Hand-in assignments
PhD course on Sequential Monte Carlo methods 2019

Martin Hellkvist
Signals and Systems, Uppsala University

H.1 Importance sampling theory

(a) Show that the estimator $\hat{Z}$ is unbiased.

Solution: Plug the estimator $\hat{Z} = \frac{1}{N}\sum_{i=1}^{N}\frac{\tilde{\pi}(x^i)}{q(x^i)}$ into the expectation:

$$\mathbb{E}[\hat{Z}] = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\frac{\tilde{\pi}(x^i)}{q(x^i)} = \frac{1}{N}N\mathbb{E}\frac{\tilde{\pi}(x)}{q(x)}, \tag{1}$$

where the last equality holds because the expectation is constant w.r.t the samples x^i. Using the definition of expectation

$$\mathbb{E}[X] = \int xp(x)dx \tag{2}$$

and that $\tilde{\pi}(x) := Z\pi(x)$, the expression becomes:

$$\mathbb{E}[\hat{Z}] = \int \frac{\tilde{\pi}(x)}{q(x)}q(x)dx = \int \tilde{\pi}(x)dx = \int Z\pi(x)dx = Z\int \pi(x)dx = Z. \tag{3}$$

So the expectation of the estimate is exactly the true value, hence the estimator is unbiased.

(b) Using the Cauchy distribution as proposal, implement an IS that samples from the standard normal distribution. Find the gamma that minimizes the asymptotic variance of the IS and provide plots for the approximation of the normalizing constant $\sqrt{2\pi}$.

Solution: the function to minimize is $\int \pi^2(x)/q(x)dx - 1$:

$$\gamma^* = \arg\min_{\gamma>0}\int \frac{\pi^2(x)}{q(x)}dx - 1 = \arg\min_{\gamma>0}\int \frac{\pi^2(x)}{q(x)}dx = \arg\min_{\gamma>0} f(\gamma). \tag{4}$$

Plugging in the expressions $\pi(x) = \frac{1}{\sqrt{2\pi}}\exp\{-x^2/2\}$ and $q(x) = \frac{\gamma}{\pi(\gamma^2+x^2)}$, the function $f(\gamma)$ becomes

$$f(\gamma) = \int \frac{\pi^2(x)}{q(x)}dx = \int \frac{1}{2\pi}\frac{\exp\{-x^2\}}{\frac{\gamma}{\pi(\gamma^2+x^2)}}dx = \int \frac{\exp\{-x^2\}}{2}\frac{\gamma^2+x^2}{\gamma}dx = \tag{5}$$

$$= \int \frac{\gamma^2}{2\gamma}\exp\{-x^2\}dx + \int \frac{1}{2\gamma}x^2\exp\{-x^2\}dx. \tag{6}$$

The integration is over the whole real axis for x, and the expression becomes

$$f(\gamma) = \frac{\gamma^2}{4\gamma}\sqrt(\pi)\mathrm{erf}(x) + \frac{1}{8\gamma}\left(\sqrt{\pi}\mathrm{erf}(x) - 2\exp\{-x^2\}x\right)\Bigg|_{x\to-\infty}^{x\to+\infty} = \tag{7}$$

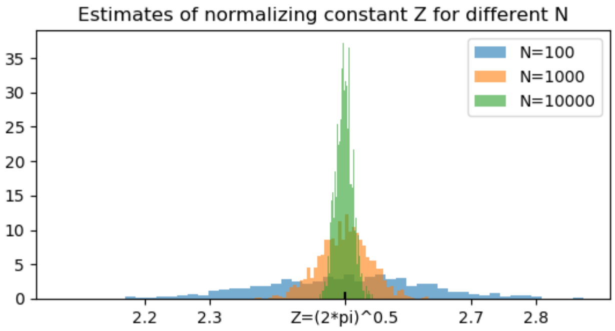

Figure 1: For three different values of N we see different variance in the estimate of Z. As N increases, the variance decreases. The average of each histogram is close to the true value $Z = \sqrt{2\pi}$. The histogram was generated with 1000 simulations for each N.

$$= \frac{(2\gamma^2+1)\sqrt{\pi}\mathrm{erf}(x) - 2\exp\{-x^2\}x}{8\gamma}\Bigg|_{x\to-\infty}^{x\to+\infty} = \tag{8}$$

$$= \frac{\Big((2\gamma^2+1)\sqrt{\pi}\cdot 1 - 2\cdot 0\Big) - \Big((2\gamma^2+1)\sqrt{\pi}\cdot(-1) - 2\cdot 0\Big)}{8\gamma} = \frac{(2\gamma^2+1)}{4\gamma}\sqrt{\pi}, \tag{9}$$

where $\mathrm{erf}(x) := \dfrac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$ is the *error function*.

The function $f(\gamma)/\sqrt{\pi} = \frac{(2\gamma^2+1)}{4\gamma} = \frac{\gamma}{2} + \frac{1}{4\gamma}$ is convex for $\gamma > 0$ and has a unique minimum at $\gamma^* \to \frac{d}{d\gamma}\left(\frac{\gamma}{2} + \frac{1}{4\gamma}\right) = 0$ which is found as

$$\frac{1}{2} - \frac{1}{4\gamma^2} = 0 \Leftrightarrow \gamma = 1/\sqrt{2}. \tag{10}$$

The simulation results illustrated in figure 1 demonstrates how the implemented IS is unbiased.

(c) Now target the Cauchy distribution using the standard normal distribution.

Solution: The variance of the estimator is now

$$\int \frac{\pi^2(x)}{q(x)}dx - 1 = \int \frac{\gamma^2}{\pi^2(\gamma^2+x^2)^2}\frac{1}{\frac{1}{\sqrt{2\pi}}\exp\{-x^2/2\}}dx - 1 = \int \frac{\gamma^2}{\pi^2(\gamma^2+x^2)^2}\sqrt{2\pi}\exp\{x^2\}dx - 1, \tag{11}$$

which does not converge, i.e., the variance is infinite. Although the estimator is theoretically unbiased, it is numerically infeasible to attain an unbiased estimate.

Figure 2 illustrates the IS with the standard normal distribution as proposal, and figure 3 instead shows the histograms generated with the normal distribution with standard deviation 20 instead. An increased variance of the proposal decreases the bias that is actually implicitly introduced by this infinite variance.

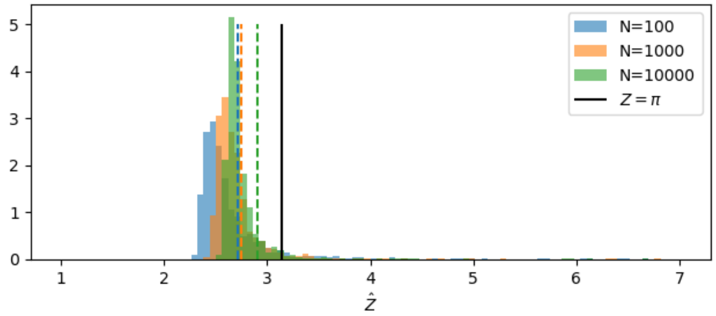

Figure 2: Task 1c. The distribution of the estimated normalization was skewed due to thin tails of the proposal.

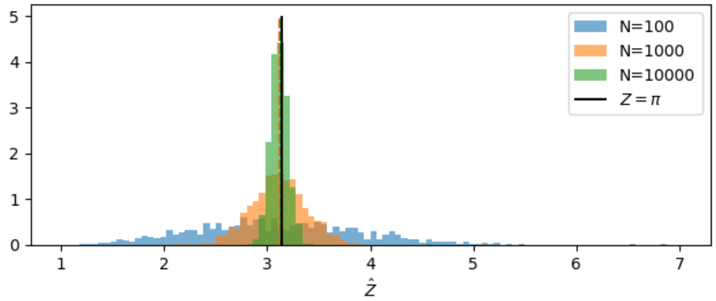

Figure 3: Task 1c. When the tails of the proposal were heavier, the estimates clearly clearly illustrated the unbiasedness of the IS.

Table 1: 2c. Simulation results for the bootstrap particle filter. MAD is the mean absolute difference between the BPF estimated trajectory and the Kalman trajectory. The variance denotes the variance of the MAD over the iterations.

N	MAD	Variance
10	0.1489	0.020
50	0.0412	0.023
100	0.0271	0.023
2000	0.0060	0.024
5000	0.0042	0.024
Kalman	0.0000	0.025

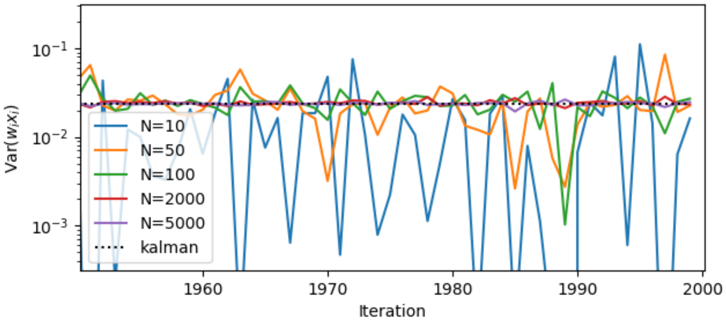

Figure 4: 2c. Variance over particles per iteration of the BPF state estimates compared to the Kalman variance.

H.2 Particle filter for a linear Gaussian state-space model

(a) The pdf's are $p(x_t|x_{t-1}) = \mathcal{N}(x_t; 0.8x_{t-1}, 0.5)$ and $p(y_t|x_t) = \mathcal{N}(y_t; 2x_t, 0.1)$.

(b) Because we perfectly know the noise covariances, and that the model matches the simulated system, the Kalman filter is the optimal linear filter. The particle filter can on the other hand model nonlinearities in a model. But because our model is linear, we do not expect the particle filter trajectory to be closer to the true on average.

(c) After implementing the bootstrap filter, the variance was computed for each N as $Var(\hat{x}_t) = \sum_{i=1}^{N} w_t^i (x_t^i)^2 + \left(\sum_{i=1}^{N} w_t^i x_t^i\right)^2$. The Kalman filter's variance converges after only a few iterations. The variances are plotted in figure 4.

Letting MAD denote the mean absolute difference between the BPF state estimate and the Kalman state estimate, averaged over time, the results together with average variance of the BPF is presented in table 1.

(d) Derivation of the Fully Adapted Particle Filter pdf's follows. The first proposal, for the adjustment multipliers $\boldsymbol{\nu}$:

$$\nu_{t-1}^i = p(y_t|x_{t-1}^i) = \frac{p(y_t, x_{t-1}^i)}{p(x_{t-1}^i)} \propto p(y_t, x_{t-1}^i). \tag{12}$$

From the probabilistic model we have

$$\mathcal{N}(y_t; 2x_t^i, 0.1), \tag{13}$$

from which we can plug in the model for x_t^i:

$$\mathcal{N}(y_t; 2(0.8x_{t-1}^i + V_t), 0.1) = \mathcal{N}(y_t; 1.6x_{t-1}^i + 2V_t, 0.1) = \mathcal{N}(y_t; 1.6x_{t-1}^i, 2^2 \cdot 0.5 + 0.1), \tag{14}$$

thus, we have

$$p(y_t|x_{t-1}^i) = \mathcal{N}(y_t; 1.6x_{t-1}^i, 2.1). \tag{15}$$

N	MAD
10	0.0388
50	0.0181
100	0.0122
2000	0.0028
5000	0.0019
Kalman	0.0000

Table 2: 2d. The mean average difference between the Fully Adapted Particle filter's state trajectory and the Kalman filter's state trajectory.

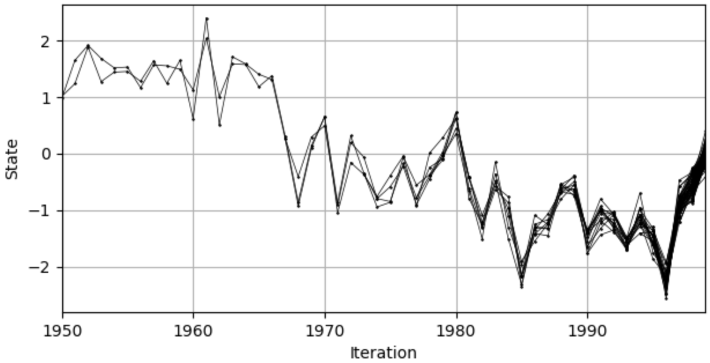

Figure 5: 2e. Particle genealogy from time 1950 to 1999 for the fully adaptive particle filter.

For the optimal proposal q:

$$q(x_t|x_{t-1}, y_t) \propto p(y_t|x_t, x_{t-1})p(x_t|x_{t-1}) = \mathcal{N}(y_t; 2x_t, 0.1)\mathcal{N}(x_t; 0.8x_{t.1}, 0.5) \propto \tag{16}$$

$$\propto \exp\{-\frac{(y_t - 2x_t)^2}{2 \cdot 0.1}\} \exp\{-\frac{(x_t - 0.8x_{t-1})^2}{2 \cdot 0.5}\} \propto \tag{17}$$

$$\propto \exp\{-\frac{(x_t - \frac{1}{42}(20y_t - 1.6x_{t-1}))^2}{2\frac{1}{42}}\}, \tag{18}$$

so we get the normal distribution q:

$$q(x_t|x_{t-1}, y_t) = \mathcal{N}(x_t;\ \frac{1}{42}(20y_t - 1.6x_{t-1}), \frac{1}{42}) \tag{19}$$

Table 2 presents the MAD between the FAPF and the Kalman state trajectories for one simulation. Compared to the results for the BPF, the MAD is now a lot lower for small N, while they are still lower for larger N's but not as strikingly.

(e) Figure 5 illustrates how only a few particles at time 1950 are remaining as ancestors for the final particles.

(f) With systematic resampling the FAPF obtains less degeneracy and the final particles have many more ancestors. As illustrated in figure 6, there are a lot more diversity in paths than in figure 5. In figure 7 it is illustrated how the final 100 particles have 2-3 ancestors from the very beginning of the particle filter iteration.

(g) For this task ESS triggered resampling was added to implementation. With the threshold set to 50, resampling triggers around every 27:th iteration. The path degeneracy became lower than in task 2f We can see that the amount of ancestors the final particles had in figure 8 is higher than in the results from 2f, in figure 7.

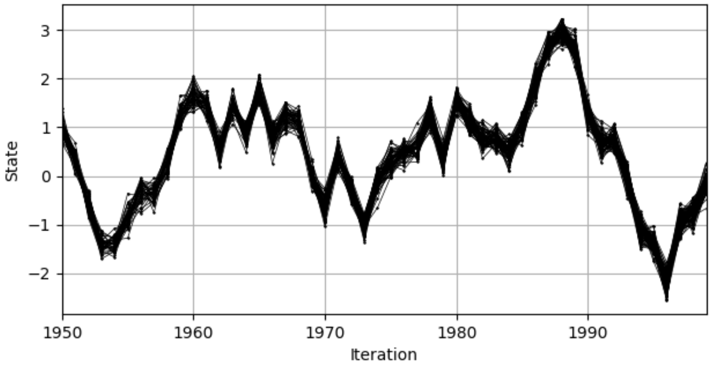

Figure 6: 2f. The genealogy for the final 100 particles.

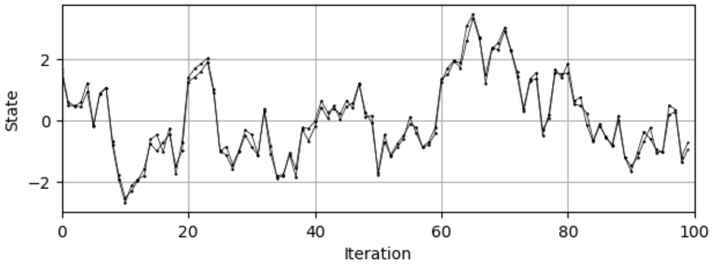

Figure 7: 2f. The early genealogy for the final 100 particles.

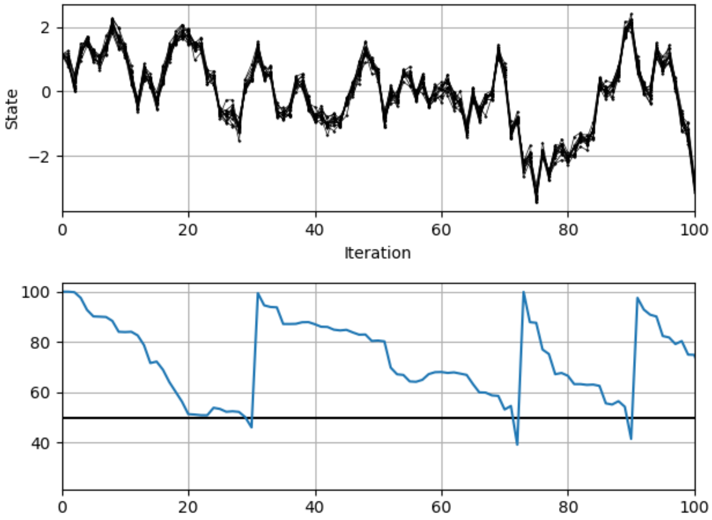

Figure 8: 2g. (Top) The early genealogy for the final 100 particles using systematic resampling with ESS trigger at $N_{\text{thresh}} = 50$. (Bottom) The ESS in each iteration from 0 to 100.

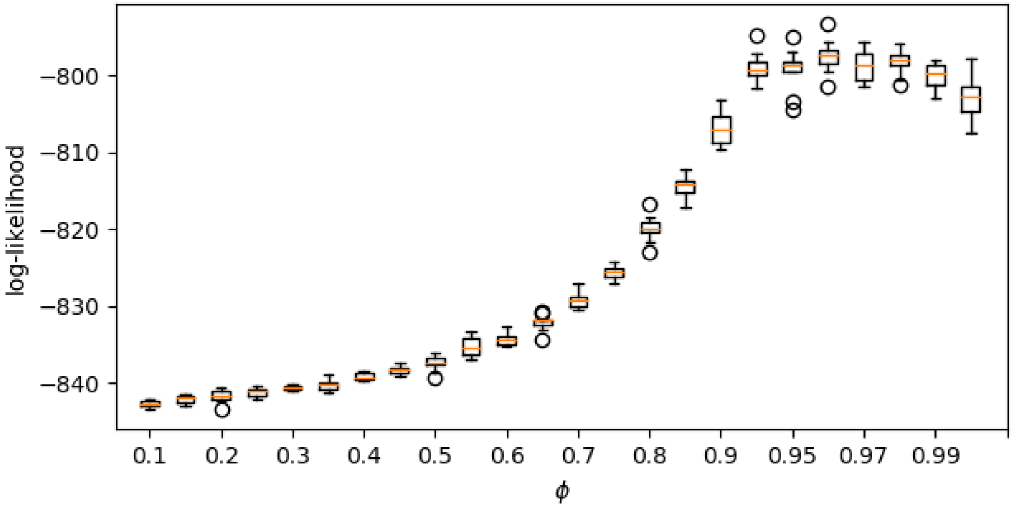

Figure 9: 3a. Boxplot of the 10 log-likelihood estimates for each value of ϕ. Note that the spacing of the phi-axis is smaller between 0.95 and 1, than it is before.

H.3 Parameter estimation in the stochastic volatility model

(a) Using a BPF with 100 particles, 10 estimates of the log-likelihood was created for the parameter ϕ on the range $\{0.10, 0.15, \dots, 0.95, 0.96, \dots, 1.00\}$. The results is presented as a boxplot in figure 9 and they indicate that phi around 0.96 gives the highest likelihood for the given data.

(b) The implementation for this exercise consisted of two main parts: the bootstrap particle filter (BPF) to return the log-likelihood, and the particle Metropolis-Hastings (PMH) algorithm to iterate over parameters proposals. In the BPF, the unnormalized logarithmic weights $\tilde{v}_t$ are computed as

$$\tilde{v}_t^i = \ln \frac{1}{\sqrt{2\pi\beta^2 \exp\{x_t^i\}}} - y_t^2, \; i = 1, \dots, N. \tag{20}$$

Then their maximum is subtracted to get the shifted log-weights v_t^i:

$$v_t^i = \tilde{v}_t^i - \max_i \tilde{v}_t^i. \tag{21}$$

The log-likelihood is then computed as

$$z_t = \max_i \tilde{v}_t^i - \ln N + \ln \sum_{i=1}^{N} \exp\{v_t^i\}, \tag{22}$$

so that the total log-likelihood Z for the PMH iteration was the sum over z_t over all $t = 1, \dots, T$:

$$Z = \sum_{t=1}^{T} z_t. \tag{23}$$

The resampling weights w_t^i are then computed as

$$w_t^i = \frac{\exp\{v_t^i\}}{\sum_{t=1}^{T} \exp\{v_t^i\}}. \tag{24}$$

The implementation computes the acceptance probabilities by the logarithms of the fraction for α as described in the lecture, by summing the log-likhelihoods Z' for the proposed parameters β', σ', and Z for the previous parameters β, σ:

$$\alpha = \min\{1, \exp\{\alpha_1' - \alpha_1\}\}, \tag{25}$$

$$\alpha_1' = Z' + \ln(\mathcal{IG}(\beta', 0.01, 0.01)) + \ln(\mathcal{IG}(\sigma', 0.01, 0.01)), \tag{26}$$

$$\alpha_1 = Z + \ln(\mathcal{IG}(\beta, 0.01, 0.01)) + \ln(\mathcal{IG}(\sigma, 0.01, 0.01)). \tag{27}$$

The step size μ for the Gaussian random walk was set to 0.01 for both β and σ, the number of PMH iterations to $M = 10000$ with $N = 1000$ particles while the initialization was set to 0.5 for both variables. The Gaussian random walk is performed on the non-squared parameters, so $\beta', \sigma' \sim \mathcal{N}(0, 0.01^2)$.

With these settings the plots in figure 10 were generated. There is seemingly a large bump in the evolution for β^2 between iteration 3000 and 5000. This might be part of the burn-in, but yet they are included in the histogram.

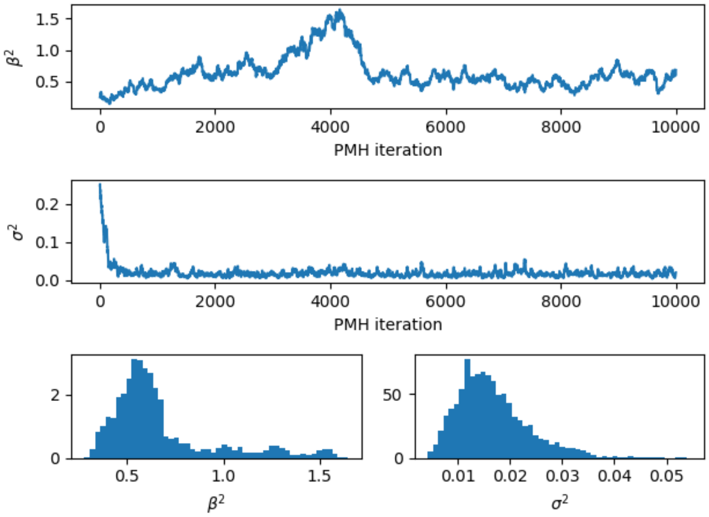

Figure 10: 3b. (Top and middle) The evolution of β^2 and σ^2 over the PMH iterations. (Bottom) Histograms of the parameters over the last 8000 iterations.

(c) The particle Gibbs algorithm requires samples from the inverse Gamma distribution, which was implemented as inverting the sample from the gamma distribution in numpy: `1/numpy.random.gamma(a, 1/b)` where `a, b` are the parameters as described in the homework information.

There was some problems in generating results from the particle Gibbs algorithm using the implementation. Some problem with the evolution of the b parameter for β makes it explode, and the sampling from the inverse Gamma distributions returns very large values on β and the algorithm breaks down. As stated in the instructions, the particle Gibbs and the PMH are consistent MCMC samplers, and should therefore return the same posterior distributions. I cannot seem to find the issue with my implementation

H.4

The annealing sequence was designed as

$$\pi_k(x) = \mathbf{1}_{0 \le x_1 \le 1} \mathbf{1}_{0 \le x_2 \le 1} \Big(\cos^2(\pi x_1) \sin^6(3\pi x_2) \exp(-30(x_1^2 + x_2^2)) \Big)^{k/K} \tag{28}$$

so that we can sample from $\pi_0(x) = \mathcal{U}(x : 0, 1)$ where both components x_1, x_2 are uniformly distributed between 0 and 1.

The presented results were generated with parameters set as $K = 80$, $N = 100$, 0.02 standard deviation for the Gaussian random walk and ESS triggered resampling at $N_{\text{thresh}} = 70$.

The particles at four different iterations are shown in figure 11, and the $N_{\text{eff}}(k)$ in 12. The estimate of normalizing constant was $\hat{Z} = 6.84$.

The average consecutive iterations between resampling was 10. For different seeds in the simulation, this number would vary within the range from 10 to 16.

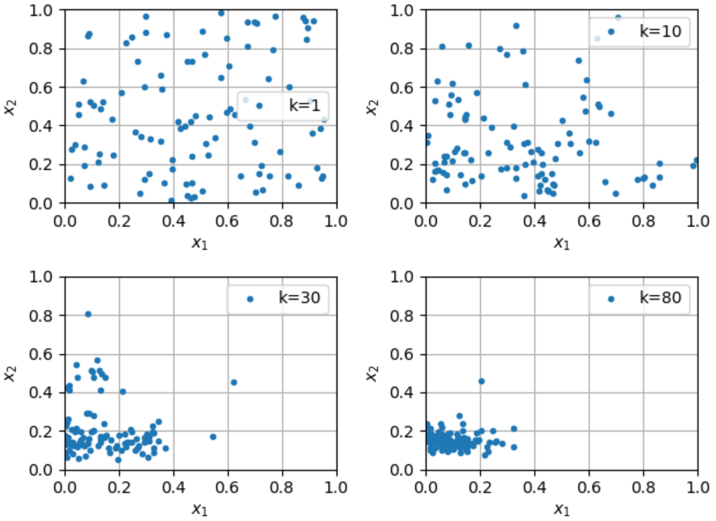

Figure 11: 4a. (Top left) The particles at initialization were sampled from the standard uniform distribution. (Top right) The particles after 10 iterations had all been moved into several clusters. (Bottom left) The particles after 30 iterations were a lot more concentrated on $x_2 = 0.15$ while more spread out for x_1, some sinusoidal effect is seen over x_2 having a cluster between 0.4 and 0.6. (Bottom right) After 80 iterations almost no effect of the sinusoidal functions were visible as they were more overshadowed by the exponential.

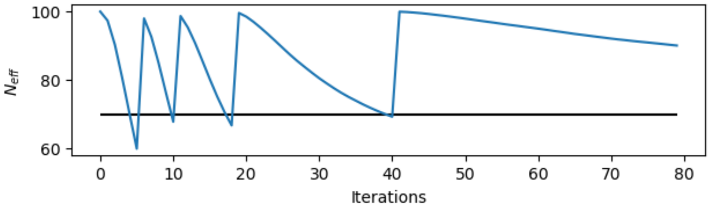

Figure 12: 4b. The number of iterations between resampling is increasing with the iteration number, while the average for these 150 iterations is 19.