HOMEWORK 5

Handed out: Wednesday, Oct. 25 2017 Due: Monday, Nov. 6 midnight

Solution 1. (a) Consider the stochastic volatility model:

$$x_t|x_{t-1} \sim \mathcal{N}\big(x_t|\phi x_{t-1}, \sigma^2\big)$$
$$y_t|x_t \sim \mathcal{N}\big(y_t|0, \beta^2 \exp(x_t)\big)$$

We use the the initial distribution $x_0 \sim \mathcal{N}\big(0, \sigma^2\big)$.

The bootstrap filtering algorithm is detailed as follows:

Q1a Generating N samples $x_{1:T}$

1: [Initialization] Generate $i = 1, \ldots, N$ samples $x_0^{(i)}$ from the initial model $\mathcal{N}(0, \sigma^2)$ and record the proposal density $q_0^{(i)}$.
2: [T=1] Generate $i = 1, \ldots, N$ samples $x_1^{(i)}$ from the model $x_1^{(i)} \sim \mathcal{N}(\phi x_0^{(i)}, \sigma^2)$;
3: Compute the unnormalized weight $w(x_1^{(i)})$ as $w(x_1^{(i)}) = p(y_1|x_1^{(i)})/q_0^{(i)}$.
4: Normalize the weights, and re-sample N samples $x_1^{(i)}$ according to the weights. Set the new weights to $1/N$.
5: **for** j = 2 to T **do**
6: **for** i = 1 to N **do**
7: Generate $x_j^{(i)} \sim \mathcal{N}(\phi x_{j-1}^{(i)}, \sigma^2)$;
8: Compute the unnormalized weight $w(x_j^{(i)})$ as $w(x_j^{(i)}) = p(y_j|x_j^{(i)})$.
9: **end for**
10: Normalize the weights, and re-sample N samples $x_j^{(i)}$ according to the weights. Set the new weights to $1/N$.
11: **end for**

(b) We can run the algorithm 10 times for different β, and then produce the box plot record. Apparently more sample can reduce the stochastic estimation variance, while more observation will create a more challenging problem.

$$\log \hat{p}(y_{1:T}) = \log\left(\prod_{t=1}^{T} \frac{1}{N}\sum_{i=1}^{N} p(y_t|x_t^{(i)})\right) = \sum_{t=1}^{T} \log\left(\frac{1}{N}\sum_{i=1}^{N} p(y_t|x_t^{(i)})\right)$$
$$= \sum_{t=1}^{T}\left(\log\big(\sum_{i=1}^{N} p(y_t|x_t^{(i)})\big) - \log N\right)$$

Figure 1: Q1 part (a). The mean of the filtering distribution $p(x_t|y_{1:t-1})$ at each time step vs the observations y_t.

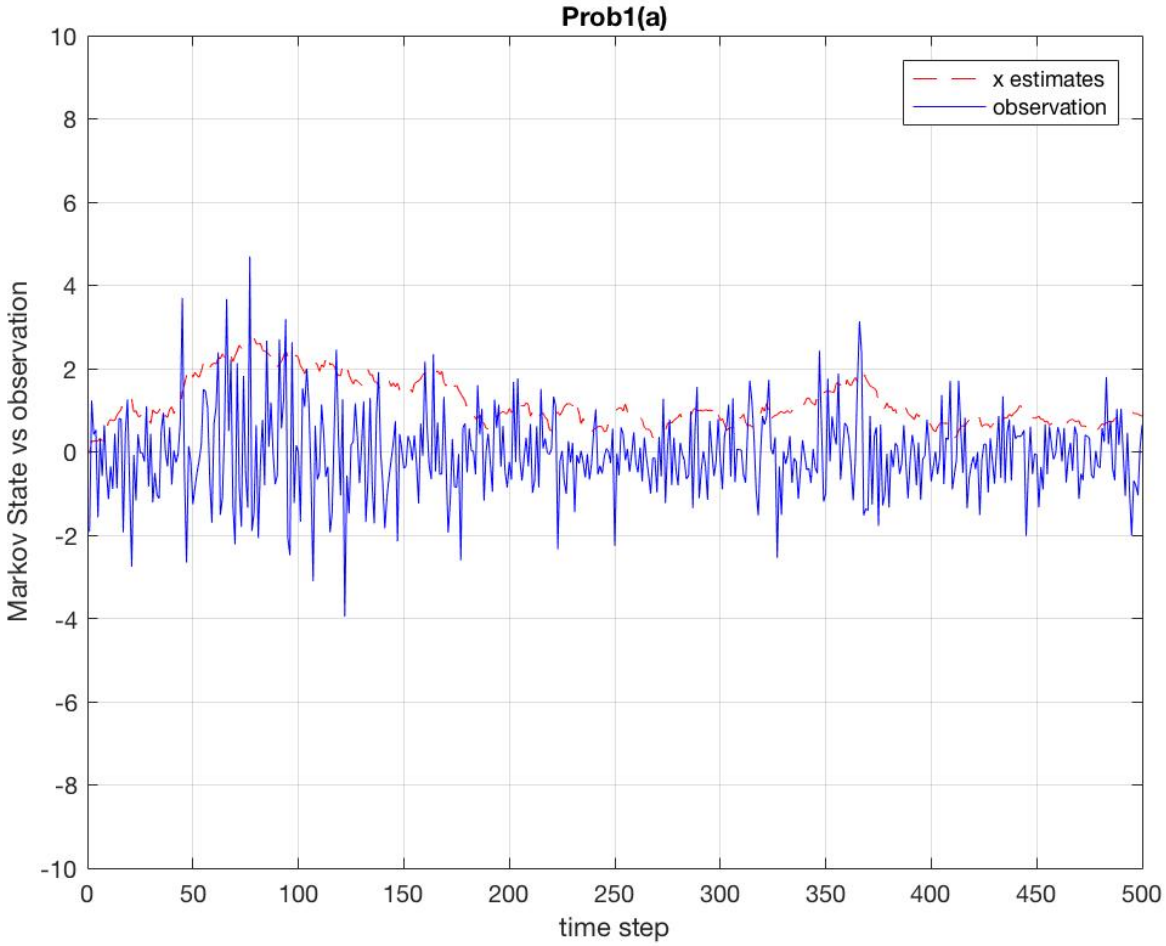


Figure 2: Q1 part (b): Likelihood box plot for $\beta \in [0, 2]$, with the combination of $N = T = 500$.

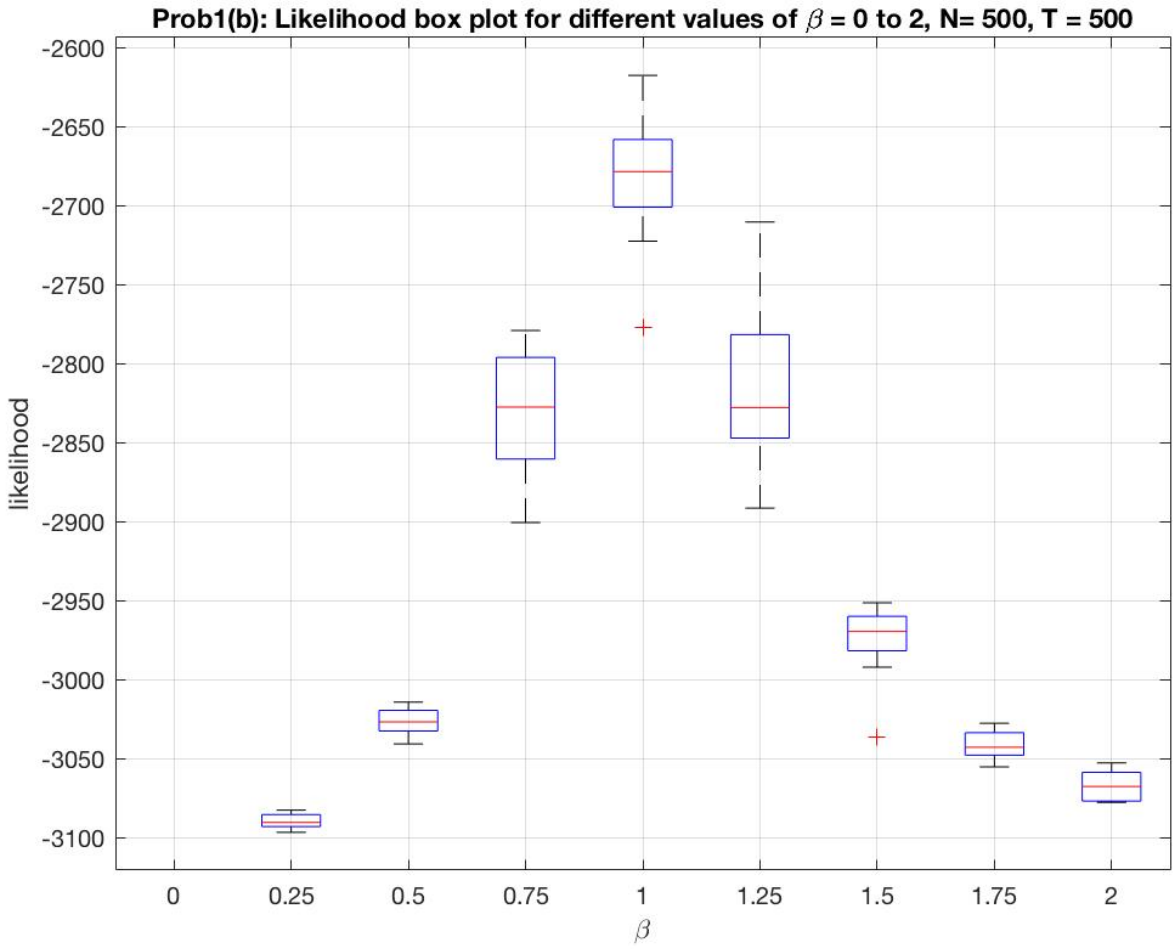


Figure 3: Q1 part (b): Likelihood box plot for $\beta \in [0, 2]$, with the combination of $N = 500, T = 100$.

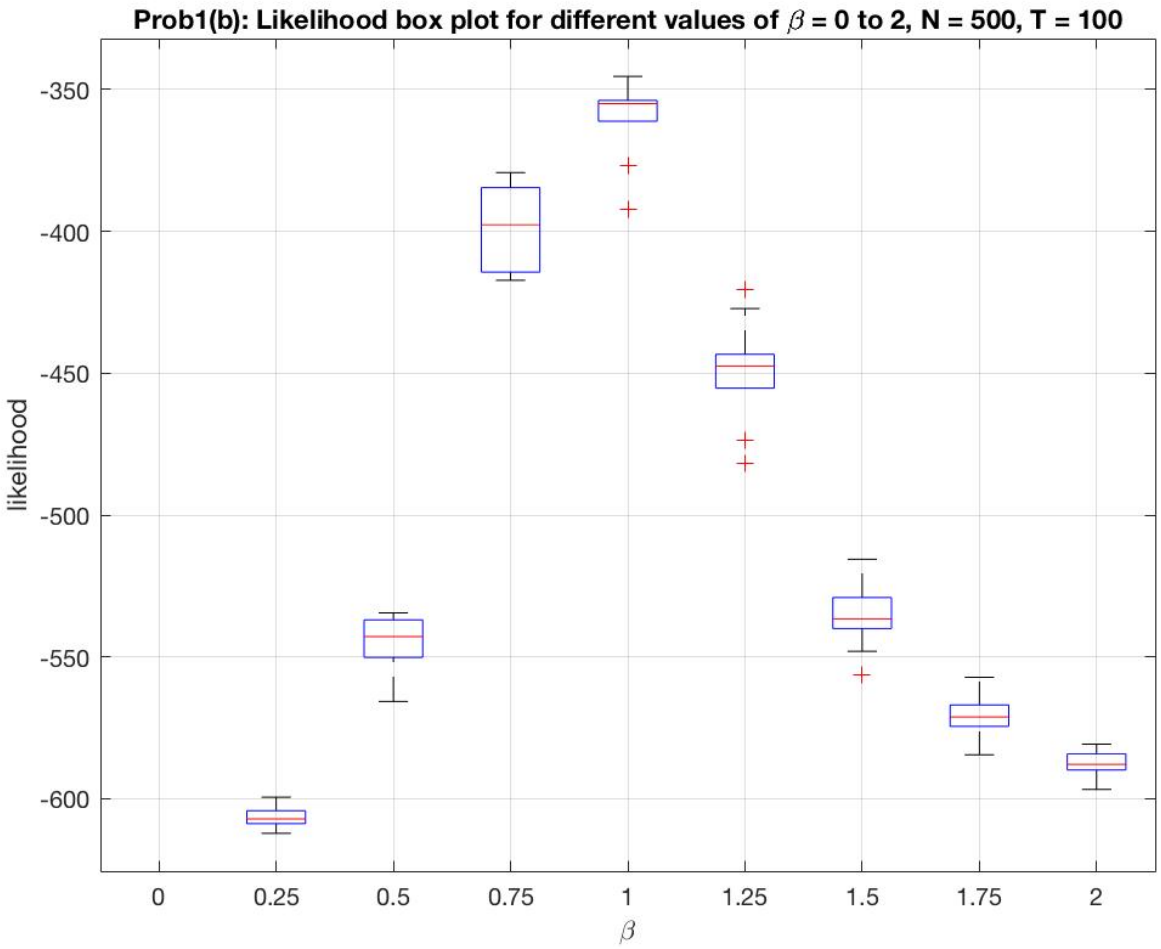


Figure 4: Q1 part (b): Likelihood box plot for $\beta \in [0, 2]$, with the combination of $N = 100, T = 500$.

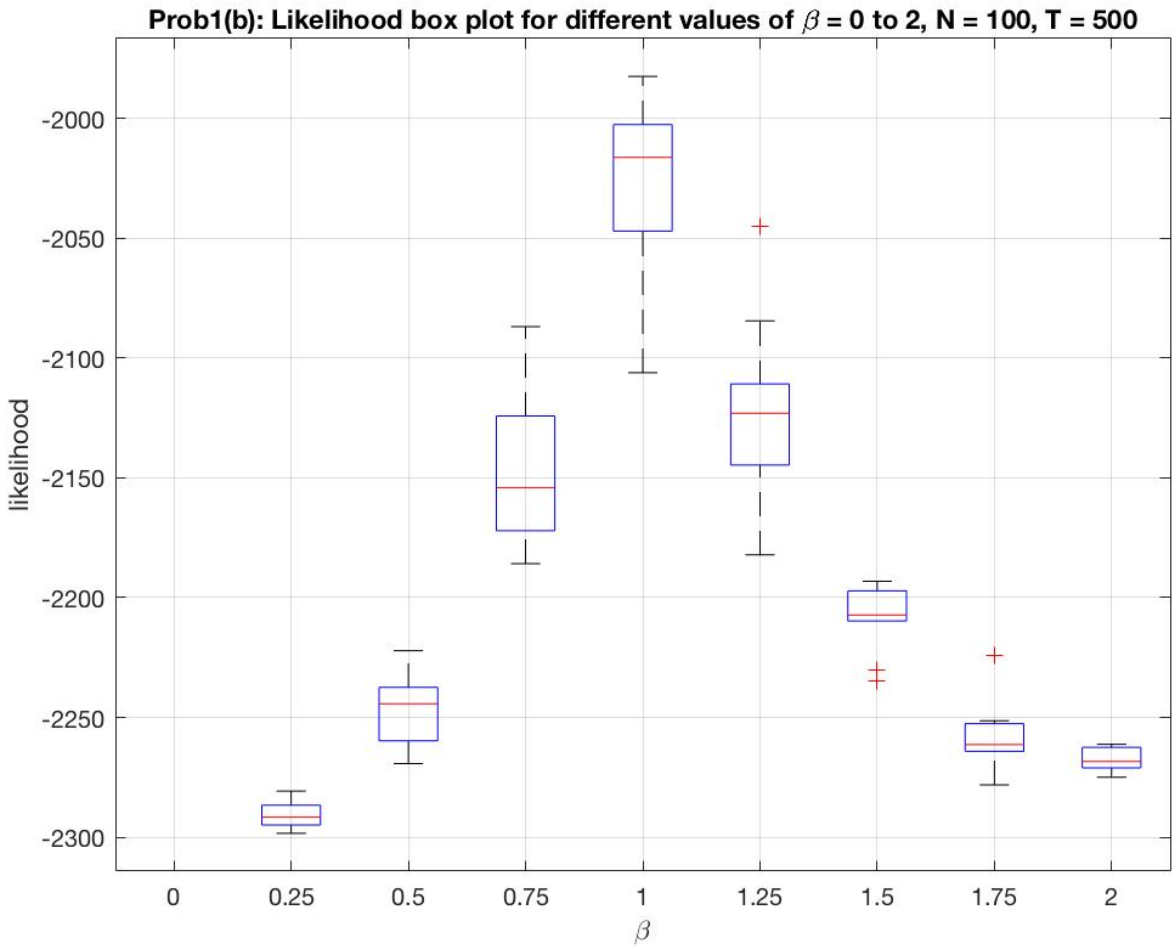
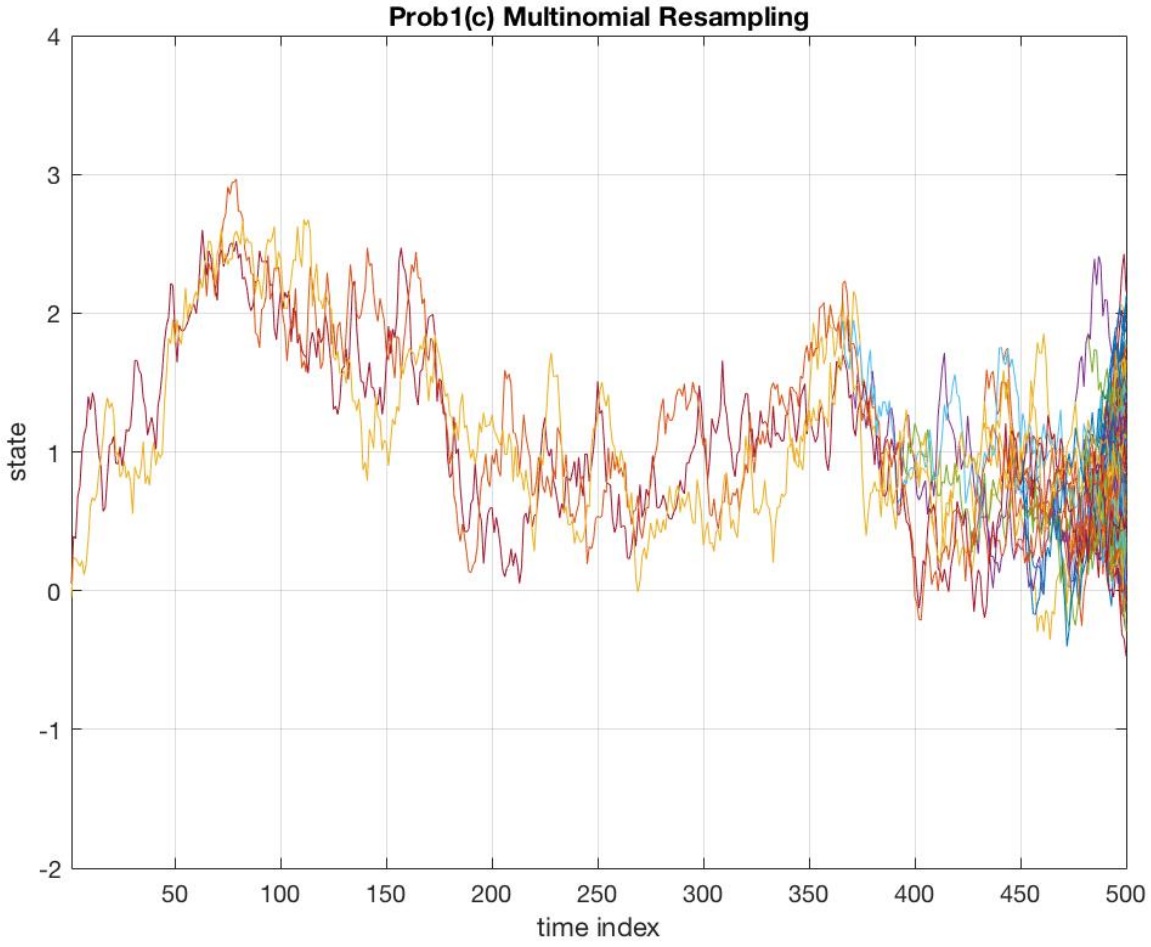


Figure 5: Q1 part (c): Path trajectories with multinomial resampling with no ESS trigger. No-resampled trajectories are removed



(c) The occurrence of degeneracy can be observed from the path trajectories plots, as only a few states in x_0 is retained. Based on this problem, systematic re-sampling works better than multinomial. Degeneracy will be alleviated with the ESS trigger. In addition, systematic re-sampling seem to work better the multinomial re-sampling.

Solution 2. We place the first point at (0,0), then sequentially grow the chain. For any position, there are four neighbor candidate points. Under the sequential importance sampling approach, for both cases (monotonic or self-avoiding), we rule out the candidate points if they are occupied by the chain already, or if they are out of the

Figure 6: Q1 part (c): Path trajectories with systematic resampling with no ESS trigger. No-resampled trajectories are removed

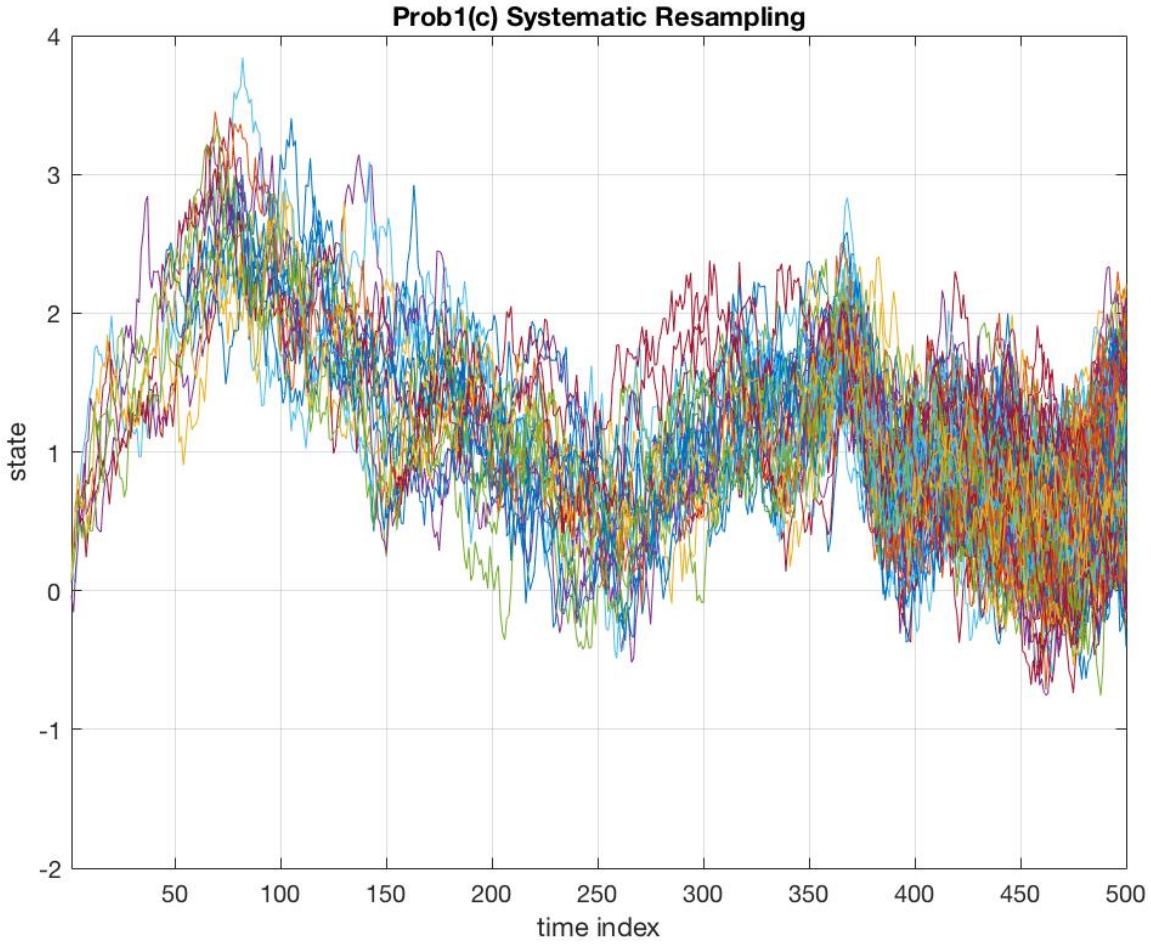


Figure 7: Q1 part (c): Path trajectories with multinomial resampling with ESS trigger = 50. No-resampled trajectories are removed

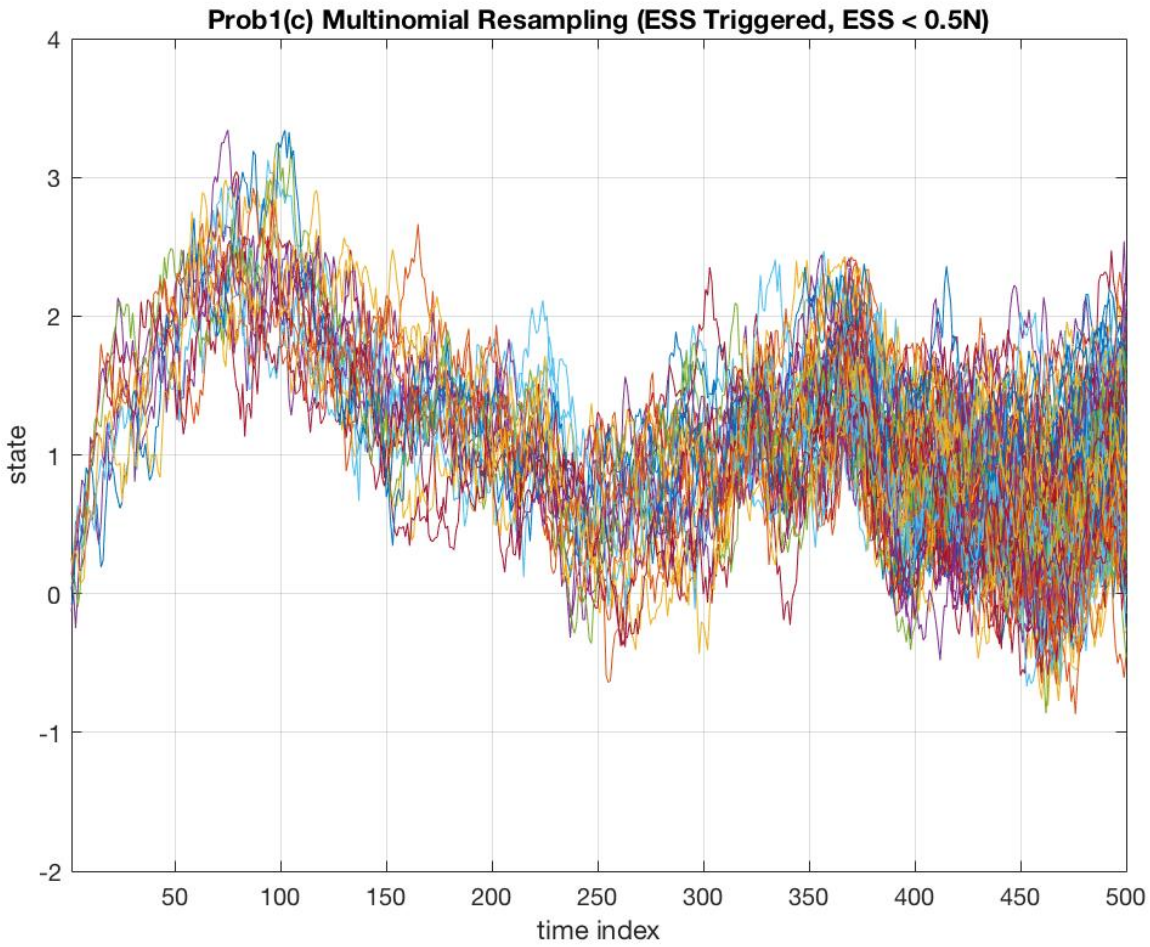
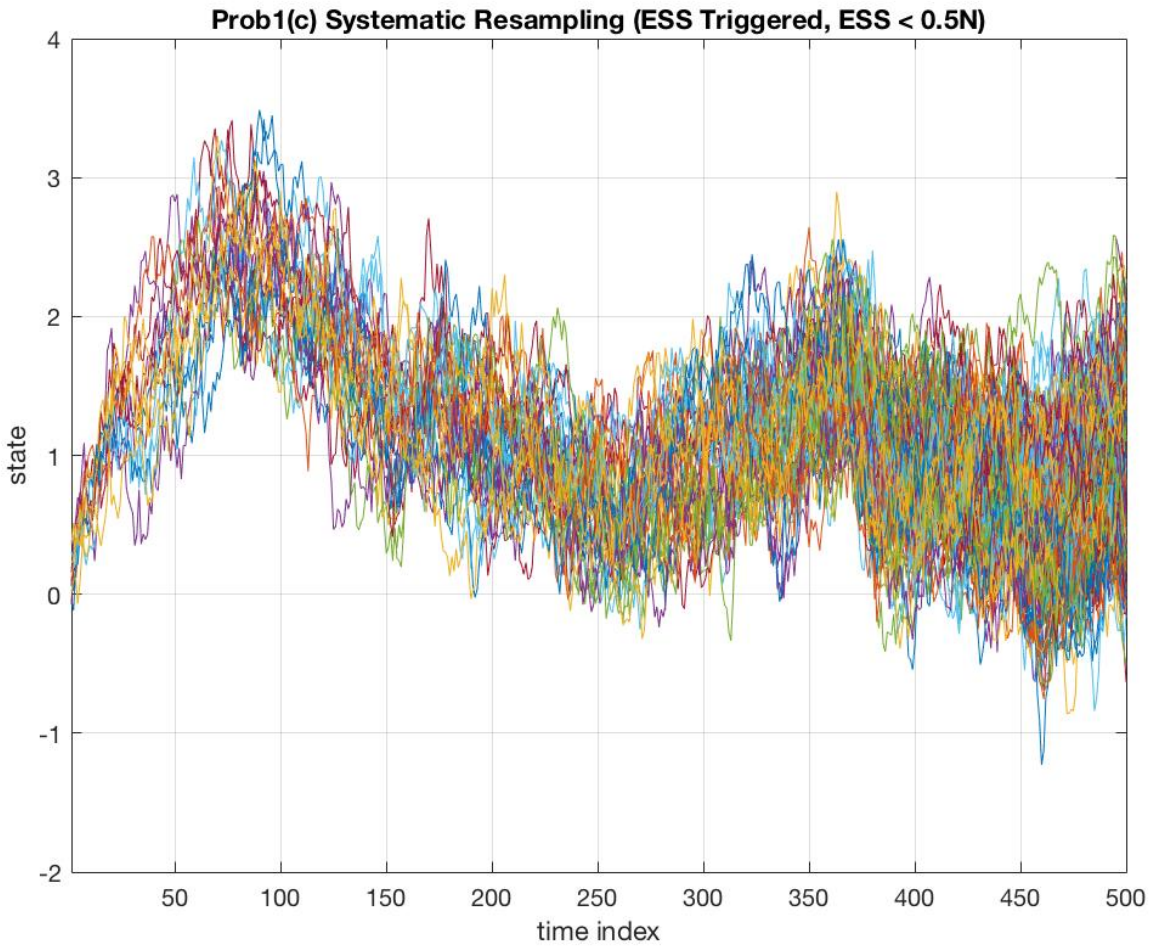


Figure 8: Q1 part (c):
Path trajectories with systematic resampling with ESS trigger = 50. No-resampled trajectories are removed



$(n+1) \times (n+1)$ lattice. Additionally for monotonic case, we rule out the candidate points if they are in the lower or left direction. Details can be referred to the algorithm Q2. Since we don't normalize the weight, the weight itself can be considered as a sample estimate of total numbers of chains.

The total number of chains can be roughly estimated as:

$$\hat{N}_{tot} = \frac{1}{n} \sum_{i=1}^{n} w_i$$

where the w_i is the un-normalized weight of a successfully generated chain. In this case every occupied, out-of-bounds, or non-monotonic candidate locations have to be ruled out.

Using 1000 samples, the average number of chains is estimated as 183275 with a sample standard deviation estimation error as 5723. It's close to the true answer $C(20, 10) = 184756$.

(b) For the second part, the total number of chains can be calculated similarly to the previous case, while the average length can be calculated as the weighted mean of the generated chains' length:

$$\hat{N}_{tot} = \frac{\frac{1}{n} \sum_{i=1}^{n} l_i \cdot w_i}{\frac{1}{n} \sum_{i=1}^{n} w_i}$$

Hint: the ratio between the target density ($\propto 1$ for every self-avoiding sample path) and the importance sampling proposal density (consider our choice in selecting the next location) can be shown to be $\propto w$.

Using 1000 samples, the average number of chains is estimated as 6.27×10^{24} with a sample standard deviation estimation error as 3.61×10^{24}. It is acceptable comparing to the true value 1.6×10^{24}. The average length is estimated as 95.7 with a sample standard deviation estimation error as 57. It is also close to the true value 91.9.

(c) For the third part, to enhance the efficiency one would be considering more efficient proposal densities. For example, in the first case, suppose the current location is (a, b). Instead of going to right and up for equal probabilities (1/2), one could set the probability to go up as $(n-a)/(n-a+n-b)$, and to go right as $(n-b)/(n-a+n-b)$, since we know the final destination is (n, n).

Q2 Steps to grow a chain and record un-normalized weights

Initialization: Chain = (0,0), w = 1.
while Chain(end,:) $\neq (n, n)$ **do**
  Identify the four candidate moves from current position Chain(end,:);
  Rule out ineligible candidates, and denote the number of remaining candidate n_c;
  if $n_c > 0$ **then**
    $w = w \cdot n_c$;
  else
    reject the sample
  end if
end while

Solution 3. We wish to solve the Fredholm equation at $x = 0$

$$f(x) = x + \frac{1}{2}\int_{-1}^{1}(t-x)f(t)\mathrm{d}t \tag{1}$$

which can be transformed into solving

$$f(x) = x + \sum_{n=1}^{\infty}\int_{-1}^{1}\left(\prod_{k=1}^{n}(t_k - t_{k-1})\right)t_k\mathrm{d}t_{1:n} \tag{2}$$

Computing Eq. (2) is challenging as it involves an infinite sum of integrals of increasing dimension. A sequential importance sampling strategy arises as a natural approach to this problem.

Consider a Markov chain starting with $x_0 = 0$ and a transition kernel $M(t, t')$ which gives the probability of moving to state t' when the current state is t. We select μ and M such that the support of μ includes $[-1, 1]$ and $M(t, t') > 0$ if $t \neq t'$. Moreover, M is chosen to have an absorbing/cemetery state $s \notin [-1, 1]$, i.e., for any $t \in [-1, 1]$ $M(t, s) = P_d > 0$.

The algorithm to solve for $\hat{f}(x)$ is detailed as follows. M can be set to the uniform distribution $U[-1-\delta, 1+\delta]$, leading to a $P_d = \delta$, with $M(.,.) = 1/(2+2\delta)$. We tried 10^6 samples with different choices of δ from 0.001 to 2. The bigger δ is, the easier the chain goes to cemetery, and hence the shorter path we obtain. $\delta = 0.001$ leads us to a estimate $\hat{f}(x_0 = 0) = 0.269$, while $\delta = 1$ leads us to $\hat{f}(x_0 = 0) = 0.146$. Comparing with the true value 0.25, it's clear that smaller δ is more suitable.

The algorithm immediately follows:

Q3 Steps to approximately solve $f(x)$

1: **for** i=1 to N **do**

2: Simulate a path using Markov chain. Start from $x^{(i)} = 0$, then generate sample $t_1^{(i)} \sim M(x^{(i)}, t)$, $t_2^{(i)} \sim M(t_1^{(i)}, t_2)$, until $t_{k+1}^{(i)}$ reaches s, the cemetery state.

3: Calculate the associated weight

$$W^{(i)}(x, t_1, t_2, \ldots, t_k) = \begin{cases} \frac{1/2}{M(x,t_1)} \left(\prod_{k=1}^{n} \frac{1/2}{M(t_k, t_{k-1})} \right) t_k / P_d, k > 0 \\ x/P_d, k = 0 \end{cases}$$

4: **end for**

Solution 4. (a) Re-organize the model as

$$\begin{aligned} X_t | X_{t-1} &\sim \mathcal{N}(0.7 X_{t-1}, 0.1) \\ Y_t | X_t &\sim \mathcal{N}(0.5 X_t, 0.1) \end{aligned}$$

The algorithm to generate synthetic measurements $y_{1:T=2000}$ is detailed as follows:

Q4a Generating N tracks of synthetic measurements

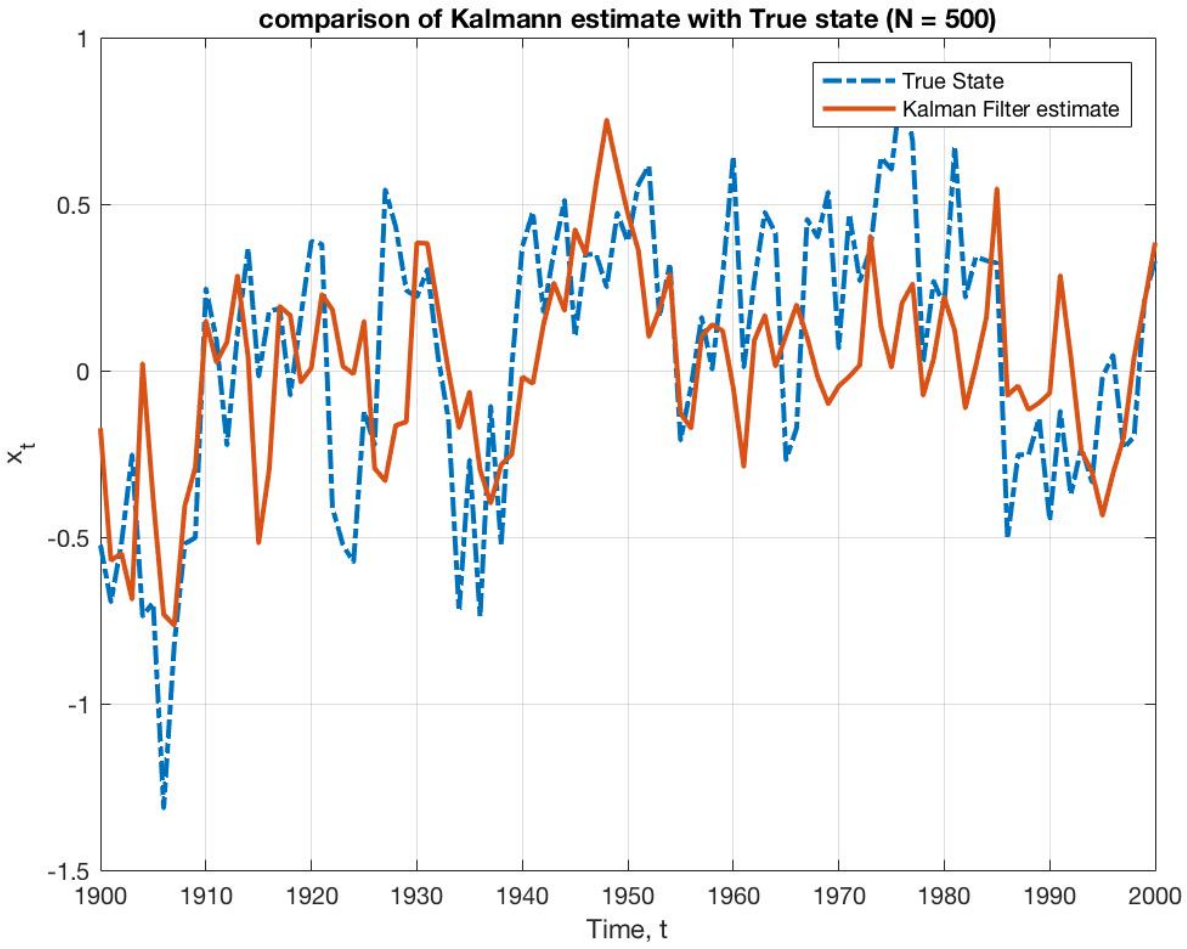
for i = 1 to N **do**
  Generate $x_0^{(i)}$ from the initial model $\mathcal{N}(0,1)$.
  for j = 1 to T **do**
    Generate $x_j^{(i)} \sim \mathcal{N}(0.7x_{j-1}^{(i)}, 0.1)$
    Generate $y_j^{(i)} \sim \mathcal{N}(0.5x_j^{(i)}, 0.1)$
  end for
end for

(b) To apply the Kalman Filter model, notice first $A = 0.7, Q = 0.1, C = 0.5, E = 0.1, P_0 = 1$. The algorithm then goes as:

Q4b Kalman filtering

$\hat{x}_0 = 0, \hat{P}_{0|0} = 1.$
for j = 1 to T **do**
  $\hat{P}_{j|j-1} = A\hat{P}_{j-1|j-1}Q^T + Q$
  $K_j = \hat{P}_{j|j-1}C^T(C\hat{P}_{j|j-1}C^T + R)^{-1}$
  Compute $\hat{x}_{j|j} = A\hat{x}_{j-1|j-1} + K_j(y_j - A\hat{x}_{j-1|j-1})$
  $\hat{P}_{j|j} = \hat{P}_{j|j-1} - K_t\hat{P}_{j|j-1}$
  $x_t \sim \mathcal{N}(\hat{x}_{j|j}, \hat{P}_{j|j})$
end for

Figure 9: Q4 part b: the true states trajectories vs. the Kalman estimated mean



Kalman filtering is the optimal method in current setting so we can't expect particle filter to beat it.

(c) The bootstrap filtering algorithm is detailed as follows. Note that the main difference between it and the one in Q1 is that we have been given an initial distribution, so no proposal is needed for $t = 0$. We increase the sample set from 100 to 500 and eventually 1000. The error in mean decreases from 0.035 to 0.0168 to 0.0118. The error in variance decreases from 0.0140 to 0.0650 to 0.0050. The general decreasing trend is obvious.

Q4c Generating N samples $x_{1:T}$

1: [Initialization] Generate $i = 1, \ldots, N$ samples $x_0^{(i)}$ from the initial model $\mathcal{N}(0, 1)$.
2: **for** j = 1 to T **do**
3: **for** i = 1 to N **do**
4: Generate $x_j^{(i)} \sim \mathcal{N}(\phi x_{j-1}^{(i)}, \sigma^2)$;
5: Compute the unnormalized weight $w(x_j^{(i)})$ as $p(y_j|x_j^{(i)})$.
6: **end for**
7: Normalize the weights, and re-sample N samples $x_j^{(i)}$ according to the weights. Set the new weights to $1/N$.
8: **end for**

(d) Fully adaptive particle filters requires using the locally optimal proposal distribution (i.e., optimal for the current step):

$$q_{opt}(x_t|x_{1:t-1}, y_{1:t}) = p(x_t|x_{t-1}, y_t) \propto p(y_t|x_t)p(x_t|x_{t-1})\mathcal{N}(0.7x_{t-1}, 0.1)\mathcal{N}(2y_t, 0.4)$$

Multiplying the two normal distributions together, the result is also a normal distribution, with mean

$$\mu_t = (0.4 \times 0.7x_{t-1} + 0.1 \times 2y_t)/(0.1 + 0.4) = 0.56x_{t-1} + 0.4y_t$$
$$\sigma_t^2 = 0.1 \times 0.4/(0.1 + 0.4) = \frac{2}{25}$$

Q4d Generating N samples $x_{1:T}$

1: [Initialization] Generate $i = 1, \ldots, N$ samples $x_0^{(i)}$ from the initial model $\mathcal{N}(0, 1)$.
2: **for** j = 1 to T **do**
3: **for** i = 1 to N **do**
4: Generate $x_j^{(i)} \sim \mathcal{N}(\mu_t, \sigma_t^2)$;
5: Compute the unnormalized weight $w(x_j^{(i)})$ as $p(y_j|x_j^{(i)})p(x_j^{(i)}|x_j^{(i-1)})/\mathcal{N}(\mu_t, \sigma_t^2)$.
6: **end for**
7: Normalize the weights, and re-sample N samples $x_j^{(i)}$ according to the weights. Set the new weights to $1/N$.
8: **end for**

We again increase the sample set from 100 to 500 and eventually 1000. the error in mean decreases from 0.0339 to 0.0156 to 0.0108, and the error in variance decreases from 0.0142 to 0.0650 to 0.0048, Comparing, respectively, to the bootstrap one, for which the error in mean decreases from 0.035 to 0.0168 to 0.0118, and the error in variance decreases from 0.0140 to 0.0650 to 0.0050, we can conclude the adapted PF works slightly better.

(e) We zoom in the traces to [1950-2000] for a better resolution of path degeneracy. It clearly happens.

(f) Systematic resampling clearly reduces the degeneracy issue.

(g) From the plots, we can observe that the degeneracy get alleviated with the addition of ESS triggers.

Figure 10: Q4 part e: fully adapted PF with multinomial resampling

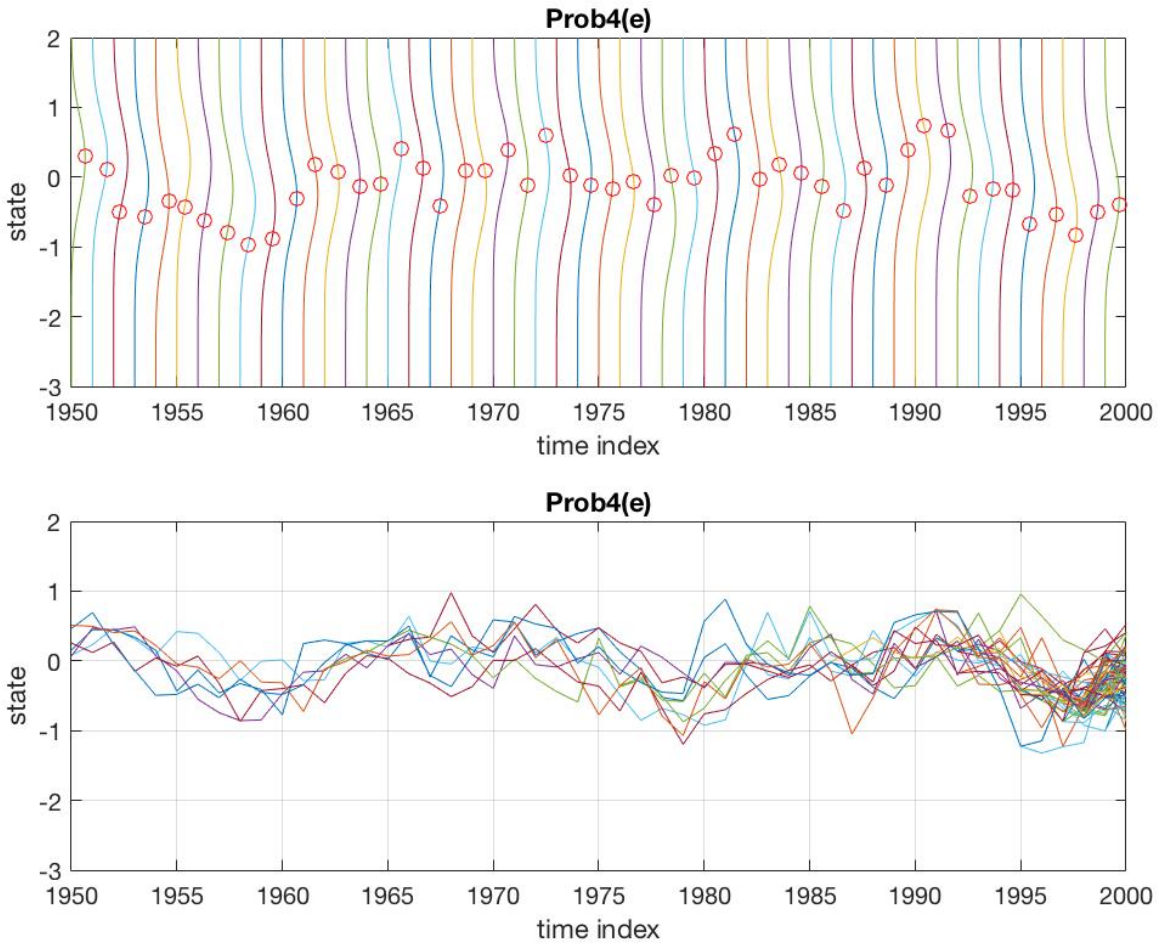


Figure 11: Q4 part f: fully adapted PF with systematic resampling

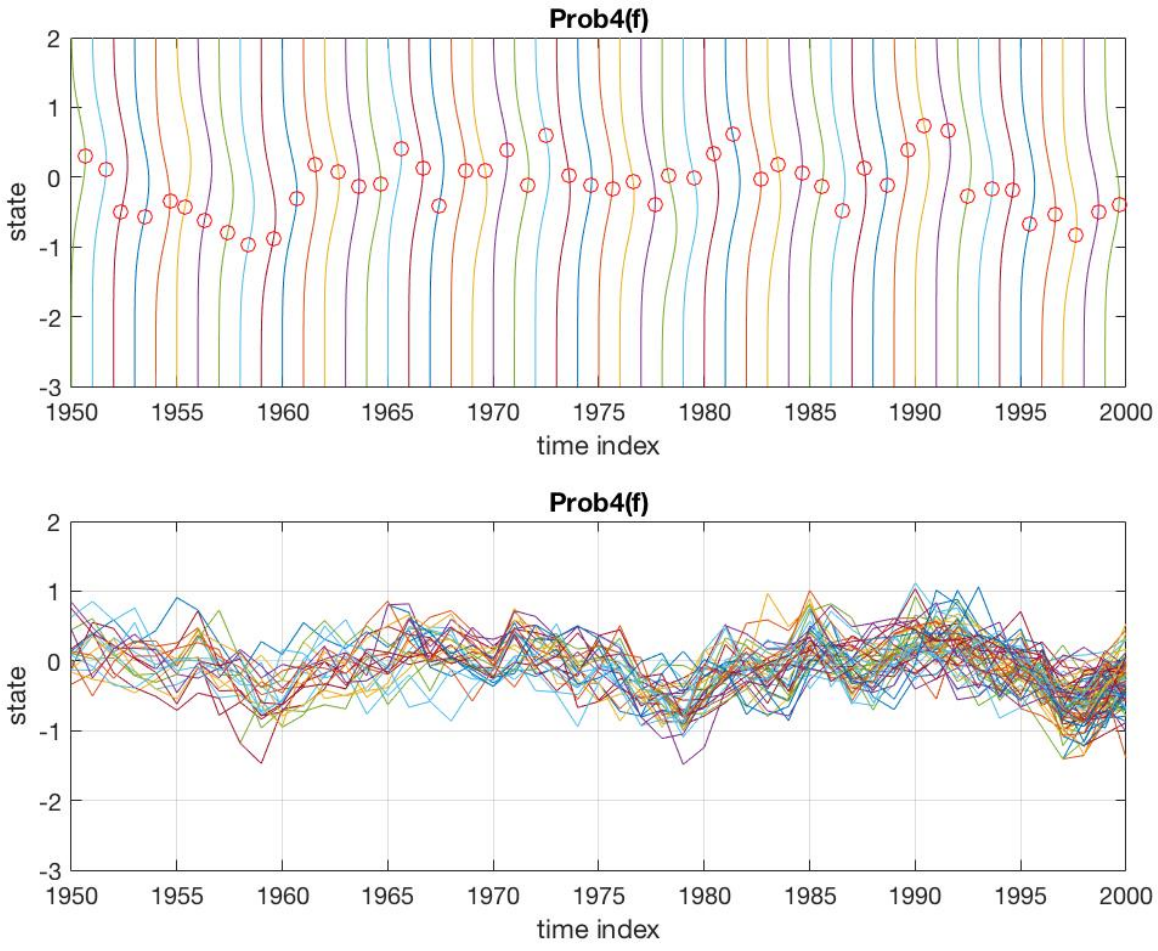


Figure 12: Q4 part g: fully adapted PF with ESS triggers