HA1 - SF2955

Group 2
Magnus Axén maaxen@kth.se 980915-7119
Henrik Fagerlund hfag@kth.se 991012-2655

April 28, 2022

Problem 1

The project is based on the following model,

$$\mathbf{X}_{n+1} = \Phi_{\mathbf{X_n}} + \Psi_z \mathbf{Z}_n + \mathbf{\Psi}_w \mathbf{W}_{n+1}, \quad n \in \mathbb{N}. \tag{1}$$

We need to deduce if $\{\mathbf{X_n}\}_{n\in N}$ and $\tilde{\mathbf{X}}_\mathbf{n} = (\mathbf{X}_n^t, \mathbf{Z}_n^t)^t$; are Markov chains. A Markov chain is characterised by only depending on the previous state, i.e.,

$$\mathbb{P}\left(X_{n+1} \in B \mid X_0, \ldots, X_n\right) = Q_n\left(X_n; B\right).$$

Where Q_n is the transition rate matrix, and B is some value s.t. $B \subseteq X_{N+1}$.

As for $\{\mathbf{X_n}\}_{n\in N}$, we can tell it depends on $\mathbf{X_n}, \mathbf{Z_n}$, and $\mathbf{W}_{n+1}$. $\mathbf{W}_{n+1}$ is noise variables and would therefore not make an impact in determining whether it is a Markov chain or not. However, seeing as we need $\mathbf{X}_{n+1}$ to only depend on $\mathbf{X}_n$, we can conclude it is **not a Markov chain** because of the dependence on $\mathbf{Z_n}$.

For $\tilde{\mathbf{X}}_\mathbf{n}$, the $\mathbf{Z}_n^t$ is now included in the matrix, which is solely dependant on $\mathbf{Z}_{\mathbf{n-1}}^\mathbf{t}$. Thus, the vector $(\mathbf{X}_n^t, \mathbf{Z}_n^t)^t$ is only dependant on its former states, $\mathbf{Z}_{n-1}, \mathbf{X}_{n-1}$, and the noise $\mathbf{W}_{n+1}$, making it a **Markov chain**.

Presented below is a randomised trajectory of the moving target, $\left\{\left(X_n^1, X_n^2\right)\right\}_{n=0}^m$, using the rng-seed: $rng(19)$.

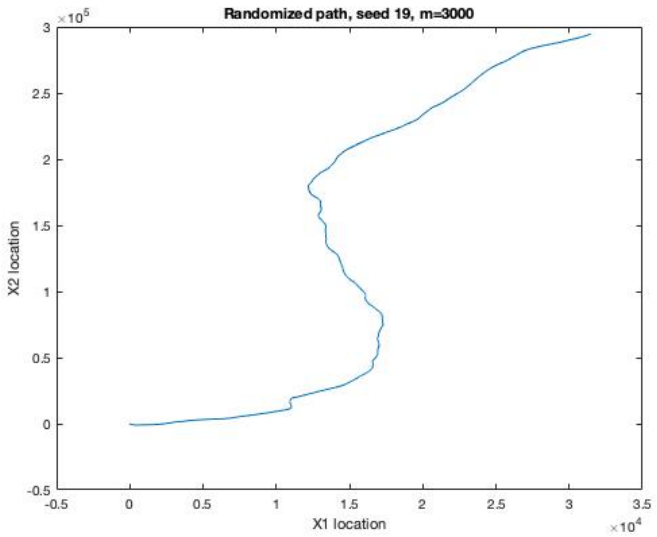


Figure 1: Plot for equation (1) with the constants in lab

Problem 2

The pair $\left\{\left(\tilde{\mathbf{X}}_n, \mathbf{Y}_n\right)\right\}_{n\in\mathbb{N}}$ is a hidden Markov model (HMM) if the following statement is fulfilled,

- $\tilde{\mathbf{X}_\mathbf{n}}$ is a Markov process which is not directly observable ("hidden"), i.e. we do not know the position of the moving target.
- $\mathrm{P}\left(Y_m \in A \mid \tilde{X}_1 = x_1, \ldots, \tilde{X}_m = x_m\right) = \mathbf{P}\left(Y_m \in A \mid \tilde{X}_m = x_m\right)$

This must be fulfilled for the components of any vector $(Y_0, Y_1, Y_2, ..., Y_n)$. Which is the same, as explained in the lectures, that the conditional distribution of each Y_m only depends on X_m.

From the given model,

$$Y_n^\ell = v - 10\eta \log_{10} \left\| \left(X_n^1, X_n^2\right)^\top - \pi_\ell \right\| + V_n^\ell,$$

it can be seen that it only depends on (X_n^1, X_n^2). $\left\{\left(\tilde{\mathbf{X}}_n, \mathbf{Y}_n\right)\right\}_{n\in\mathbb{N}}$ is therefore said to form a HMM.

The next task is to find the transition density,

$$p(\mathbf{y}_n|\hat{\mathbf{x}_n}) \quad \text{of} \quad \mathbf{Y_n}|\hat{\mathbf{X}}_n.$$

We need note some things. Firstly, the basis stations are independent of one another. Secondly, seeing as

$$Y_n^\ell = v - 10\eta \log_{10} \left\| \left(X_n^1, X_n^2\right)^\top - \pi_\ell \right\| + V_n^\ell,$$

we note that $y|x$ is a linear combination of constants and Gaussian variables. (Gaussian coming from the noise $V_n^\ell \sim N(0, \varsigma^2)$). This implies that the result must be Gaussian as well.

By the Independence property we can write the joint density as,

$$p(\mathbf{Y_n}|\hat{\mathbf{x}_n}) = \prod_{\ell=1}^{6} p(\mathbf{y}_n^\ell|\hat{\mathbf{x}_n}).$$

Lastly, note that for the values $\mathbf{y}_n^\ell|\hat{\mathbf{x}_n}$, we have,

$$\mathrm{E}[\mathbf{y}_n^\ell|\hat{\mathbf{x}_n}] = \mathrm{E}\left[v - 10\eta \log_{10} \left\| \left(X_n^1, X_n^2\right)^\top - \pi_\ell \right\| + V_n^\ell\right] =$$

$$\mathrm{E}\left[v - 10\eta \log_{10} \left\| \left(X_n^1, X_n^2\right)^\top - \pi_\ell \right\|\right] = v - 10\eta \log_{10} \left\| \left(X_n^1, X_n^2\right)^\top - \pi_\ell \right\|$$

Since the variance is invariant with respect to location parameter, we have that,

$$\mathrm{Var}[\mathbf{y}_n^\ell|\hat{\mathbf{x}_n}] = \mathrm{Var}\left[v - 10\eta \log_{10} \left\| \left(X_n^1, X_n^2\right)^\top - \pi_\ell \right\| + V_n^\ell\right] =$$

$$\text{Var}\left[V_n^\ell\right] = \varsigma^2.$$

Finally, we obtain the transition density,

$$\begin{aligned} p(\mathbf{Y_n}|\hat{\mathbf{x}_n}) &= \prod_{\ell=1}^{6} p(\mathbf{y}_n^\ell|\hat{\mathbf{x}_n}) \\ &= \prod_{\ell=1}^{6} \frac{1}{\sqrt{2\pi\varsigma^2}} e^{-\frac{1}{2}\left(\frac{y_\ell - v - 10\eta\log_{10}\left\|\left(X_n^1, X_n^2\right)^\top - \pi_\ell\right\|}{\varsigma}\right)^2} \\ &= \frac{1}{(2\pi\varsigma^2)^3} e^{-\frac{1}{2}\left(\frac{\sum_{l=1}^{6} y_\ell - \sum_{l=1}^{6} v - 10\eta\log_{10}\left\|\left(X_n^1, X_n^2\right)^\top - \pi_\ell\right\|}{\varsigma}\right)^2}. \end{aligned}$$

Problem 3

In problem 3 we implement the SIS-algorithm. This is done similar to how it's presented in the lectures. To combat some problems with the SIS method we decided to normalise the weights and use log to see it's behaviour with these adjustments, as per instruction.

Firstly using regular SIS.

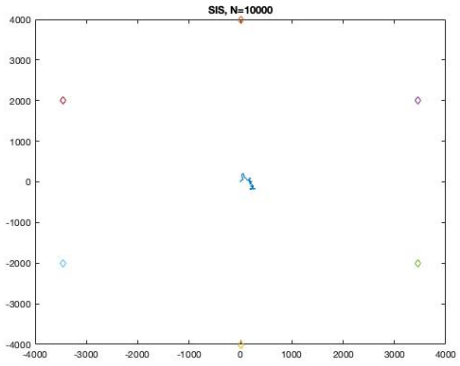
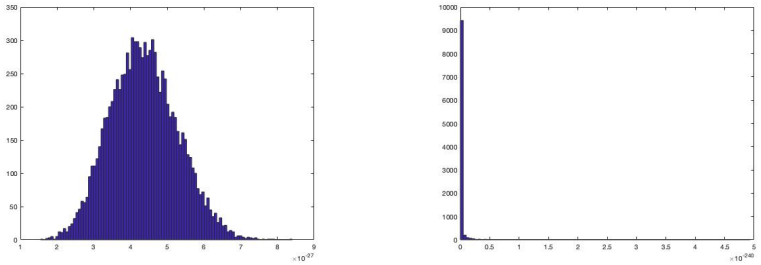


Figure 2: SIS-algorithm

Visually from the plot it is clear that there is not a lot of movement, this is simply due to the fact that the weights rapidly go to 0, see figure 3.



(a) n=5 (b) n=50

Figure 3: Histogram of weights

Moreover, at the last instance before the models explodes (n=70) the efficient sample size went to 0. At the first instance (n=20) we have roughly 3250 points, at (n=40) we had roughly 539, and at (N=60) we had 72. This shows just how quickly the computational error of the algorithm emerge.

Where the efficient sample size is computed by,

$$ESS_m = \frac{1}{\sum_{i=1}^{N}(\frac{\omega_m^i}{\Omega_m})^2}.$$

Now we shall instead use the maximum log weight and after the log transformation revert it back, as seen in the pseudo-code below.

Require: log-weights $\left(\ell_n^i\right)_{i=1}^N$ corresponding to the natural logarithms of $\left(\omega_n^i\right)_{i=1}^N$.

1. Determine the maximum log-weight: $L \leftarrow \max_{i=1,\ldots,N} \ell_n^i$;
2. **for** $i = 1 \rightarrow N$ **do**
3. $\quad \bar{\omega}_n^i \leftarrow \exp\left(\ell_n^i - L\right)$;
4. **end for**
5. **for** $i = 1 \rightarrow N$ **do**
6. $\quad$ Normalize: $W_n^i \leftarrow \frac{\bar{\omega}_n^i}{\sum_{j=1}^N \bar{\omega}_n^j}$;
7. **end for**
8. **return** Normalized weights $\left(W_n^i\right)_{i=1}^N$ corresponding to $\left(\frac{\omega_n^i}{\Omega_n}\right)_{i=1}^N$.

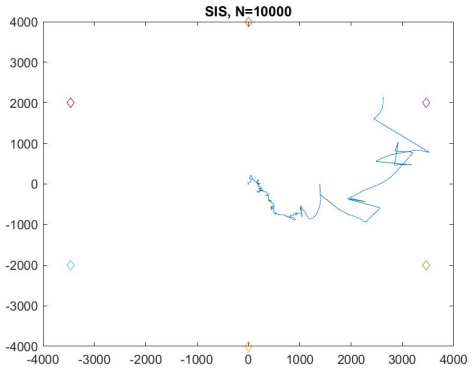
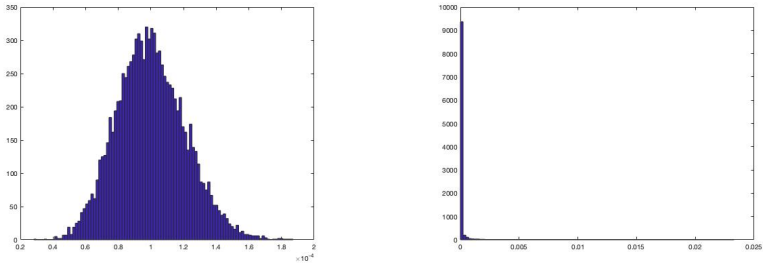


Figure 4: SIS-algorithm with log normalised weights

The advantageous portion here is that the efficient sample size more slowly goes to 0 after n=70. Now, at $n = 70$ we have a sample size of 55, whilst for $n < 70$ the sample sizes between the two methods are rather similar. Moreover, instead of all the weights vanishing, it instead has an efficient sample size of 1 as n grows more closely to 500, which could be seen in the plot when it is less sporadic and has more straight lines. Also in the histogram, in the right plot of figure 5 it is difficult to decipher but looking at the x-axis we can tell it has values at 0.025, indicative of weights converging slower to 0. Lastly, instead of giving rise to the numerical problem as seen in the first plot, we now still have a trajectory that continues.



(a) n=5 (b) n=50

Figure 5: Histogram for log/normalised weights

Problem 4

In problem 4 we implement the SISR algorithm as in the lectures.

Most noticeable is that running the algorithm multiple times gave much more precise and similar results than the SIS method, i.e. less variance. This is since the variance is increasing exponentially fast for the SIS method, seeing as the weight is multiplied iteratively with itself. Whilst instead in the SISR we re-sample the weights in such a way where we prioritise the larger weights rather than the smaller ones, and remove the multiplicative factor of the algorithm, which exists in the SIS method.

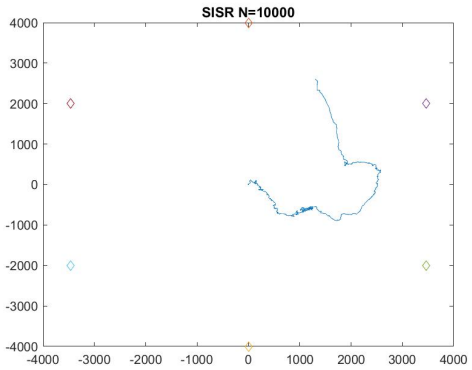


Figure 6: SISR algorithm

When discussing the most probable driving command, we take into consideration the weights of the selective states, and check which probability they cover combined.

$$Z_m^* = \arg\max_{c \in \mathbb{C}} \quad p(z_m = c | Y_{0:m} = y_{0:m}), \tag{2}$$

where $\mathbb{C}$ is the class of driving commands, i.e

$$\mathbb{C} = \{\{0,0\}\{3.5,0\}\{-3.5,0\}\{0,3.5\}\{0,-3.5\}\}.$$

Moreover, (2) means we want to find the largest sum of weights corresponding to that c, i.e.

$$\arg\max_{\forall c} \sum_{i=1}^{N} \frac{\omega_i^\ell}{\Omega^\ell} \mathbb{1}_{z_i = c}.$$

We sum all the weights for the corresponding command, and take the maximizing sum of weights.

We analysed the most common driving command by plotting the histogram of the different commands. No great disparate is seen, i.e all commands seem to occur equally frequently with a fifth probability.

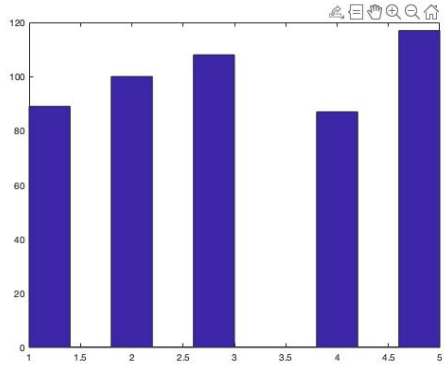


Figure 7: Histogram from

Even if it is the case that there exist some pattern in the way the commands occurred, it can not be seen. This is most likely due to the fact that, in equation (1) the term $\Psi_z \mathbf{Z}_n$ plays a very small role in how much of an impact it has on the next state.

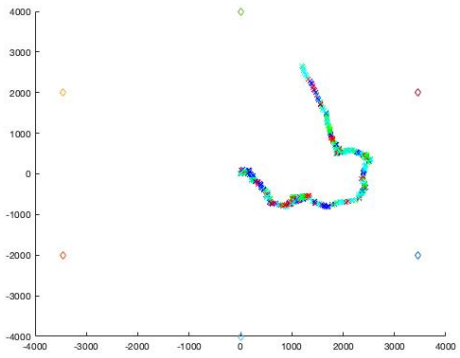


Figure 8: Most probable driving command

Problem 5

For this problem, some new RSSI-data is introduced. This data is measured on a target with an unknown standard deviation, $\varsigma \in (0, 3)$, for the independent Gaussian noise variables $\left\{V_n^\ell\right\}_{\ell=1}^s$. The SISR algorithm from problem 4 is implemented with Monte Carlo estimates of ς.

A well-defined grid of ς values in the range $(0, 3)$ is used to compute the log-likelihood values.

To find the optimal ς_j, the likelihood function, $c_j(\varsigma_j)$, is calculated for each ς_j and the one that produce the maximum likelihood value is defined as an estimation of the optimal ς. The likelihood function is given by the product of the PDF.

$$c_j(\varsigma_j) = \prod_{k=1}^n \frac{\Omega_k(\varsigma_j)}{N},$$

$$\Omega_k(\varsigma_j) = \sum_{i=1}^N \omega_i$$

The weight for every corresponding ς_j, is the transition density and therefore used for the likelihood function. The function is rewritten to natural log likelihood.

$$c_j(\varsigma_j) = \sum_{k=1}^n \ln\left(\frac{\Omega_k(\varsigma_j)}{N}\right),$$

Each c_j is then stored in an array in order to find the maximum.

It was discovered that the model does not work for $\varsigma < 0.3$ probably due to numerical errors at these evaluations. Moreover the grid was later made even finer in the closing areas of the maximum value, to more accurately identify the maximizing ς_j. The maximizing value was found to be $\hat{\varsigma_j} \approx 2.23$. Note however that this value will vary depending on iteration, and we got estimates of the max which would range between $\varsigma_j \in (2.1, 2.3)$.

Lastly, the trajectory for each ς_j is presented in the following plot.

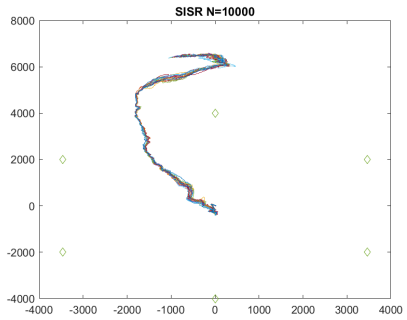


Figure 9: SISR algorithm based on data with varying ς