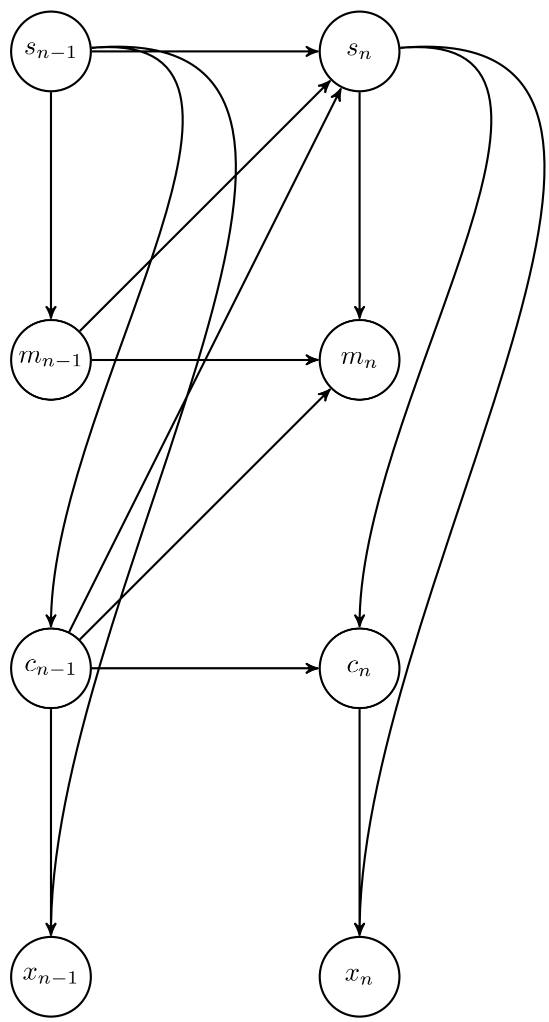
Mini-project : Barcode Decoding with HMMs

Hamidou Wissem & Ferjani Ahmed & Ahmadi Itaf

1. Question 1

The directed graphical model for the described model is :



2. Question 2

• We define the variable $\Psi_n = [s_n, m_n, c_n]$ which encapsulates the state of the model at pixel n, and for j $\in \{1 \ldots (S \times M \times C)\}$ we also define Ω a vector of possible states such that $\Psi_n = \Omega(j)$.
The transition matrix is defined as : A(i,j) = p($\Psi_{n+1} = \Omega(i) \mid \Psi_n = \Omega(j)$).
And we have :

$$\begin{aligned}
A(i,j) &= p(\Psi_n = \Omega_i \mid \Psi_{n-1} = \Omega_j) \\
&= p(s_n, m_n, c_n | s_{n-1}, m_{n-1}, c_{n-1}) \\
&= p(c_n | s_{n-1}, m_{n-1}, c_{n-1}) p(s_n | s_{n-1}, m_{n-1}, c_{n-1}) p(m_n | s_{n-1}, m_{n-1}, c_{n-1}) \\
&= \begin{cases} \delta(c_n - c_{n-1} - 1)\delta(m_n - m_{n-1})\delta(s_n - s_{n-1}), \ c_{n-1} \neq \mathrm{l}(s_n) \\ \delta(c_n - 1)\tau_s(s_n | s_{n-1}, m_{n-1})\tau_m(m_n | s_n, m_{n-1}), \ c_{n-1} = \mathrm{l}(s_n) \end{cases}
\end{aligned}$$

• The observation model is given by :

$$\begin{aligned}
p(x_n | \Omega_n) &= p(x_n | s_n, m_n, c_n) \\
&= p(x_n | s_n, c_n) \\
&= \prod_{i=0}^{1} \mathcal{N}(x_n, \mu_i, \sigma_i^2)^{1(f(s_n, c_n) = i)}
\end{aligned}$$

3. Question 3

In this part, we used the previous question to simulate the HMM and visualize the simulated data.
We remark that the simulated barecode look like a real scanline (see figure 1 and figure 2 below)

4. Question 4

See the code of part 1, 2 and 3 in template_code.m

Figure 1: An example of simulated barecode

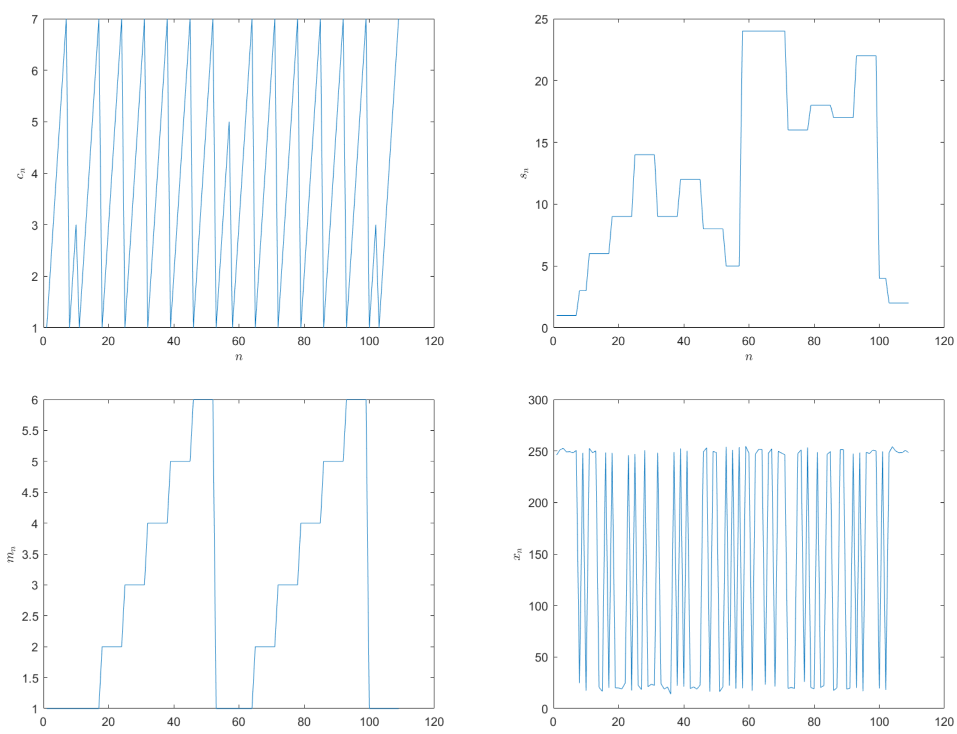


Figure 2: A representation of s_n, c_n, m_n and x_n of the simulated barecode

5. **Question 5**

See part 4 in template_code.m

- The marginals are given by :

$$p(s_n|x_{1:n}) = \sum_{c_n,m_n} p(s_n, m_n, c_n|x_{1:n}) = \sum_{c_n,m_n} p(\Psi_n|x_{1:n})$$

$$p(m_n|x_{1:n}) = \sum_{c_n,s_n} p(s_n, m_n, c_n|x_{1:n}) = \sum_{c_n,s_n} p(\Psi_n|x_{1:n})$$

$$p(c_n|x_{1:n}) = \sum_{m_n,s_n} p(s_n, m_n, c_n|x_{1:n}) = \sum_{s_n,m_n} p(\Psi_n|x_{1:n})$$

6. **Question 6**

- See part 5 in template_code.m

In this part, we tested our result using the function "log_sum_exp ", and we obtained a constant vector, which confirms that our result is correct, because all numbers must be equal to the marginal likelihood $p(x_{1:N})$.

We have :

```
Trial>> log_sum_exp(log_gamma)

ans =

  Columns 1 through 14

 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.046

  Columns 15 through 28

 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.046

  Columns 29 through 42

 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.0468 -142.046
```

- The marginals are given by :

$$p(s_{1:N}|x_{1:N}) = \sum_{c_{1:N},m_{1:N}} p(\Psi_{1:N}|x_{1:N})$$

$$p(m_{1:N}|x_{1:N}) = \sum_{c_{1:N},s_{1:N}} p(\Psi_{1:N}|x_{1:N})$$

$$p(c_{1:N}|x_{1:N}) = \sum_{m_{1:N},s_{1:N}} p(\Psi_{1:N}|x_{1:N})$$

7. **Question 7**

for this question, we used the pseudo-code of viterbi algorithm of wikipedia.

function *VITERBI* $(O, S, \Pi, Y, A, B) : X$
    for each state $i \in \{1, 2, \ldots, K\}$ **do**
        $T_1[i,1] \leftarrow \pi_i \cdot B_{iy_1}$
        $T_2[i,1] \leftarrow 0$
    end for
    for each observation $i = 2, 3, \ldots, T$ **do**
        for each state $j \in \{1, 2, \ldots, K\}$ **do**
            $T_1[j,i] \leftarrow \max_k (T_1[k, i-1] \cdot A_{kj} \cdot B_{jy_i})$
            $T_2[j,i] \leftarrow \arg\max_k (T_1[k, i-1] \cdot A_{kj} \cdot B_{jy_i})$
        end for
    end for
    $z_T \leftarrow \arg\max_k (T_1[k, T])$
    $x_T \leftarrow s_{z_T}$
    for $i \leftarrow T, T\text{-}1, \ldots, 2$ **do**
        $z_{i-1} \leftarrow T_2[z_i, i]$
        $x_{i-1} \leftarrow s_{z_{i-1}}$
    end for
    return X
end function

• See part 6 and part 7 of template_code.m for the most likely path computation using viterbi and the decoding of the barecode string.

Obtained result :

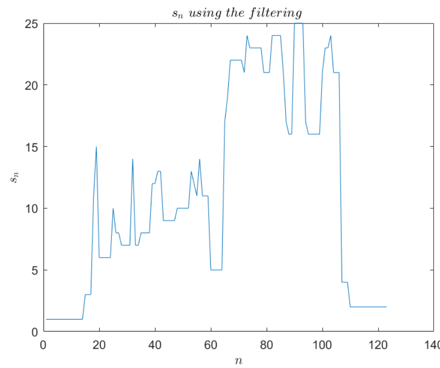
```
Real code:   012345678905
Decoded code:   012345678905
```

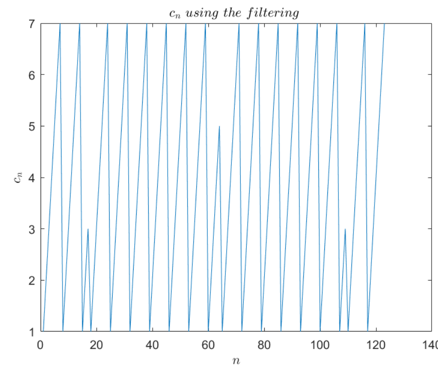
⇒ We succeeded to decode correctly the real code (we took here $noise = 0$).

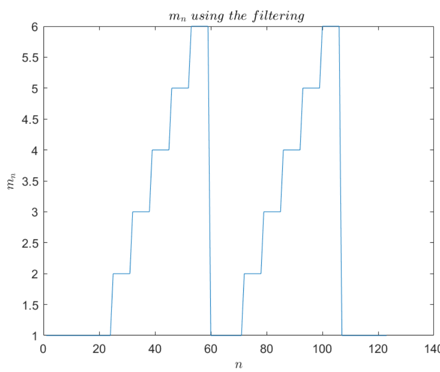
8. **Question 8**

1.

Visualize the obtained s_n, c_n and m_n using **the forward recursion (filtering)** :

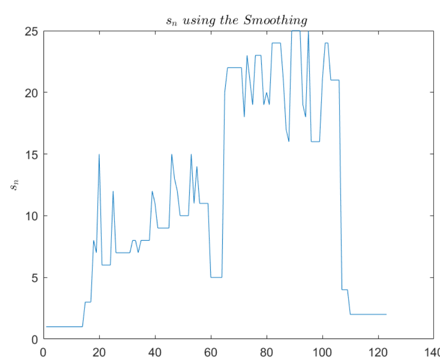


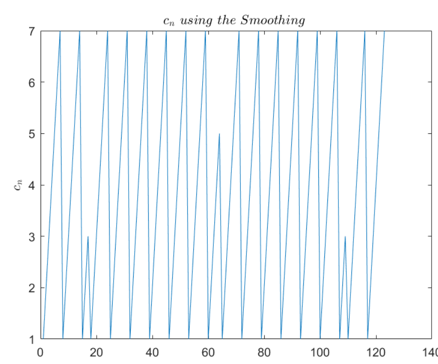


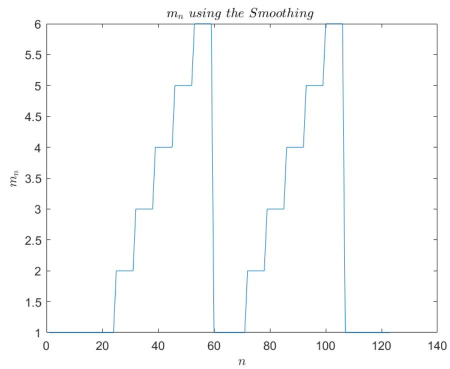


2.

Visualize the obtained s_n, c_n and m_n using **the forward-backward recursions (smoothing)** :

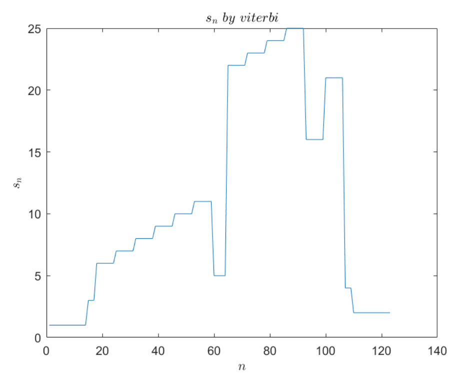


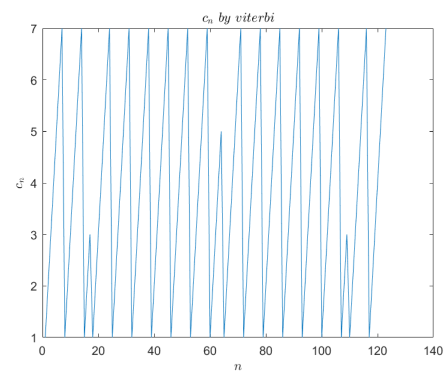


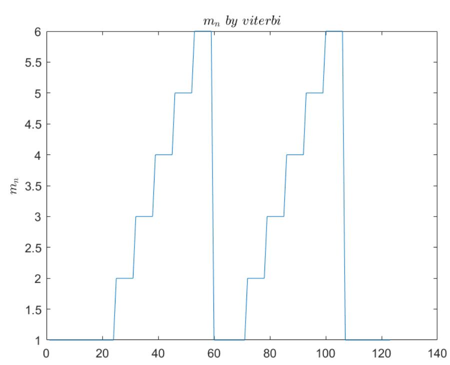


3.

We visualize the most likely path obtained by the Viterbi algorithm :







4.

In this part we tried different values for the observation noise.

For noise = 10

0 1 2 3 4 5 6 7 8 9 0 5

```
Real code:  012345678905
Decoded code:   012345678905
```

For noise = 50

0 1 2 3 4 5 6 7 8 9 0 5

```
Real code:  012345678905
Decoded code:   012345678905
```

For noise = 100

0 1 2 3 4 5 6 7 8 9 0 5

```
Real code:  012345678905
Decoded code:   012345678935
```

For noise = 150

0 1 2 3 4 5 6 7 8 9 0 5

```
Real code:  012345678905
Decoded code:   017368178005
```

For noise = 200

0 1 2 3 4 5 6 7 8 9 0 5

```
Real code:  012345678905
Decoded code:   317845670205
```

Comments :

We notice that the algorithm is very robust to the noise. In fact, even with a value of $noise = 50$, we still have a correct decoded code. With a value of $noise = 100$, the algorithm fails to decode only one digit. However, when we test our algorithm with a value of noise equal to 150 or 200, the results are not satisfactory, the decoded code does not correspond to the real one.