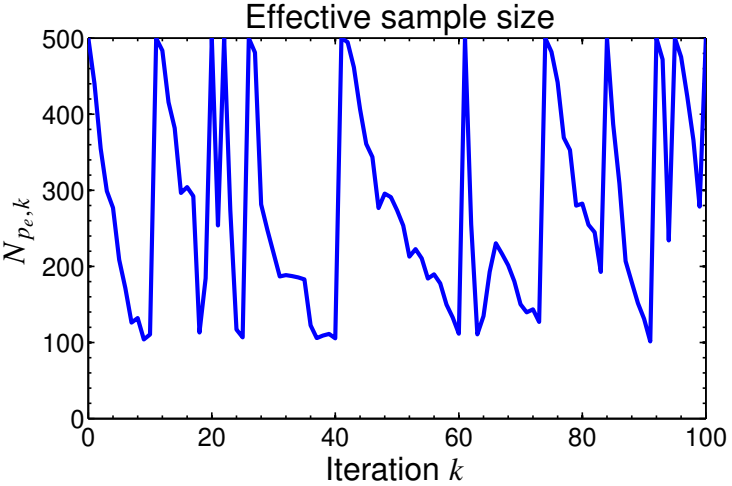
- The particle weights themselves have improved in diversity.





- The weight standard deviation has improved, and the effective number of particles has also improved.





- So, with resampling, particle filters become a much more effective methodology.

Where from here

- This section on particle filters is only an introduction:
  - There is a lot of literature on choosing importance densities, on finding MAP estimators, etc.
  - Prof. McNames' course covers some of these more advanced topics, if you are interested.
  - The material we have covered here should prepare you for his lectures and help you read and interpret the literature.
- We now take a different direction and look at multi-target tracking applications of Kalman filters.