Probabilistic Reasoning — A.Y. 2025–2026

Assignment on Hidden Markov model — 15 december 2025

Students: Sasha Bravo, Nathan Ferrari

Instructors: Marco Forgione, Giorgio Corani

Load Libraries

In [265…

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
```

Maze

The robot can move on the cells with light background, but cannot go over the obstacles, which have the dark blue background.

In [266…

```
# Boolean representation of a random maze. True=feasible, False=obstacle.
# Use the maze corresponding to your group!
#maze = np.ones((2, 5), dtype=bool)
maze = np.ones((4, 16), dtype=bool)
rows, cols = maze.shape
N = rows*cols
obstacles = [11,12,13,14,19,21,23,25,27,30,35,39,48,49,50,51,52,53,55,59,62]
#obstacles = [2,4,5]
for cell in obstacles:
    r = cell // cols
    c = cell % cols
    maze[r][c] = False
```

In [267…

```
def plot_maze(ax, maze, skip_walls=True):
    cell_idx = 0
    rows, cols = maze.shape
    for r in range(rows):
        for c in range(cols):
            color = 'lightblue' if maze[r, c] == 1 else 'blue'
            ax.add_patch(patches.Rectangle((c, rows - 1 - r), 1, 1, edgecolor='b
            if maze[r, c] == 1 or not skip_walls:
                ax.text(c + 0.5, rows - 1 - r + 0.5, str(cell_idx), color='black
            cell_idx += 1

    ax.set_xlim(0, cols)
    ax.set_ylim(0, rows)


    ax.set_aspect('equal')
    ax.axis('off')

fig, ax = plt.subplots(figsize=(8, 6))
```

```
plot_maze(ax, maze)
```

In [268…

```
rows, cols = maze.shape
states = [(r, c) for r in range(rows) for c in range(cols) if maze[r, c]]

states_with_idx = []
for r in range(rows):
    for c in range(cols):
        state = (r, c)
        states_with_idx.append(state) if maze[r, c] else states_with_idx.append(

states_with_idx
```

Out[268…

```
[(0, 0),
 (0, 1),
 (0, 2),
 (0, 3),
 (0, 4),
 (0, 5),
 (0, 6),
 (0, 7),
 (0, 8),
 (0, 9),
 (0, 10),
 None,
 None,
 None,
 None,
 (0, 15),
 (1, 0),
 (1, 1),
 (1, 2),
 None,
 (1, 4),
 None,
 (1, 6),
 None,
 (1, 8),
 None,
 (1, 10),
 None,
 (1, 12),
 (1, 13),
 None,
 (1, 15),
 (2, 0),
 (2, 1),
 (2, 2),
 None,
 (2, 4),
 (2, 5),
 (2, 6),
 None,
 (2, 8),
 (2, 9),
 (2, 10),
 (2, 11),
 (2, 12),
 (2, 13),
 (2, 14),
 (2, 15),
 None,
 None,
 None,
 None,
 None,
 None,
 (3, 6),
 None,
 (3, 8),
 (3, 9),
 (3, 10),
 None,
```

```
 (3, 12),
 (3, 13),
 None,
 (3, 15)]
```

In [269…
```
# Suggested names for the possible observations. If can be seen as a binary digi
# This may be usefule to code the emission matrix
letters = ['N', 'S', 'E', 'W']
vocabulary = []
for i in range(2**len(letters)):  # observations can be indexed with numbers 0 t
    binary = format(i, '04b')  # 4-bit format interpreted as possibility to move
    combination = ''.join([letters[j] for j in range(4) if binary[j] == '1'])
    vocabulary.append(combination)
```

In [270…
```
vocabulary
```

Out[270…
```
['',
 'W',
 'E',
 'EW',
 'S',
 'SW',
 'SE',
 'SEW',
 'N',
 'NW',
 'NE',
 'NEW',
 'NS',
 'NSW',
 'NSE',
 'NSEW']
```

In [271…
```
pi = np.zeros(cols*rows, dtype=float)
for i in range(len(pi)):
    pi[i] = 1/len(states) if states_with_idx[i] is not None else 0
pi
```

Out[271…
```
array([0.02325581, 0.02325581, 0.02325581, 0.02325581, 0.02325581,
       0.02325581, 0.02325581, 0.02325581, 0.02325581, 0.02325581,
       0.02325581, 0.        , 0.        , 0.        , 0.        ,
       0.02325581, 0.02325581, 0.02325581, 0.02325581, 0.        ,
       0.02325581, 0.        , 0.02325581, 0.        , 0.02325581,
       0.        , 0.02325581, 0.        , 0.02325581, 0.02325581,
       0.        , 0.02325581, 0.02325581, 0.02325581, 0.02325581,
       0.        , 0.02325581, 0.02325581, 0.02325581, 0.        ,
       0.02325581, 0.02325581, 0.02325581, 0.02325581, 0.02325581,
       0.02325581, 0.02325581, 0.02325581, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.02325581,
       0.        , 0.02325581, 0.02325581, 0.02325581, 0.        ,
       0.02325581, 0.02325581, 0.        , 0.02325581])
```

1 HMM Definition and Trajectory Generation

1.1 Transition probability definition

Define the *transition probability matrix* A corresponding to the robot maze. Consider that:

- The robot can only reach a neighboring cell.
- The robot can move North, South, East, West.
- Diagonal moves are not allowed.
- Passing over obstacles is not allowed.
- Feasible transitions are equally probable.

In [272…
```python
def count_neighbours(states, state):
    count = 0
    neighbours = []
    if (state[0], state[1]-1) in states:
        count += 1
        neighbours.append((state[0], state[1]-1))
    if (state[0], state[1]+1) in states:
        count += 1
        neighbours.append((state[0], state[1]+1))
    if (state[0]-1, state[1]) in states:
        count += 1
        neighbours.append((state[0]-1, state[1]))
    if (state[0]+1, state[1]) in states:
        count += 1
        neighbours.append((state[0]+1, state[1]))
    return count, neighbours
```

In [273…
```python
A = np.zeros((rows*cols, rows*cols))
for state in states:
    count_neighbours_res = count_neighbours(states, (state[0],state[1]))
    for neighbour in count_neighbours_res[1]:
        A[states_with_idx.index((state[0],state[1])), neighbour[1] + neighbour[0

for row in A:
    for val in row:
        print(f"{val:.2f}", end=" ")
    print()
```

```
0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.33 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.33 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.50 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.33 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.50 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.33 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.33 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.50 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

```
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.25 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.25 0.00 0.25 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.25 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

```
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.33 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.50 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.33 0.00 0.33 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.25 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.25 0.00 0.25 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.25 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.50 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.25 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.25 0.00 0.25 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.25 0.00 0.00 0.00
```

```
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.25 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.25 0.00 0.25 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.25 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.50
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.33
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.33 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.33
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.33 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.33 0.00 0.33 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

```
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.50 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

The transition probability matrix A models the motion of the robot inside the grid-based environment under strict physical constraints. You may transition only between neighboring cells in the four cardinal directions. You may not move diagonally, or through any obstacles. At each valid state, every feasible exit transition towards reachable neighbours is given equal probability, ensuring uniform redistribution of probability mass among admissible moves. With this construction the matrix will be stochastic and will model the local and non deterministic behaviour of the robot in maze.

The matrix A reflects the motion dynamics of the robot in the maze. For each state, it tells us the probability of going to any of the states in one time-step. A starting state is represented by a row in the matrix while each column corresponds to a destination state. According to the entries of the matrix, the robot can only move to its physically reachable neighbouring cells. The transition probabilities being equal reflects that we assume uniform random choice over all feasible movements.

The transition matrix A indicates the probability of transitioning from one hidden state to another in the Hidden Markov Model. Each element A_{ij} is the probability of going from state i to state j in one time step.

$$A_{ij} = P(X_{t+1} = j \mid X_t = i).$$

Here, the hidden state in time t is X_t. In a grid world, actors can only transition to adjacent and accessible non-obstructed cells (North, South, East, West). Diagonal moves and moves through obstructed cells are not allowed.

If state i has n_i valid neighboring cells then any valid transition is equally likely.

$$A_{ij} = \begin{cases} \dfrac{1}{n_i}, & \text{if state } j \text{ is a valid neighbor of } i \\ 0, & \text{otherwise} \end{cases}$$

1.2 Emission probability definition

Define the *emission probability matrix* B.

- After each move, the robot senses whether the cells to the North, South, East, West are feasible.
- There are 16 possible observations (previously defined variable `vocabulary`).

In [274…
```python
def vocabulary_counter(vocabulary, states):
    dict = {}
    for voc in vocabulary:
        dict[voc] = []
    for state in states:
        obs = ''
        if state is not None:
            if (state[0]-1, state[1]) in states:
                obs += 'N'
            if (state[0]+1, state[1]) in states:
                obs += 'S'
            if (state[0], state[1]+1) in states:
                obs += 'E'
            if (state[0], state[1]-1) in states:
                obs += 'W'

        dict[obs].append(state)

    return dict
```

In [275…
```python
B = np.zeros((rows*cols, len(vocabulary)))
dict = vocabulary_counter(vocabulary, states)
for voc in dict:
    for state in dict[voc]:
        B[state[0]*cols + state[1], vocabulary.index(voc)] = 1

## if any row sums to zero we have to put 1 in the first column
for row in B:
    if np.sum(row) == 0:
        row[0] = 1

for row in B:
    for val in row:
        print(f"{val:.2f}", end=" ")
    print()
```

```
0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00
0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00
0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

```
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00
1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

The emission probability matrix encodes the relationship between the robot's position and its observations. In each position of the grid, the robot identifies whether the neighboring positions North, South, East, West are accessible (i.e. are free). Every possible observation defined in variable `vocabulary` corresponds to a specific combination of available directions. The matrix B is constructed such that for a given state, the column corresponding to the matching observation is assigned a probability of one, while all other entries are zero. The deterministic assignment assumes that the robot's sensors totally detect the local configuration of free and blocked neighbouring cells.

An emission matrix B provides the probability of obtaining a certain reading from a sensor given the robot's state. Every element B_{ik} indicates the probability for observing O_k when the robot is in state i.

$$B_{ik} = P(O_t = k \mid X_t = i)$$

where the observation O_t happened at the time t. The robot checks if the neighboring cells in the North, South, East, West are free. Any combination of accessible paths pertains to a predetermined observation in the `vocabulary`.

The matrix is deterministic:

$$B_{ik} = \begin{cases} 1, & \text{if observation } k \text{ matches the accessible neighbors of state } i \\ 0, & \text{otherwise} \end{cases}$$

This reflects perfect sensing: the robot can fully detect which neighboring cells are free.

1.3 Trajectory Sampling

Sample and visualize a state and observation sequences Z and Y of length $T = 20$ compatible with the previous hypotheses.

- Use the previously defined transition and emission matrices
- Consider the initial position equiprobable among the admissible states

In [304…

```python
def generate_sequence(T, A, B, pi, vocabulary):
    Z = np.empty(shape=T, dtype=int)
    Y = np.empty(shape=T, dtype=object)

    # initial state
    Z[0] = np.random.choice(N, p=pi)
    Y[0] = vocabulary[np.random.choice(len(vocabulary), p=B[Z[0]])]

    for t in range(1, T):
        # transition
        Z[t] = np.random.choice(N, p=A[Z[t-1]])
        # emission according to noisy probabilities
        Y[t] = vocabulary[np.random.choice(len(vocabulary), p=B[Z[t]])]
    return Z, Y
```

```
T = 20
Z, Y = generate_sequence(T, A, B, pi, vocabulary)
print(Z)
print(Y)
```

```
[38 22  6  7  6 22 38 22 38 37 38 37 38 22 38 54 38 37 36 37]
['NSW' 'NS' 'SEW' 'EW' 'SEW' 'NS' 'NSW' 'NS' 'NSW' 'EW' 'NSW' 'EW' 'NSW'
 'NS' 'NSW' 'N' 'NSW' 'EW' 'NE' 'EW']
```

In this snippet, we sample and visualize a sequence of hidden states Z and observations Y of length $T = 20$ from the Hidden Markov Model defined earlier. The starting state Z_0 is from the uniform distribution π over all admissible states, meaning it would be uniformly random. For $t > 0$, the states Z_t are obtained from the transition matrix A, which gives the probabilities with which the state moves into an accessible neighboring cell:

$$Z_0 \sim \pi, \quad Z_t \sim P(X_t \mid X_{t-1}) = A_{Z_{t-1}}$$

For every hidden state Z_t, the observation Y_t is locked in by the emission matrix B. This matrix captures the robot's immediate sensory input—what adjacent cells are free:

$$Y_t = O_k \quad \text{such that } B_{Z_t,k} = 1$$

Here's the kicker: this setup creates sequences (Z, Y) that align perfectly with the model's rules. The robot only moves where it can, and at each step, it sees exactly which nearby cells are open.

2 Decoding and inference

2.1 Likelihood

Compute the likelihood of the obtained observation sequence Y.

In [277…
```
def likelihood(A, B, pi, Y):
    alpha_all = np.zeros((T, N))

    for t in range(T):
        for z_t in range(N):
            if t == 0:
                alpha_all[t, z_t] = pi[z_t] * B[z_t, vocabulary.index(Y[t])]
            else:
                for z_prev in range(N):
                    alpha_all[t, z_t] += alpha_all[t-1, z_prev] * A[z_prev, z_t]

    #print(alpha_all)
    P_Y = np.sum(alpha_all[T-1, :])
    return P_Y
```

In [305…
```
print(f"The probability of our sequence is: {float(likelihood(A, B, pi, Y))}")
```

```
The probability of our sequence is: 4.615303726861451e-09
```

The likelihood is the probability of having a specific sequence of observations Y. To compute the likelihood is used the forward algorithm that compute for every t the probabilities of every state taking in consideration A and B matrixes following the following formulas:

Initialization (for each state i):

$$\alpha_1(i) = \pi_i \, b_i(y_1)$$

Recursion (for each time $t = 2, \ldots, T$ and state j):

$$\alpha_t(j) = \left(\sum_{i=1}^{N} \alpha_{t-1}(i) \, a_{ij} \right) b_j(y_t)$$

where π_i is the initial probability of being in state i, a_{ij} is the element in position (i, j) of the transition matrix A, and $b_j(y_t)$ is the emission probability of observing y_t in state j, as defined by the emission matrix B.

2.1 Decoding

Obtain the most probable state sequence, given the observation sequence Y.

In [279…
```
def viterbi(A, B, pi, Y):
    v_all = np.zeros((T, N))
    bp_all = np.zeros((T, N), dtype=int)

    for i in range(N):
        v_all[0, i] = pi[i] * B[i, vocabulary.index(Y[0])]

    for t in range(1, T):
        for j in range(N):
            max_val = -1
            max_idx = -1
            for i in range(N):
                val = v_all[t-1, i] * A[i, j]
                if val > max_val:
                    max_val = val
                    max_idx = i
            v_all[t, j] = max_val * B[j, vocabulary.index(Y[t])]
            bp_all[t, j] = max_idx

    Z_vit = np.zeros(T, dtype=int)
    Z_vit[T - 1] = np.argmax(v_all[T - 1])
    for t in range(T - 2, -1, -1):
        Z_vit[t] = bp_all[t + 1, Z_vit[t + 1]]

    return Z_vit
```

In [306…
```
Z_vit = viterbi(A, B, pi, Y)
print([states_with_idx[z][1]+states_with_idx[z][0]*cols for z in Z_vit])
print([int(z) for z in Z])
Z_vit_list = [states_with_idx[z][1]+states_with_idx[z][0]*cols for z in Z_vit]

diffs = [a - b for a, b in zip(Z, Z_vit_list)]
res = [f'error at pos {i}' if int(z) != 0 else 'correct' for i,z in enumerate(di
```

```
print(res)

mismatches = sum(r != "correct" for r in res )
print(f"mismatch: {mismatches/len(res)}")
```

```
[38, 22, 6, 5, 6, 22, 38, 22, 38, 37, 38, 37, 38, 22, 38, 54, 38, 37, 36, 37]
[38, 22, 6, 7, 6, 22, 38, 22, 38, 37, 38, 37, 38, 22, 38, 54, 38, 37, 36, 37]
['correct', 'correct', 'correct', 'error at pos 3', 'correct', 'correct', 'correc
t', 'correct', 'correct', 'correct', 'correct', 'correct', 'correct', 'correct',
'correct', 'correct', 'correct', 'correct', 'correct', 'correct']
mismatch: 0.05
```

The Viterbi algorithm is used to compute the most probable hidden state sequence for an observation sequence $Y = (y_1, y_2, \ldots, y_T)$. To find the best sequence, we need to find the probability of being in each state. We also want the max one for each t steps. Viterbi verifies both matrix A and matrix B for applicable matrices as per the following formula.

Initialization (for each state i).

$$\delta_1(i) = \pi_i \, b_i(y_1)$$

Recursion (for each time $t = 2, \ldots, T$ and state j).

$$\delta_t(j) = \max_{i=1}^{N} \left[\delta_{t-1}(i) \, a_{ij} \right] b_j(y_t)$$

Backtracking.

$$q_T^* = \arg\max_{i=1}^{N} \delta_T(i)$$

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, \ldots, 1$$

where π_i is the initial probability of being in state i, a_{ij} is the (i, j) entry of the transition matrix A, and $b_j(y_t)$ is the probability of observing y_t in state j, as defined by the emission matrix B.

2.2 Filtering

Obtain the filtering distribution $P(z_t \mid y_{1:t})$ at each time step t.

In [281…

```
def filtering(A, B, pi, Y):
    alpha_all = np.zeros((T, N))

    for t in range(T):
        for z_t in range(N):
            if t == 0:
                alpha_all[t, z_t] = pi[z_t] * B[z_t, vocabulary.index(Y[t])]
            else:
                for z_prev in range(N):
                    alpha_all[t, z_t] += alpha_all[t-1, z_prev] * A[z_prev, z_t]
        alpha_all[t, :] = alpha_all[t, :] / np.sum(alpha_all[t, :])
    #print(alpha_all)
    return alpha_all
```

In [307…

```
alpha_all = filtering(A, B, pi, Y)
for t in range(T):
    print(f"t = {t}, alpha = {alpha_all[t]}")
```

```
t = 0, alpha = [0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.33333333 0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.33333333 0.         0.         0.
 0.         0.         0.         0.         0.         0.33333333
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
t = 1, alpha = [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
0.  0.
 0.  0.  0.  0.  0.5 0.  0.  0.  0.  0.  0.  0.  0.5 0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
t = 2, alpha = [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
t = 3, alpha = [0.  0.  0.  0.  0.  0.5 0.  0.5 0.  0.  0.  0.  0.  0.  0.  0.
0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
t = 4, alpha = [0.   0.   0.   0.   0.25 0.   0.5  0.   0.25 0.   0.   0.   0.
0.
 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
 0.   0.   0.   0.   0.   0.   0.   0.  ]
t = 5, alpha = [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
0.
 0.   0.   0.   0.   0.   0.   0.25 0.   0.5  0.   0.25 0.   0.   0.
 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
 0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.
 0.   0.   0.   0.   0.   0.   0.   0.  ]
t = 6, alpha = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
t = 7, alpha = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
1. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
t = 8, alpha = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
t = 9, alpha = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
t = 10, alpha = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
t = 11, alpha = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
```

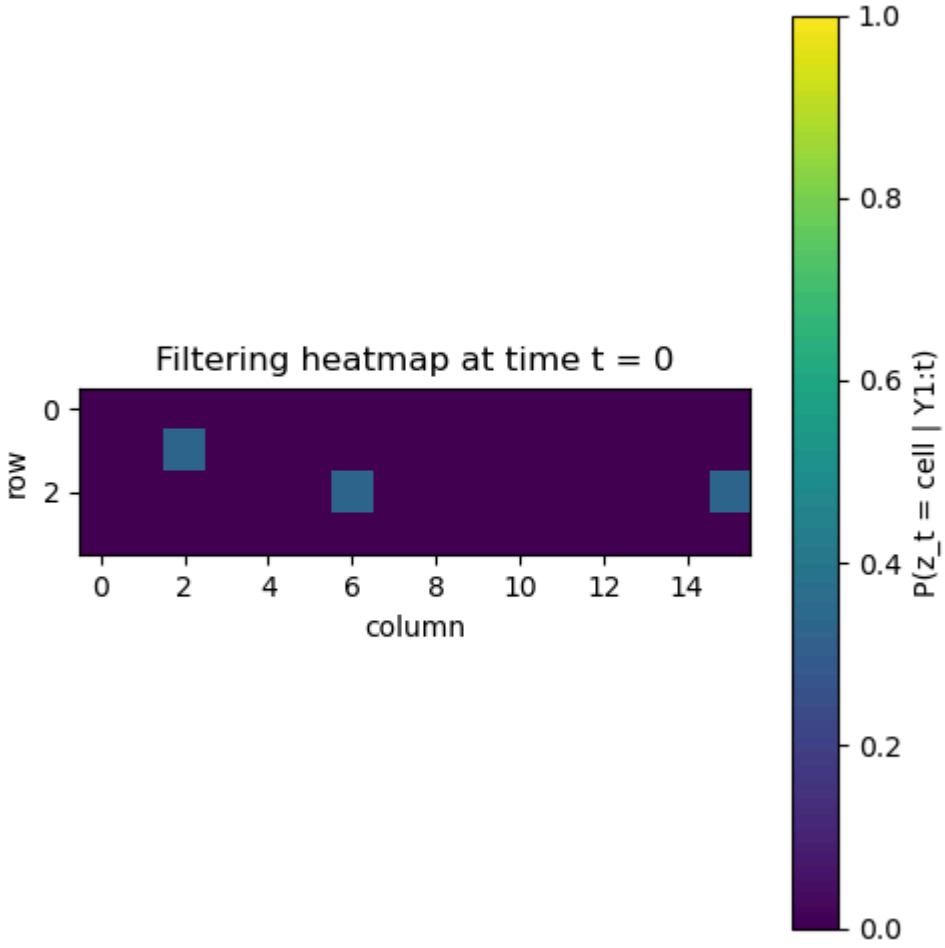
```
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
t = 12, alpha = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
t = 13, alpha = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 1. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
t = 14, alpha = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
t = 15, alpha = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
t = 16, alpha = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
t = 17, alpha = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
t = 18, alpha = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
t = 19, alpha = [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

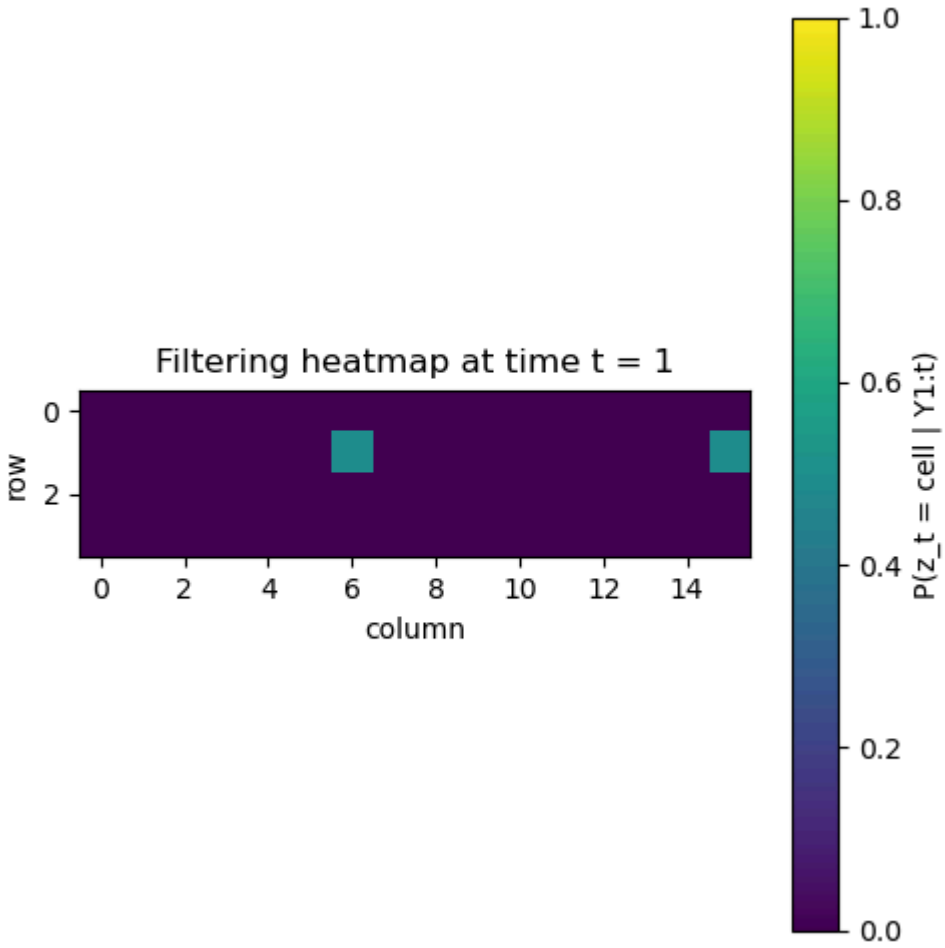
In [283…

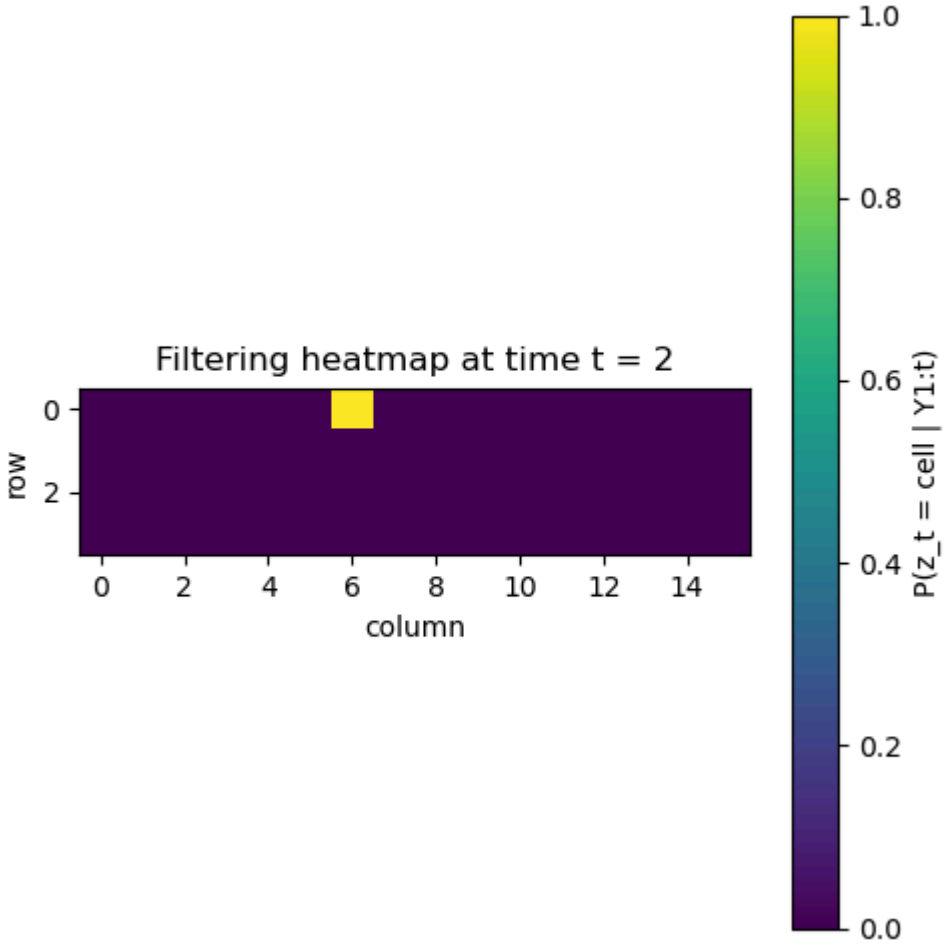
```python
def plot_heatmap(alpha_t, t):
    grid = alpha_t.reshape((4, 16))

    plt.figure(figsize=(5, 5))
    im = plt.imshow(grid, origin='upper', cmap='viridis',vmax=1)
    plt.colorbar(im, label='P(z_t = cell | Y1:t)')
    plt.title(f'Filtering heatmap at time t = {t}')
    plt.xlabel('column')
    plt.ylabel('row')
    plt.tight_layout()
    plt.show()
```

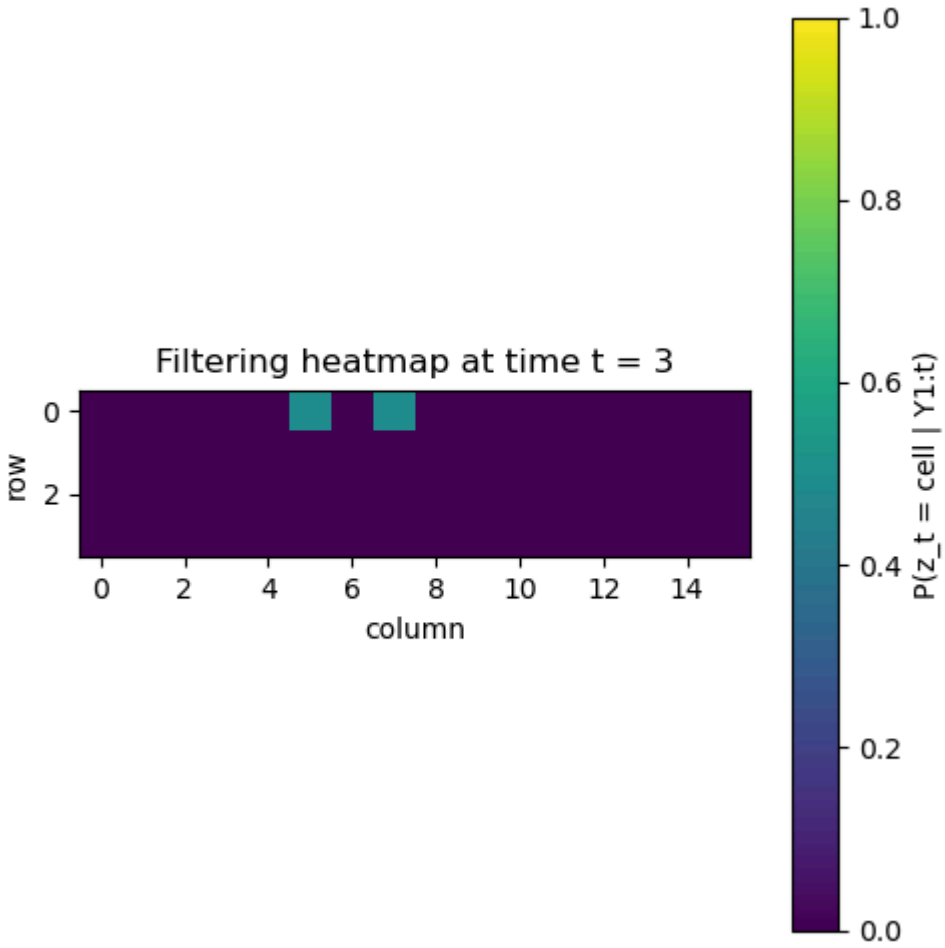
In [308…

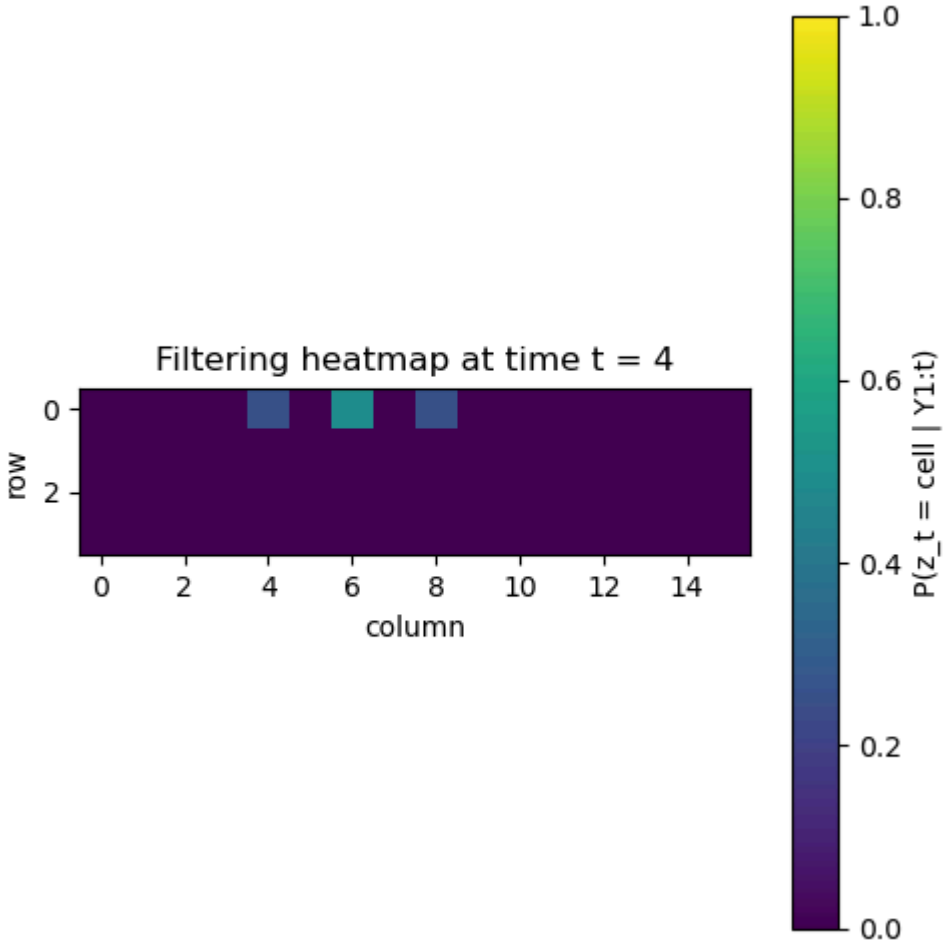
```python
for t in range(T):
    plot_heatmap(alpha_all[t],t)
```

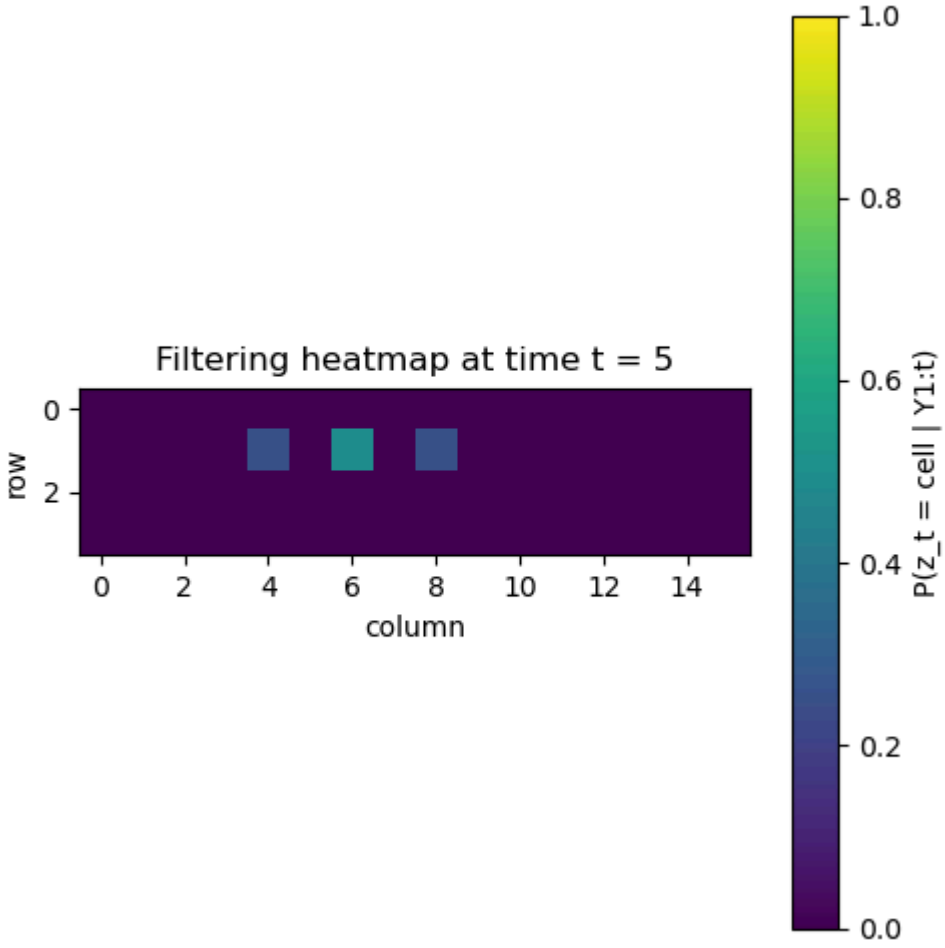

Filtering heatmap at time t = 0
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

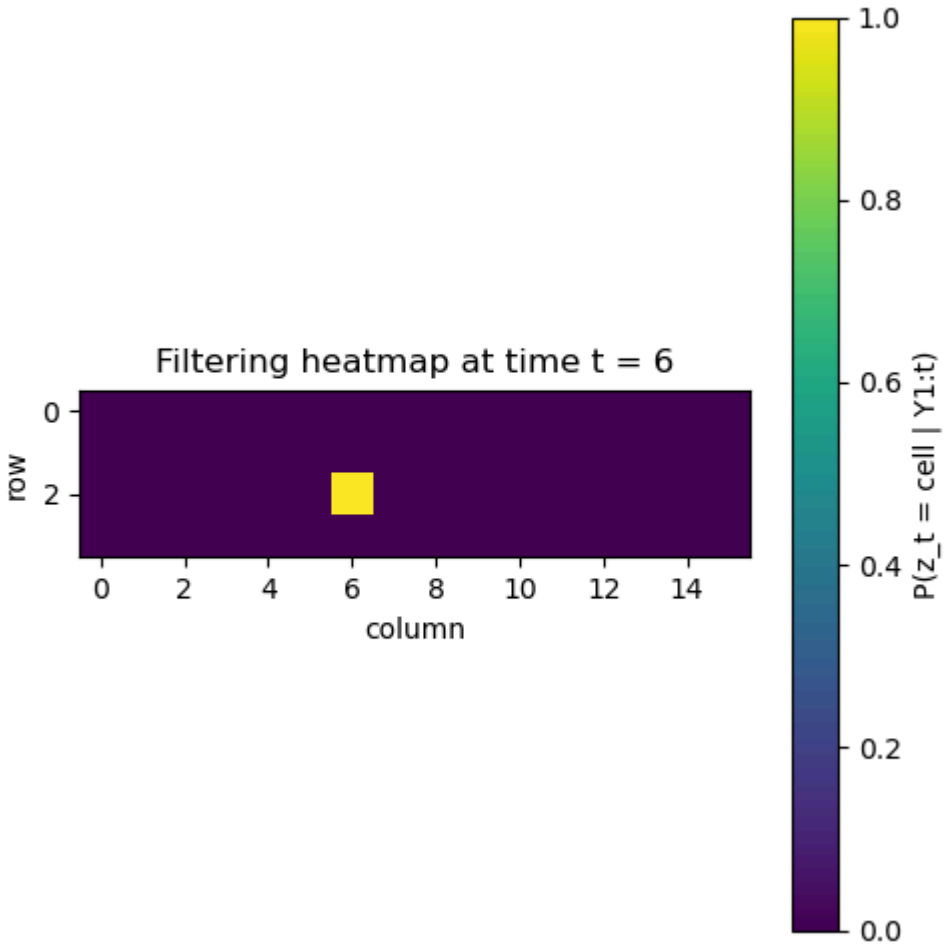

Filtering heatmap at time t = 1
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

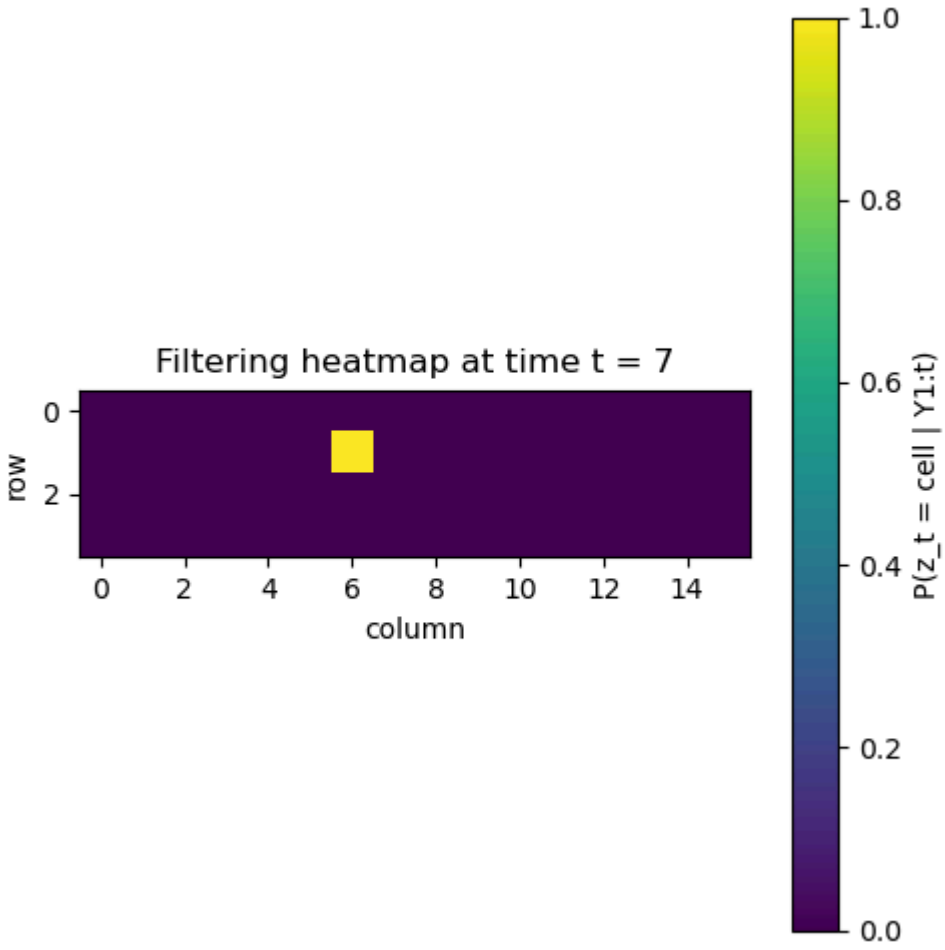

Filtering heatmap at time t = 2
row
column
P(z_t = cell | Y1:t)

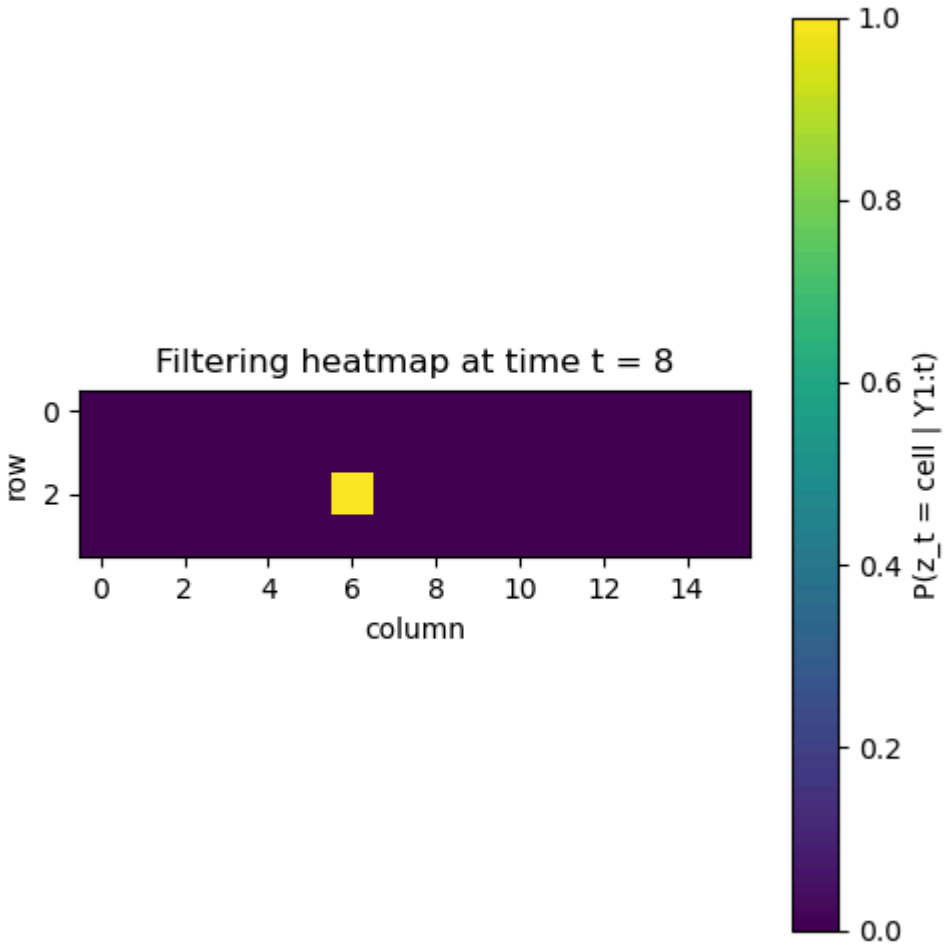

Filtering heatmap at time t = 3
row
column
P(z_t = cell | Y1:t)

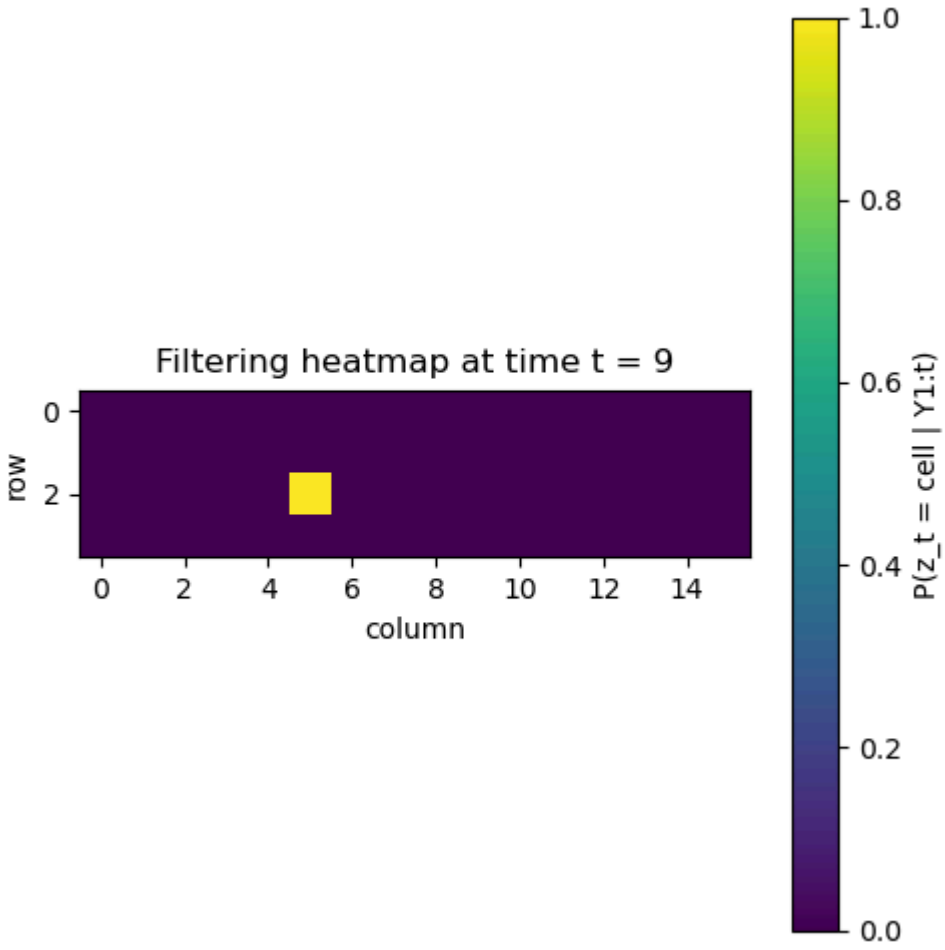

Filtering heatmap at time t = 4
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

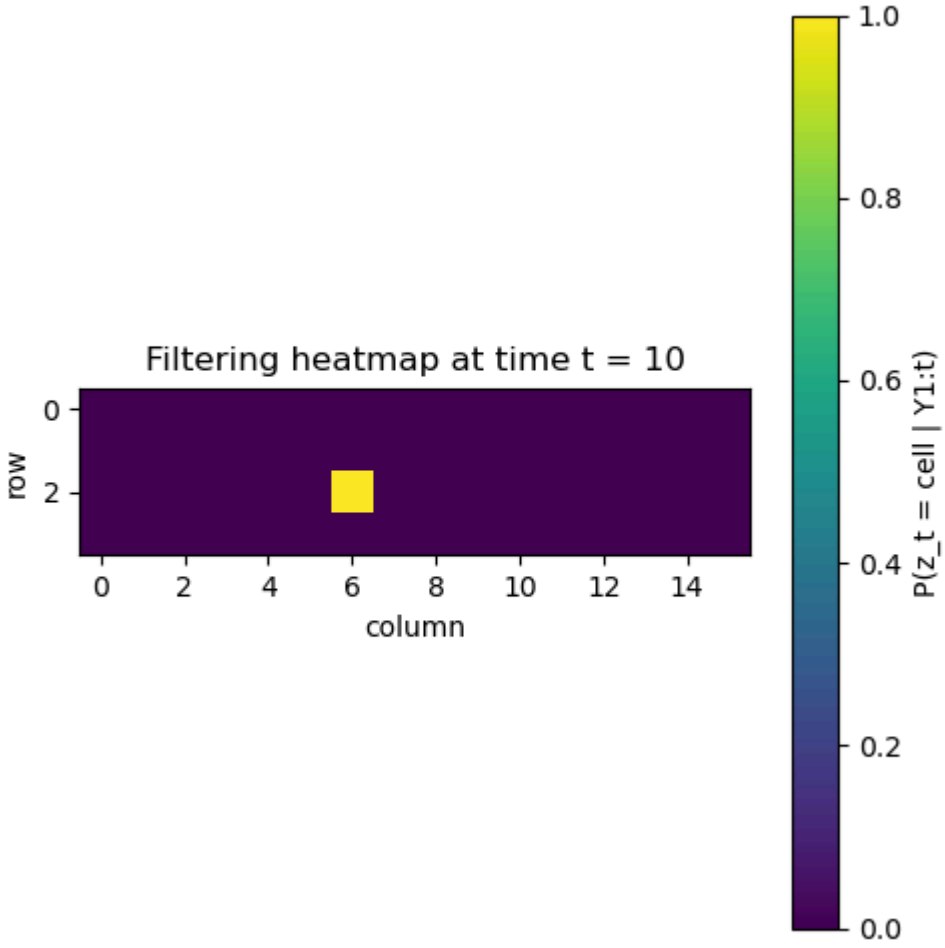

Filtering heatmap at time t = 5
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

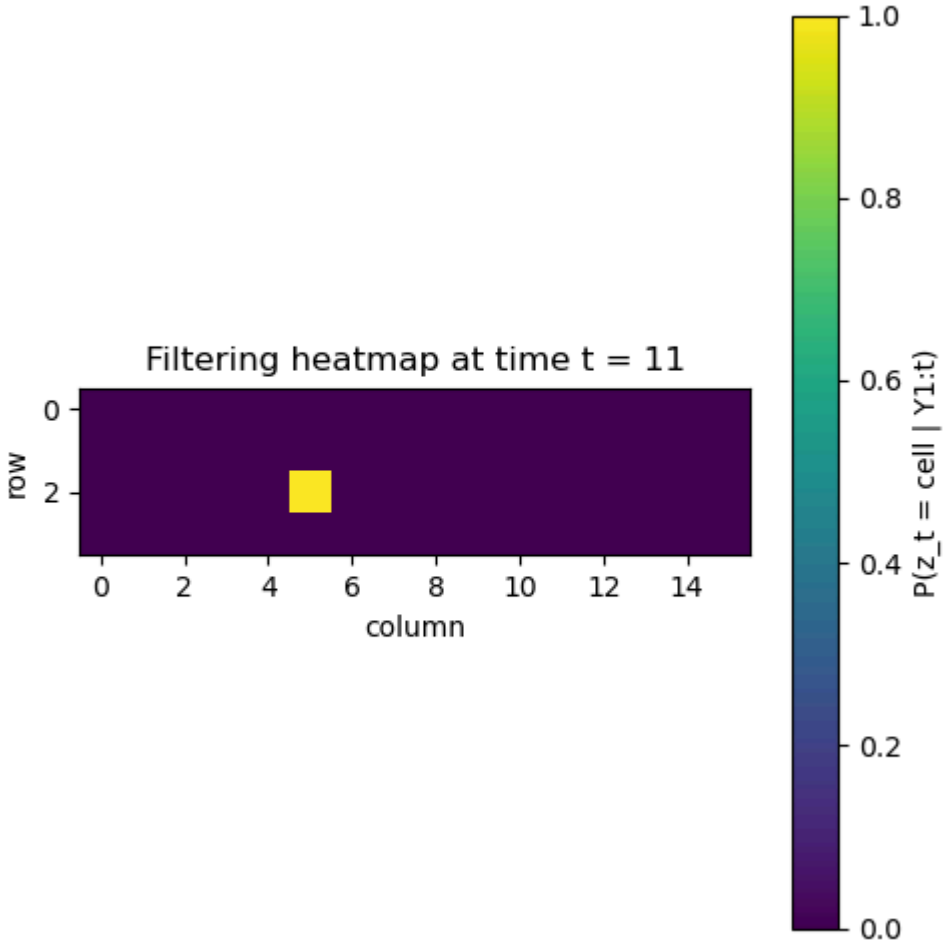

Filtering heatmap at time t = 6
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

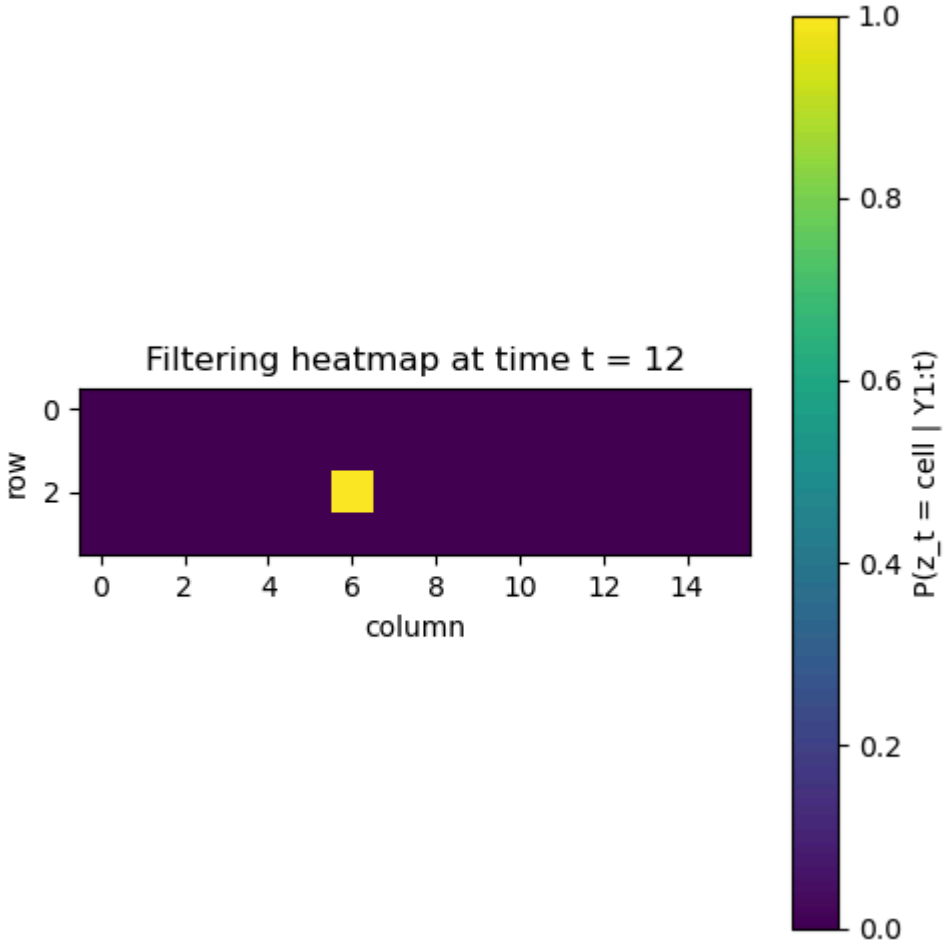

Filtering heatmap at time t = 7
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

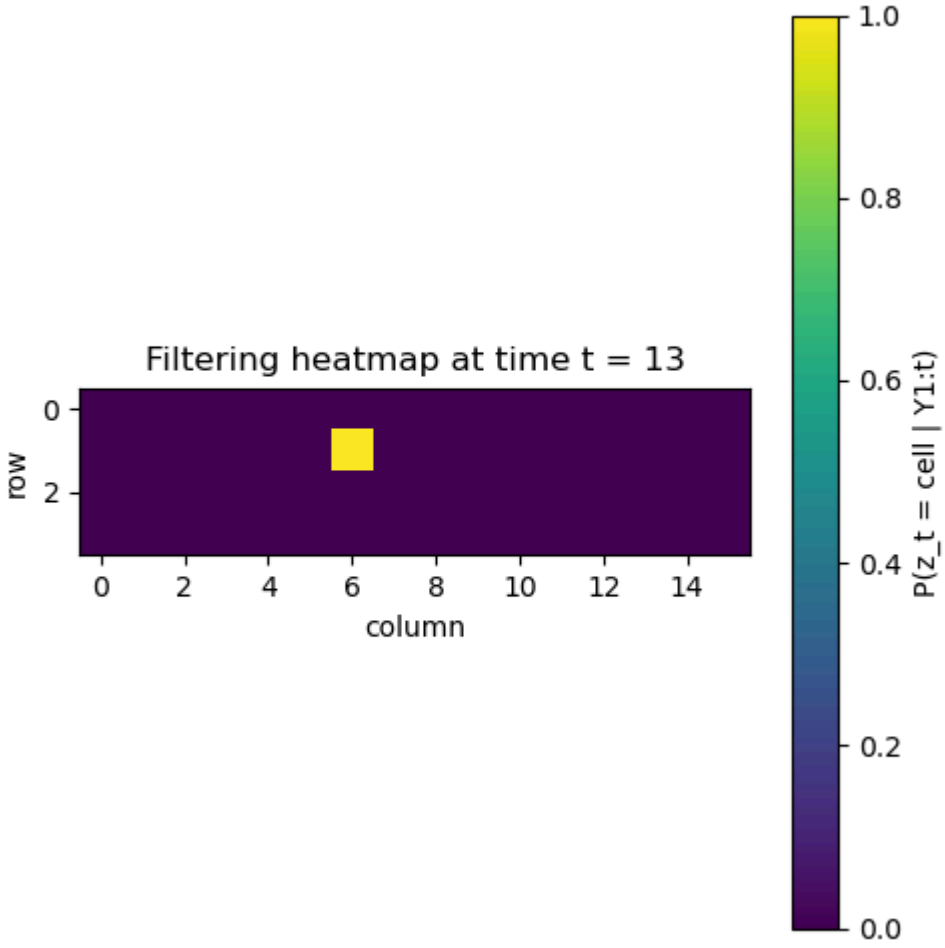

Filtering heatmap at time t = 8
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

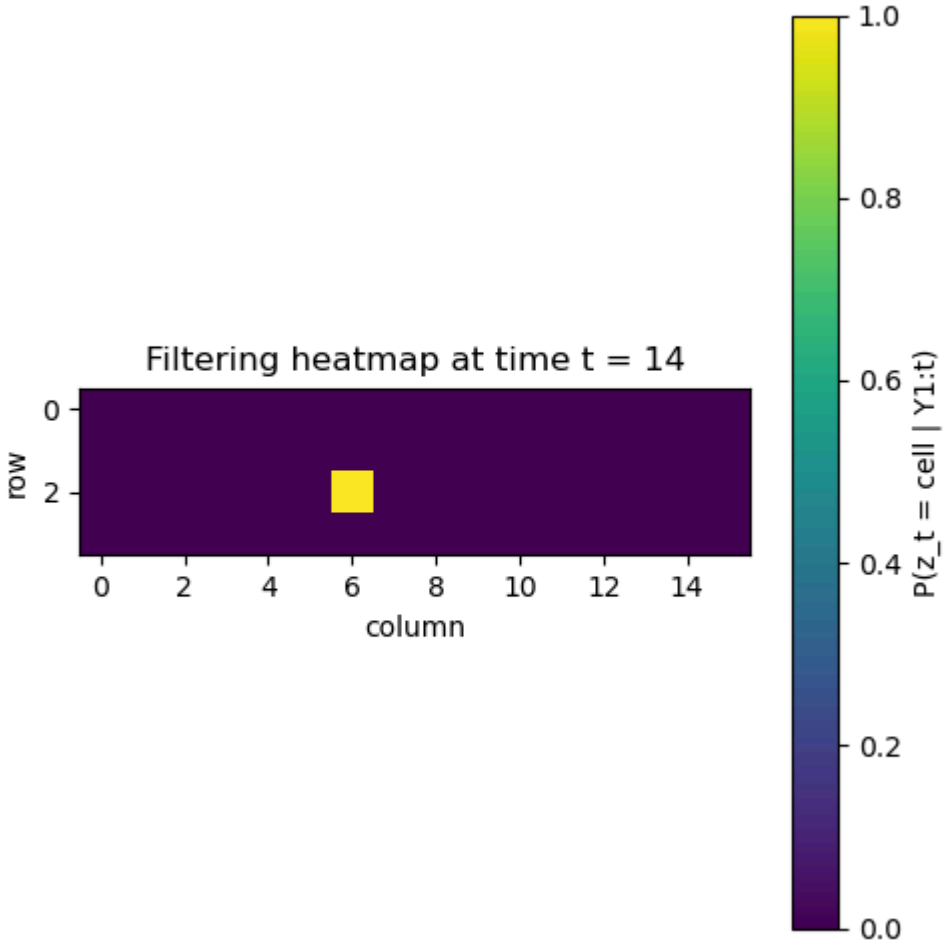

Filtering heatmap at time t = 9
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

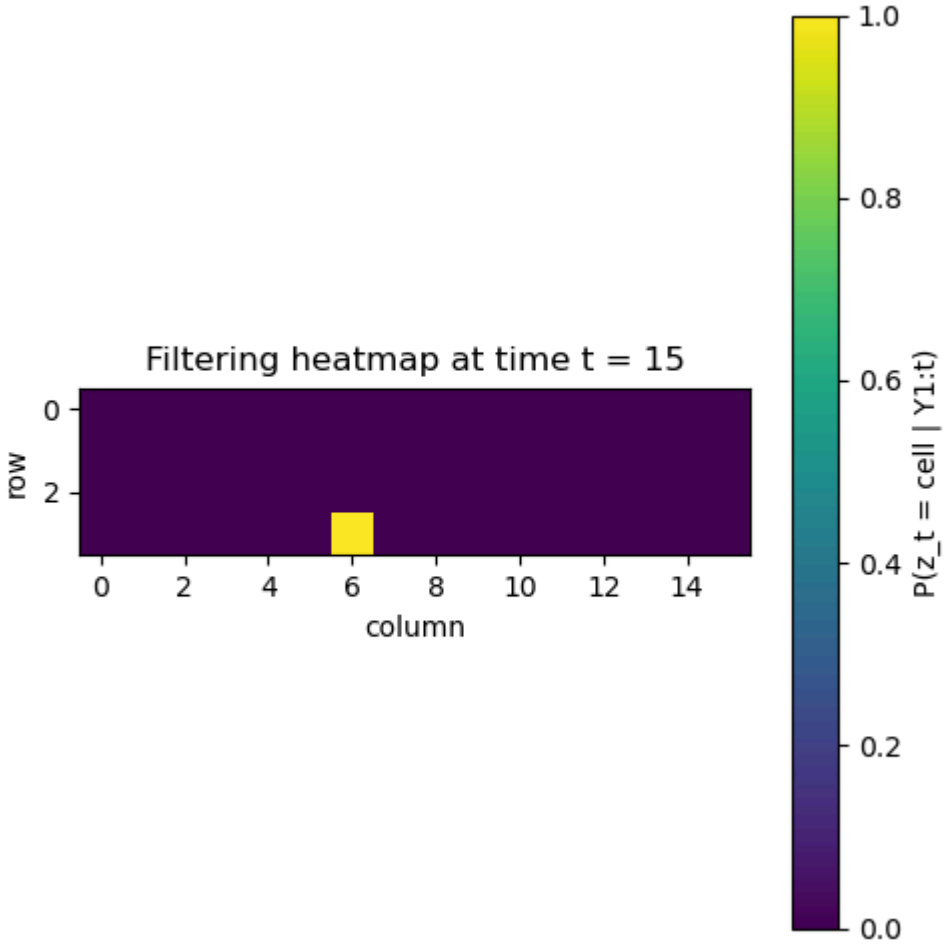

Filtering heatmap at time t = 10
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

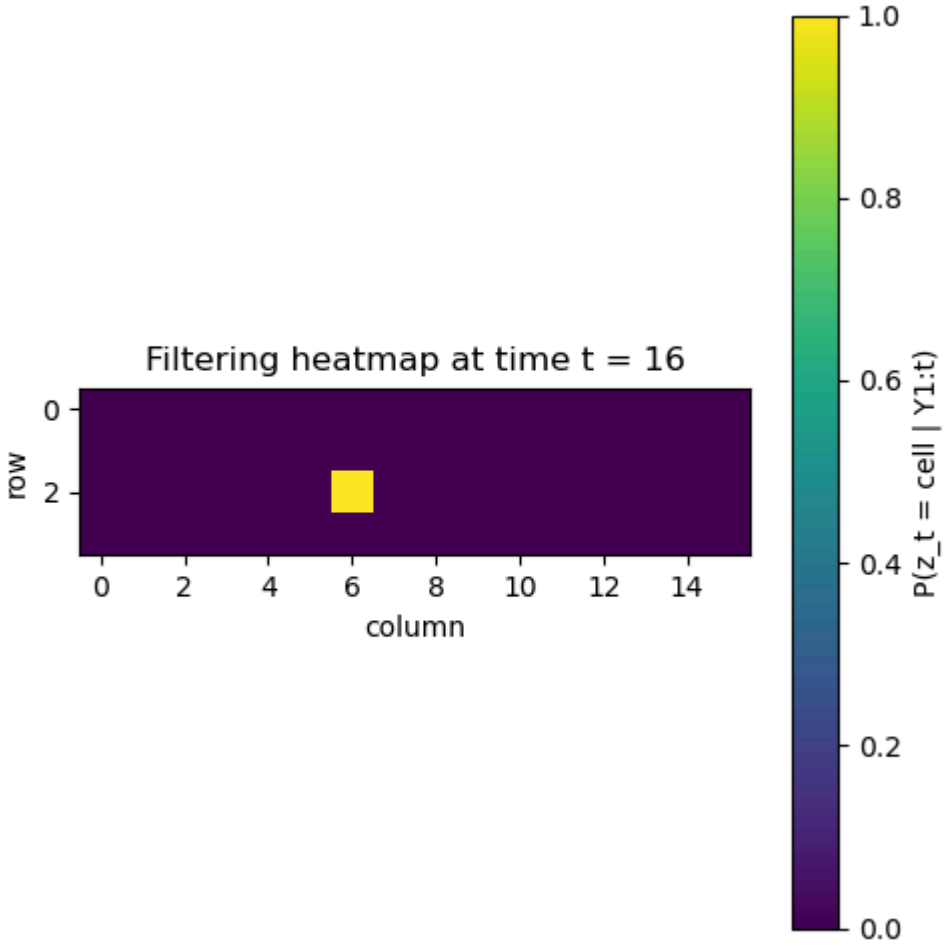

Filtering heatmap at time t = 11
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

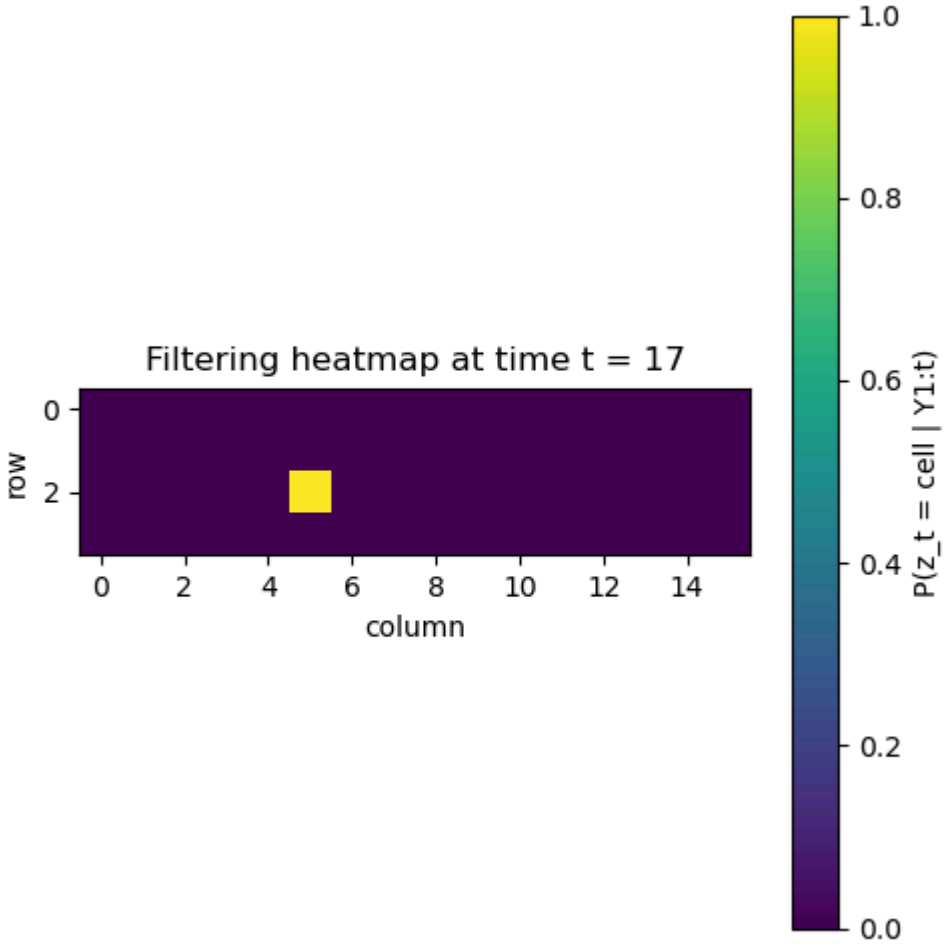

Filtering heatmap at time t = 12
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

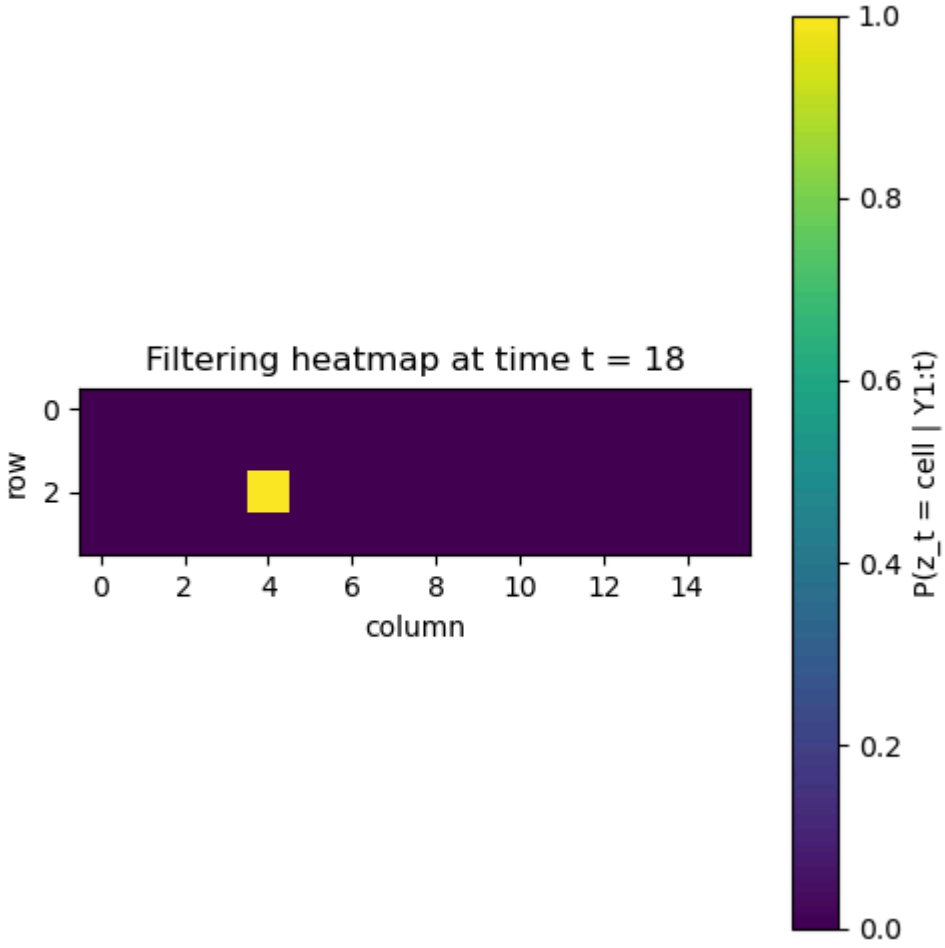

Filtering heatmap at time t = 13
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

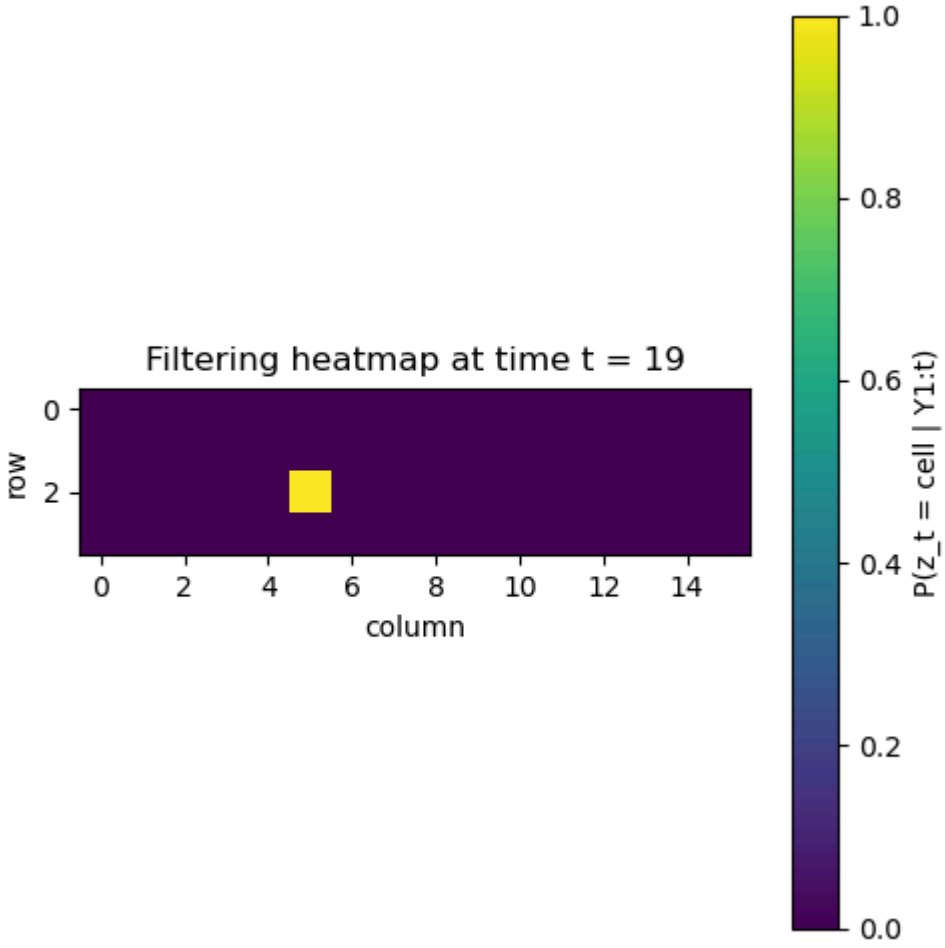

Filtering heatmap at time t = 14
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0


Filtering heatmap at time t = 15
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0


Filtering heatmap at time t = 16
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0


Filtering heatmap at time t = 17
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0


Filtering heatmap at time t = 18
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0


Filtering heatmap at time t = 19
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

In *filtering* for Hidden Markov Models (HMMs), we take into account all the observations made up to each time step.
The goal of this algorithm is to estimate the probability distribution over the agent's position in every cell, given all the observations available up to time (t).

For example, if we are at time (t = 4), the filtering algorithm computes the probability of being in each cell by using both the transition model and all previous observations — specifically those from (t = 1, 2, 3) — together with the current observation.

The algorithm it's a modified forward that at the end of the loop over a specific time t normalize the probabilities to sum to one.

To better visualize our results, we decided to use a heatmap of the maze to represent the intensity of the probabilities associated with each cell.
If a cell appears completely yellow, it means that the probability of being in that cell is maximal; we are therefore (ideally) 100% certain that the agent is located there.

2.3 Noisy observations

Repeat steps 2.1 and 2.2 in the presence of sensor error noise. In particular, consider that:

- With probability $1 - \epsilon$, the sensor provides the correct reading
- With probability ϵ, the sensor returns a wrong reading.
- The ϵ probability is equally split among all wrong readings.
- Set $\epsilon = 0.2$. Optionally, test also other values.

In [285…
```
B_noisy = B.copy()
noise_level = 0.2
for i in range(B_noisy.shape[0]):
    for j in range(B_noisy.shape[1]):
        if B_noisy[i, j] > 0:
            B_noisy[i, j] = B_noisy[i, j] - noise_level
        else:
            B_noisy[i, j] = (noise_level / (B_noisy.shape[1] - 1))
#print(B_noisy)
#print the sum of each row to verify they sum to 1
#for row in B_noisy:
#    print(f"Sum of row: {np.sum(row)}")
```

In [286…
```
Z_noisy, Y_noisy = generate_sequence(T, A, B_noisy, pi, vocabulary)
print(Z_noisy)
print(Y_noisy)
```

```
[ 8  9 10 26 10 26 42 41 40 56 40 56 40 41 40 41 57 56 57 58]
['SEW' 'EW' 'SW' 'NS' 'NSW' 'NS' 'NSEW' 'SEW' 'NSE' 'NE' 'NSE' 'NE' 'NSE'
 'SEW' 'NSE' 'SEW' 'NEW' 'E' 'NEW' 'NW']
```

In [287…
```
print(f"The probability of our sequence is: {float(likelihood(A, B_noisy, pi, Y_
```

```
The probability of our sequence is: 6.768789440495186e-15
```

In [288…
```
Z_vit = viterbi(A, B_noisy, pi, Y_noisy)
print([states_with_idx[z][1]+states_with_idx[z][0]*cols for z in Z_vit])
print([int(z) for z in Z_noisy])
```

```
Z_vit_list = [states_with_idx[z][1]+states_with_idx[z][0]*cols for z in Z_vit]

diffs = [a - b for a, b in zip(Z_noisy, Z_vit_list)]
res = [f'error at pos {i}' if int(z) != 0 else 'correct' for i,z in enumerate(di
print(res)

mismatches = sum(r != "correct" for r in res )
print(f"mismatch: {mismatches/len(res)}")
```

```
[8, 9, 10, 26, 10, 26, 42, 41, 40, 56, 40, 56, 40, 41, 40, 41, 57, 56, 57, 58]
[8, 9, 10, 26, 10, 26, 42, 41, 40, 56, 40, 56, 40, 41, 40, 41, 57, 56, 57, 58]
['correct', 'correct', 'correct', 'correct', 'correct', 'correct', 'correct', 'co
rrect', 'correct', 'correct', 'correct', 'correct', 'correct', 'correct', 'correc
t', 'correct', 'correct', 'correct', 'correct', 'correct']
mismatch: 0.0
```

In [289…

```
alpha_all = filtering(A, B_noisy, pi, Y_noisy)
for t in range(T):
    print(f"t = {t}, alpha = {alpha_all[t]}")
for t in range(T):
    plot_heatmap(alpha_all[t],t)
```

```
t = 0, alpha = [0.00251889 0.1511335  0.1511335  0.00251889 0.1511335  0.00251889
 0.1511335  0.00251889 0.1511335  0.00251889 0.00251889 0.
 0.         0.         0.         0.00251889 0.00251889 0.00251889
 0.00251889 0.         0.00251889 0.         0.00251889 0.
 0.00251889 0.         0.00251889 0.         0.00251889 0.00251889
 0.         0.00251889 0.00251889 0.00251889 0.00251889 0.
 0.00251889 0.00251889 0.00251889 0.         0.00251889 0.1511335
 0.00251889 0.00251889 0.00251889 0.00251889 0.00251889 0.00251889
 0.         0.         0.         0.         0.         0.
 0.00251889 0.         0.00251889 0.00251889 0.00251889 0.
 0.00251889 0.00251889 0.         0.00251889]
t = 1, alpha = [2.31071547e-03 2.35806620e-03 2.36753634e-03 2.72740187e-01
 1.70462617e-04 2.72740187e-01 1.70462617e-04 2.72740187e-01
 1.70462617e-04 1.39779346e-01 1.13641744e-04 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 5.68208722e-05
 1.42052181e-04 2.38647663e-03 2.35806620e-03 0.00000000e+00
 2.32965576e-03 0.00000000e+00 2.31071547e-03 0.00000000e+00
 2.31071547e-03 0.00000000e+00 8.52313083e-05 0.00000000e+00
 8.52313083e-05 8.52313083e-05 0.00000000e+00 1.51522326e-04
 7.57611629e-05 1.42052181e-04 7.57611629e-05 0.00000000e+00
 1.13641744e-04 5.68208722e-03 2.27283489e-04 0.00000000e+00
 2.38647663e-03 1.04171599e-04 2.44329750e-03 3.40925233e-03
 1.98873053e-04 1.98873053e-04 3.97746105e-03 2.27283489e-04
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 3.78805815e-05 0.00000000e+00
 7.57611629e-05 2.38647663e-03 6.62910176e-05 0.00000000e+00
 8.52313083e-05 8.52313083e-05 0.00000000e+00 3.78805815e-05]
t = 2, alpha = [1.62404847e-04 4.95211748e-04 2.68816931e-02 1.64865526e-04
 5.33776748e-02 2.21461155e-05 5.33758293e-02 2.21461155e-05
 4.04203517e-02 2.21461155e-05 8.17689948e-01 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 1.47640770e-05
 3.48801318e-04 3.24809693e-04 2.77441613e-04 0.00000000e+00
 2.21461155e-05 0.00000000e+00 2.58371347e-05 0.00000000e+00
 1.66095866e-04 0.00000000e+00 1.30108428e-04 0.00000000e+00
 1.79937188e-05 1.07962313e-03 0.00000000e+00 2.58371347e-05
 1.84550962e-05 1.31031183e-04 1.62404847e-04 0.00000000e+00
 7.80650570e-04 2.58371347e-05 7.86187099e-04 0.00000000e+00
 2.39301081e-04 4.29080987e-04 3.53722677e-04 1.28724296e-04
 3.58490244e-04 4.13855533e-04 2.44530025e-05 4.09703136e-04
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 1.47640770e-05 0.00000000e+00
 3.10045616e-04 2.06081908e-05 2.74058179e-04 0.00000000e+00
 1.79937188e-05 1.79937188e-05 0.00000000e+00 1.47640770e-05]
t = 3, alpha = [1.00214269e-05 3.24965951e-04 1.21104416e-05 9.52963198e-04
 3.72516599e-06 1.26754251e-03 1.24902624e-06 1.11369315e-03
 3.74707873e-06 1.50432502e-02 2.71170171e-06 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 4.60167563e-07
 6.11365477e-06 1.48714470e-05 3.24965951e-04 0.00000000e+00
 3.88611507e-02 0.00000000e+00 3.85857076e-02 0.00000000e+00
 2.89664525e-02 0.00000000e+00 8.73988036e-01 0.00000000e+00
 2.24208428e-05 4.00592298e-06 0.00000000e+00 3.23432059e-04
 5.69731269e-06 6.11365477e-06 4.85002003e-06 0.00000000e+00
 8.54596903e-07 2.32384620e-05 1.44624091e-06 0.00000000e+00
 1.35749431e-05 6.23600091e-06 1.45856683e-05 6.34236901e-06
 6.61901736e-06 2.31768324e-05 8.55007767e-06 1.42158908e-06
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 9.33482771e-06 0.00000000e+00
 3.08604437e-06 1.54977862e-05 3.39464881e-06 0.00000000e+00
 3.51288631e-06 4.00592298e-06 0.00000000e+00 4.86462853e-06]
t = 4, alpha = [5.13254984e-05 5.93684974e-06 3.22354635e-04 2.45490945e-06
```

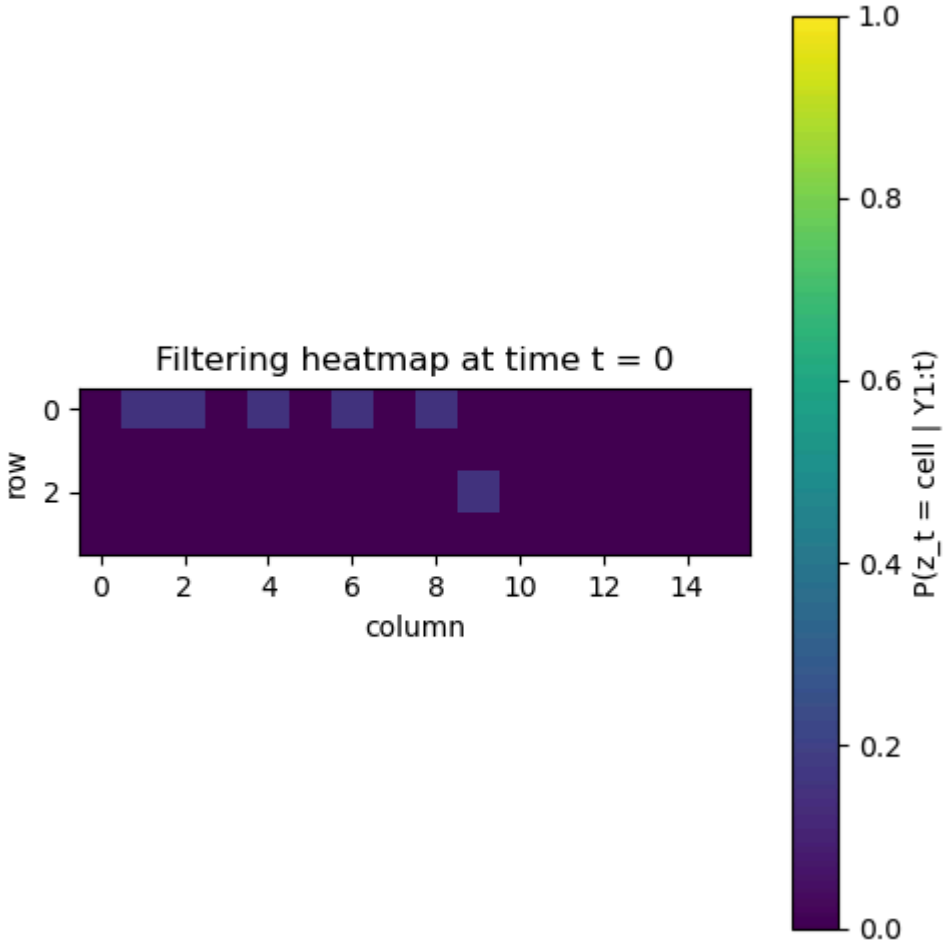
```
 9.55300382e-03 7.71122386e-07 9.52632876e-03 7.74519401e-07
 1.04928576e-02 1.21146044e-06 2.06732640e-01 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 7.52099103e-05
 5.38426864e-06 1.02650997e-04 2.84058387e-04 0.00000000e+00
 7.76217908e-07 0.00000000e+00 4.17832835e-07 0.00000000e+00
 2.68534029e-06 0.00000000e+00 2.32642443e-06 0.00000000e+00
 1.70110830e-06 7.90840996e-06 0.00000000e+00 4.34393282e-07
 1.89553432e-06 4.18172536e-06 5.13254984e-05 0.00000000e+00
 9.04205796e-03 4.22928357e-07 5.38940918e-01 0.00000000e+00
 6.73745538e-03 6.20284308e-06 2.03237759e-01 2.46543665e-06
 1.02001211e-05 4.62084262e-06 2.91511650e-06 4.76763188e-03
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 2.24202984e-07 0.00000000e+00
 4.50698954e-06 2.47373457e-06 4.09839234e-06 0.00000000e+00
 1.70110830e-06 3.51161109e-06 0.00000000e+00 2.20381343e-07]
t = 5, alpha = [1.73307933e-07 7.29185586e-06 4.49572703e-06 1.51175697e-04
 9.19019135e-08 2.92073590e-04 4.50864105e-08 3.06461225e-04
 1.07265485e-07 4.90773995e-03 8.12389059e-08 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 9.97478330e-09
 2.40065635e-06 4.58579585e-06 7.29185586e-06 0.00000000e+00
 2.12321880e-02 0.00000000e+00 5.03768557e-01 0.00000000e+00
 1.58260862e-02 0.00000000e+00 4.24832479e-01 0.00000000e+00
 2.98707830e-07 9.21150031e-08 0.00000000e+00 4.58632346e-03
 1.46439829e-07 2.40065635e-06 4.41248791e-06 0.00000000e+00
 2.75354722e-08 8.45793883e-03 2.96025827e-08 0.00000000e+00
 2.60109782e-07 2.43660733e-03 2.99098460e-07 2.33354707e-03
 1.87789556e-07 4.46281970e-07 7.30377528e-05 8.70343720e-08
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 8.25031000e-03 0.00000000e+00
 1.03177367e-04 2.92557079e-07 2.33346783e-03 0.00000000e+00
 1.97746057e-07 9.21150031e-08 0.00000000e+00 7.29846996e-05]
t = 6, alpha = [2.34295904e-07 1.98097591e-07 5.83405750e-06 1.10896189e-07
 7.85936513e-04 3.31140188e-09 1.82880054e-02 3.68278360e-09
 7.62906413e-04 5.53858245e-09 1.55820860e-02 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 1.66296948e-04
 9.47327434e-08 2.81155084e-05 3.51807354e-07 0.00000000e+00
 3.21995100e-09 0.00000000e+00 1.80544783e-09 0.00000000e+00
 8.88051706e-09 0.00000000e+00 8.36821977e-09 0.00000000e+00
 6.74458728e-09 1.89218807e-08 0.00000000e+00 2.82722878e-09
 1.16061540e-07 2.48442507e-07 2.34295904e-07 0.00000000e+00
 1.07654367e-03 1.71399695e-09 1.91712832e-02 0.00000000e+00
 6.36483998e-04 1.87821006e-08 9.37935440e-01 8.82711302e-09
 5.07833465e-03 1.59503011e-04 1.01948105e-08 1.74237997e-04
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 7.15579585e-10 0.00000000e+00
 1.33595479e-08 1.47250895e-04 1.24944989e-08 0.00000000e+00
 6.74458728e-09 1.52610759e-08 0.00000000e+00 2.10387115e-09]
t = 7, alpha = [6.57284314e-09 3.67288691e-05 9.64611917e-07 1.77720105e-05
 2.37219117e-07 4.28131971e-04 1.77764195e-08 4.27615039e-04
 3.65681905e-08 5.41755264e-04 4.68226233e-10 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 9.51895813e-11
 4.85104752e-07 2.00459971e-08 6.12147818e-07 0.00000000e+00
 5.38870621e-05 0.00000000e+00 8.40807795e-04 0.00000000e+00
 3.14105903e-05 0.00000000e+00 1.63142415e-02 0.00000000e+00
 8.54915165e-05 2.68536968e-06 0.00000000e+00 1.51089889e-05
 7.70288055e-09 4.85104752e-07 1.34731540e-08 0.00000000e+00
 1.66120423e-10 4.66562886e-04 1.66681211e-10 0.00000000e+00
 1.17037905e-09 9.48432122e-01 1.42120423e-09 1.58751013e-02
 2.68589396e-06 8.54923736e-05 6.59607289e-06 5.80107172e-10
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
```

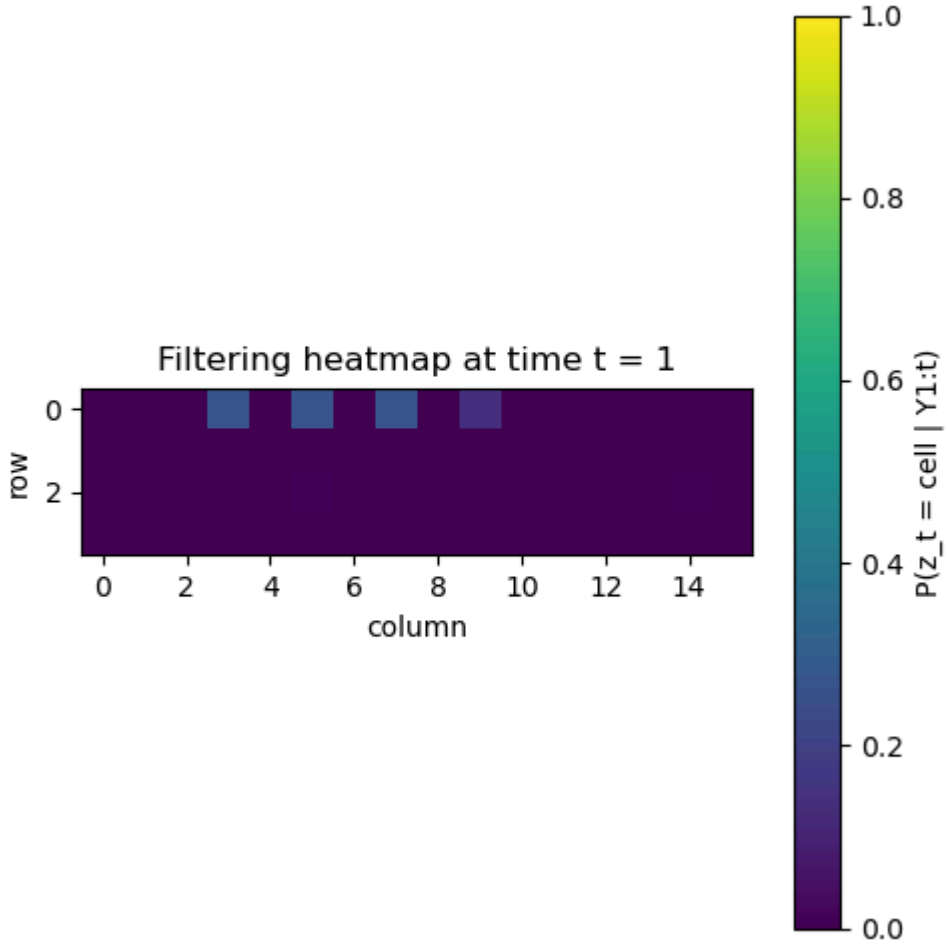
```
 0.00000000e+00 0.00000000e+00 4.30316884e-04 0.00000000e+00
 1.75916424e-05 1.29205772e-09 1.57929156e-02 0.00000000e+00
 8.54913933e-05 2.68536968e-06 0.00000000e+00 3.91093029e-06]
t = 8, alpha = [6.31153600e-07 1.67821387e-08 1.08543151e-06 2.03832030e-08
 1.27147769e-05 4.32475585e-09 4.31606691e-05 9.21691308e-10
 2.54600517e-05 6.32112797e-10 4.28819686e-04 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 3.84375483e-07
 3.70898402e-08 6.49763139e-07 1.69576839e-08 0.00000000e+00
 4.02749171e-09 0.00000000e+00 3.04317207e-10 0.00000000e+00
 6.40050808e-10 0.00000000e+00 2.99896120e-11 0.00000000e+00
 1.02481123e-07 3.26239097e-06 0.00000000e+00 1.46819706e-11
 1.64548732e-08 7.93709191e-10 1.86095393e-08 0.00000000e+00
 1.32403433e-05 7.05306834e-12 5.51544355e-05 0.00000000e+00
 9.65206365e-01 5.98410819e-11 1.73062037e-02 3.41828982e-08
 4.09302830e-04 3.38602744e-07 1.08748076e-06 7.51170386e-07
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 2.82693396e-12 0.00000000e+00
 4.17632322e-11 1.64877484e-02 3.99913084e-11 0.00000000e+00
 1.02481123e-07 3.26238784e-06 0.00000000e+00 9.83868941e-12]
t = 9, alpha = [8.84040574e-10 4.13448023e-08 1.05540470e-09 2.26461778e-07
 7.07324236e-10 9.16917520e-07 1.36632281e-10 1.12606781e-06
 5.40018242e-11 1.09732134e-05 1.62976766e-11 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 3.61397279e-13
 2.39378803e-08 1.17534140e-09 2.62670063e-08 0.00000000e+00
 5.34561385e-07 0.00000000e+00 1.61335342e-06 0.00000000e+00
 1.58394793e-02 0.00000000e+00 2.23551852e-04 0.00000000e+00
 5.11780644e-06 6.68993672e-09 0.00000000e+00 3.12495605e-08
 3.73002018e-08 8.86008439e-09 2.91300822e-10 0.00000000e+00
 5.95862862e-09 1.23099729e-06 7.80356165e-12 0.00000000e+00
 1.77648729e-11 1.63226224e-02 8.44117949e-10 2.18033942e-04
 1.00539269e-08 5.22487863e-06 1.64940854e-08 2.67692271e-08
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 9.05085720e-07 0.00000000e+00
 9.66577555e-01 2.99438491e-12 4.83560818e-04 0.00000000e+00
 3.07068382e-04 6.68993672e-09 0.00000000e+00 1.23267255e-08]
t = 10, alpha = [7.18126454e-10 3.58935347e-11 4.48045497e-09 1.93904759e-11
 2.76866746e-08 9.28374076e-12 6.03310131e-08 2.09702464e-12
 2.61557219e-04 8.62952094e-13 3.86975490e-06 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 5.15629916e-10
 3.83850997e-08 1.10453357e-09 2.61130980e-11 0.00000000e+00
 1.06100442e-10 0.00000000e+00 1.58883176e-12 0.00000000e+00
 7.89452297e-13 0.00000000e+00 7.23305542e-12 0.00000000e+00
 1.93333415e-10 1.27552065e-07 0.00000000e+00 2.94480374e-10
 3.60786125e-10 6.29971226e-10 3.86407116e-10 0.00000000e+00
 2.91324055e-08 9.84055335e-11 7.68014278e-08 0.00000000e+00
 9.83389000e-01 7.19249464e-12 1.94818362e-04 8.99109920e-11
 8.79194486e-06 5.75878808e-10 4.34007343e-08 1.19458029e-09
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 8.58412038e-14 0.00000000e+00
 2.28357225e-13 1.61364380e-02 6.99707642e-12 0.00000000e+00
 1.93333415e-10 5.10985413e-06 0.00000000e+00 2.94468447e-10]
t = 11, alpha = [6.19964980e-10 1.03045856e-10 1.46987519e-12 5.19052012e-10
 3.26210602e-12 1.42026219e-09 3.13918070e-13 4.22148689e-06
 9.07517414e-14 4.31417751e-06 1.95956977e-13 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 7.12764703e-12
 3.94813094e-11 6.30551622e-10 9.50168718e-11 0.00000000e+00
 1.15188005e-09 0.00000000e+00 2.21278274e-09 0.00000000e+00
 1.58722835e-02 0.00000000e+00 2.45137052e-06 0.00000000e+00
 1.09488064e-07 1.16487993e-11 0.00000000e+00 4.42366340e-11
 3.77730646e-08 3.14523249e-11 1.05866421e-11 0.00000000e+00
```

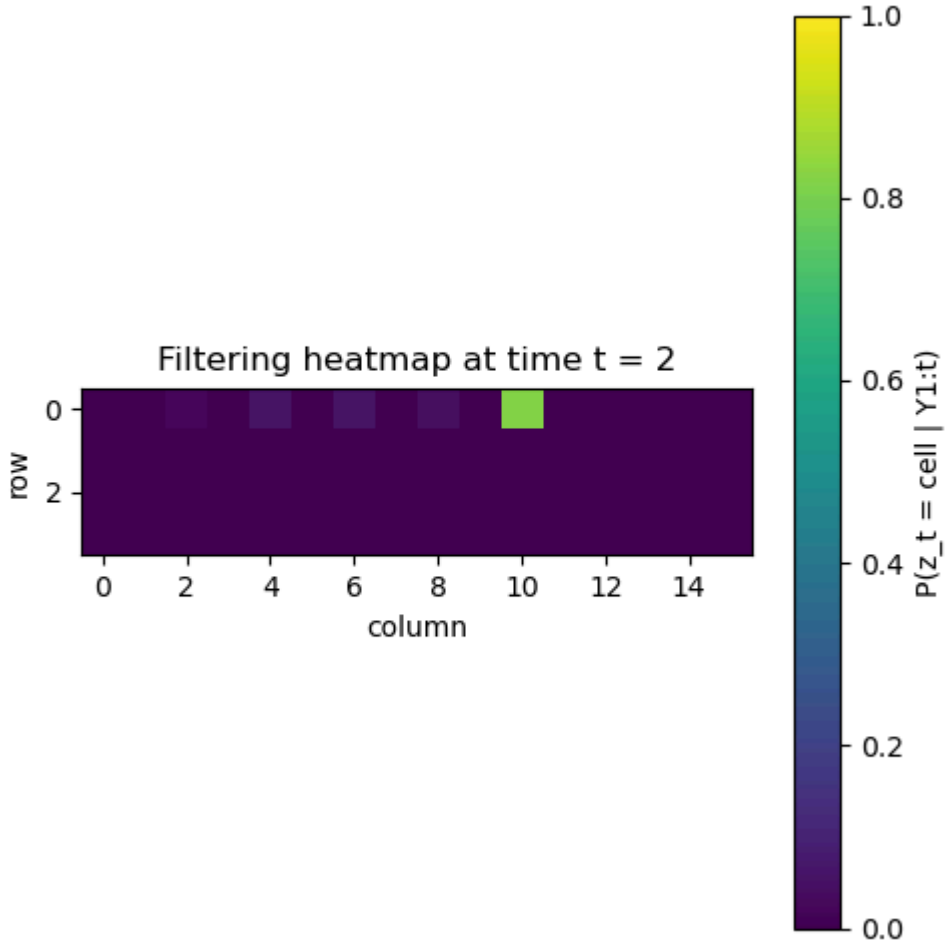
```
 2.96993595e-10 1.94440060e-09 2.42443400e-12 0.00000000e+00
 1.40694025e-13 1.61307999e-02 2.63670616e-12 2.46410717e-06
 1.85044893e-11 2.34218216e-07 2.62451952e-11 1.07186023e-09
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 1.23927550e-09 0.00000000e+00
 9.67706531e-01 2.90944288e-13 2.62736880e-04 0.00000000e+00
 1.38048269e-05 1.16487993e-11 0.00000000e+00 1.92758667e-11]
t = 12, alpha = [1.56625934e-12 1.54324731e-11 1.07324992e-11 5.20006817e-14
 5.09546216e-11 3.92976392e-14 6.96460237e-08 4.44699697e-15
 2.61776216e-04 4.22740416e-15 1.11521927e-07 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 7.29187741e-13
 3.82835933e-08 2.95605447e-12 5.38760362e-12 0.00000000e+00
 4.93143036e-12 0.00000000e+00 3.00922950e-14 0.00000000e+00
 2.54340351e-15 0.00000000e+00 2.49615811e-14 0.00000000e+00
 3.44528588e-13 3.73518201e-09 0.00000000e+00 1.20138664e-11
 7.79503802e-13 6.28015019e-10 1.38979513e-12 0.00000000e+00
 5.10384649e-11 4.92222502e-12 1.09382105e-10 0.00000000e+00
 9.83422745e-01 2.64748251e-14 1.81676492e-04 1.74243598e-13
 2.71909006e-07 9.69165540e-13 1.94218177e-09 1.79728723e-12
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 2.66425870e-14 0.00000000e+00
 4.74335918e-15 1.61330745e-02 2.49287105e-14 0.00000000e+00
 3.44528588e-13 2.29486382e-07 0.00000000e+00 1.17788851e-11]
t = 13, alpha = [6.17907284e-10 1.48097548e-11 2.02298740e-11 9.95245192e-13
 7.29319484e-12 1.12447431e-09 1.07214058e-13 4.22455825e-06
 1.62886829e-14 4.22613350e-06 7.06392117e-16 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 2.90743254e-13
 9.25381735e-14 6.28126449e-10 2.42558700e-13 0.00000000e+00
 2.05725313e-12 0.00000000e+00 1.12541696e-09 0.00000000e+00
 1.58705298e-02 0.00000000e+00 2.20104257e-06 0.00000000e+00
 3.38057974e-09 2.00650253e-14 0.00000000e+00 6.42906201e-14
 6.27790543e-10 8.82676267e-14 1.02191658e-11 0.00000000e+00
 2.38464764e-13 2.99990811e-12 1.21138782e-13 0.00000000e+00
 6.03482788e-16 9.67727512e-01 5.85132199e-15 2.20163386e-06
 3.26196441e-14 8.98129902e-09 4.07242135e-14 4.78628976e-11
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 1.76474461e-12 0.00000000e+00
 1.61265935e-02 1.14522154e-15 2.62485492e-04 0.00000000e+00
 8.84390319e-09 2.00650253e-14 0.00000000e+00 2.89970004e-14]
t = 14, alpha = [2.36816226e-13 2.25367579e-11 2.62923792e-13 4.37376266e-13
 2.68767339e-11 1.17601828e-13 1.00753966e-07 1.96261428e-15
 3.78505532e-04 2.75685901e-16 1.53203897e-07 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 1.53248852e-15
 2.23079128e-09 2.42073473e-13 8.05136230e-12 0.00000000e+00
 1.21582320e-13 0.00000000e+00 3.62881456e-15 0.00000000e+00
 2.68437812e-16 0.00000000e+00 8.65768552e-17 0.00000000e+00
 8.67063281e-16 1.87625636e-10 0.00000000e+00 7.74462622e-13
 2.87323214e-15 2.26944590e-11 5.25724746e-15 0.00000000e+00
 1.20547003e-13 7.60930664e-15 2.69820823e-11 0.00000000e+00
 9.68467338e-01 9.75277797e-17 1.53847721e-02 4.58514118e-16
 5.28785437e-08 2.31608976e-15 1.07803726e-10 3.88562387e-15
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 1.92504799e-15 0.00000000e+00
 2.77891166e-17 1.57690750e-02 8.79376773e-17 0.00000000e+00
 8.67063281e-16 3.17854283e-10 0.00000000e+00 7.60601791e-13]
t = 15, alpha = [3.64907141e-11 7.77030464e-13 3.03583657e-11 4.39503470e-13
 9.86068388e-13 1.63205928e-09 1.79550819e-13 6.13120395e-06
 3.65350238e-15 6.13329384e-06 8.79980177e-18 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 1.88126365e-14
 8.76247681e-15 3.69886163e-11 7.32565929e-15 0.00000000e+00
```

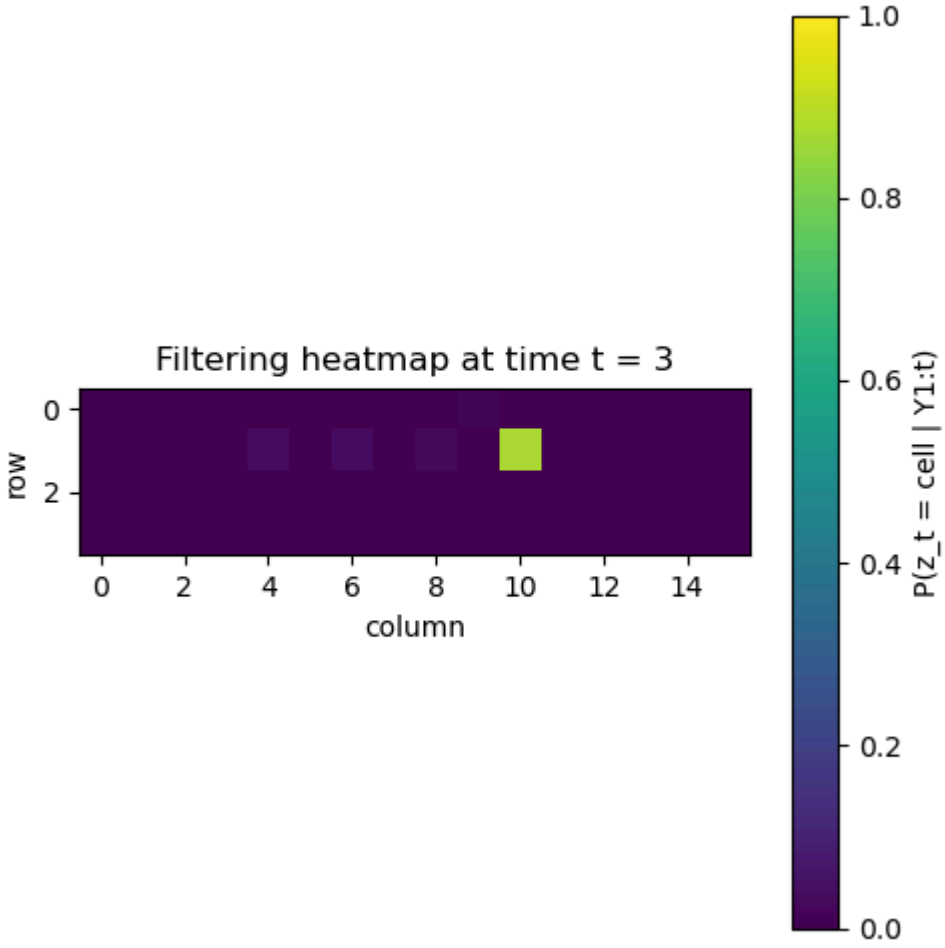
```
 4.38173878e-13 0.00000000e+00 1.63206098e-09 0.00000000e+00
 1.56896271e-02 0.00000000e+00 1.86861110e-04 0.00000000e+00
 6.46799637e-10 4.91923336e-17 0.00000000e+00 1.37376355e-16
 3.64932680e-11 3.13762836e-15 4.97902196e-13 0.00000000e+00
 3.13822132e-15 4.39879905e-13 3.66511097e-16 0.00000000e+00
 8.77509011e-18 9.67543296e-01 1.69564125e-17 1.86858030e-04
 8.13922165e-17 6.57139387e-10 9.10546593e-17 2.67444778e-12
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 4.36951672e-13 0.00000000e+00
 1.59388642e-02 4.39052291e-18 4.42224028e-04 0.00000000e+00
 6.49963049e-10 4.91923336e-17 0.00000000e+00 6.29243446e-17]
t = 16, alpha = [1.27698991e-14 1.83369304e-12 2.34600873e-14 5.09377521e-13
 3.98052192e-11 1.89424451e-14 1.49536603e-07 2.97724829e-15
 3.82756502e-04 5.95874730e-17 4.70451997e-06 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 3.34874894e-18
 2.22991670e-12 1.29399377e-14 9.56315770e-13 0.00000000e+00
 1.61010685e-14 0.00000000e+00 2.92383149e-15 0.00000000e+00
 5.95155687e-17 0.00000000e+00 4.21176637e-19 0.00000000e+00
 2.19116183e-18 2.37760495e-11 0.00000000e+00 4.43795470e-14
 1.93388276e-16 8.11523660e-11 1.70038595e-16 0.00000000e+00
 2.14038412e-14 8.24548820e-17 3.98158915e-11 0.00000000e+00
 1.64945096e-02 4.20622647e-19 1.57434088e-02 1.19869561e-18
 4.55497655e-06 5.60988676e-18 8.05282771e-12 8.63608431e-18
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 5.95616158e-18 0.00000000e+00
 2.13954009e-19 9.67369916e-01 2.78018903e-19 0.00000000e+00
 2.19116183e-18 2.38531623e-11 0.00000000e+00 4.34623761e-14]
t = 17, alpha = [1.35453658e-12 1.74399631e-14 1.18469170e-12 1.32762264e-11
 2.72210517e-13 4.98588029e-08 1.24217624e-14 1.27635346e-04
 1.54817567e-15 1.29937761e-04 3.00043248e-17 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 2.21897735e-14
 9.71662811e-15 2.87240972e-11 1.11400328e-14 0.00000000e+00
 1.32791083e-11 0.00000000e+00 4.98588064e-08 0.00000000e+00
 5.62575537e-03 0.00000000e+00 3.93820447e-03 0.00000000e+00
 1.13875603e-06 2.49805261e-18 0.00000000e+00 6.22744371e-18
 2.77940942e-11 3.41669786e-15 2.73695606e-11 0.00000000e+00
 8.09176169e-15 1.32826657e-11 1.50909935e-15 0.00000000e+00
 3.00049689e-17 3.31890661e-01 1.08915313e-18 3.93699095e-03
 4.19298133e-18 1.13877198e-06 4.28116646e-18 4.09206601e-12
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 1.32719638e-11 0.00000000e+00
 3.27954808e-01 3.86194005e-19 3.26392491e-01 0.00000000e+00
 1.13875606e-06 2.49805261e-18 0.00000000e+00 2.87869477e-18]
t = 18, alpha = [3.37385485e-16 3.07605593e-13 2.47764955e-13 1.81001237e-14
 9.29641481e-10 3.53618755e-15 2.38041418e-06 1.73558394e-16
 1.09639108e-04 1.97932255e-17 7.58120999e-05 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 1.16051888e-19
 8.10845784e-13 5.18234016e-16 7.92410079e-13 0.00000000e+00
 3.53265773e-15 0.00000000e+00 1.73072921e-16 0.00000000e+00
 1.96068511e-17 0.00000000e+00 5.69295820e-19 0.00000000e+00
 8.56218770e-20 3.18321794e-08 0.00000000e+00 5.16656596e-14
 1.63164571e-16 7.77390162e-11 1.80848531e-16 0.00000000e+00
 4.94993478e-13 1.69543099e-16 9.29888864e-10 0.00000000e+00
 1.03397717e-02 3.87719339e-19 1.03525766e-02 4.92177612e-20
 7.34210763e-05 2.11956406e-19 1.06108764e-08 3.03125979e-19
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 1.87486055e-17 0.00000000e+00
 3.77570850e-19 9.79046323e-01 1.49464485e-20 0.00000000e+00
 8.56218770e-20 3.18321801e-08 0.00000000e+00 5.08386219e-14]
t = 19, alpha = [1.82688315e-14 4.06161806e-15 1.84111741e-14 1.51888497e-11
```

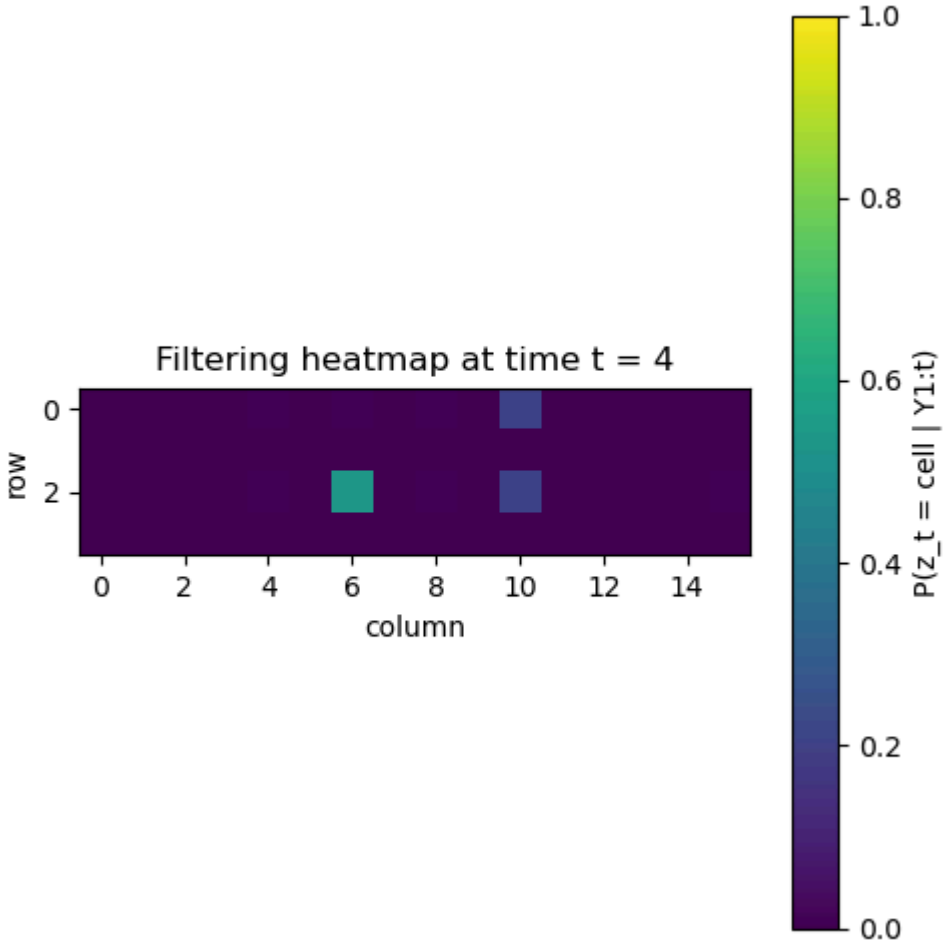
```
6.16666484e-16 3.88969693e-08 9.51331879e-17 1.82973156e-06
5.21770880e-18 3.64832678e-06 4.98903409e-19 0.00000000e+00
0.00000000e+00 0.00000000e+00 0.00000000e+00 1.26586356e-15
1.86126503e-17 1.30100442e-12 4.05778274e-15 0.00000000e+00
1.51969306e-11 0.00000000e+00 3.88969733e-08 0.00000000e+00
1.70681111e-04 0.00000000e+00 1.28682040e-04 0.00000000e+00
9.00227114e-07 4.69440556e-21 0.00000000e+00 1.06380615e-20
1.28303672e-12 1.47773286e-17 7.69641356e-11 0.00000000e+00
9.07078541e-17 1.52009714e-11 9.31317805e-18 0.00000000e+00
4.95972596e-19 1.62874983e-02 2.18534677e-20 1.27724010e-04
7.99811997e-21 9.01267014e-07 7.54785098e-21 2.59981504e-10
0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
0.00000000e+00 0.00000000e+00 1.51888435e-11 0.00000000e+00
1.61606737e-02 1.59501148e-20 9.67116483e-01 0.00000000e+00
9.00227114e-07 2.81664333e-19 0.00000000e+00 4.95127239e-21]
```

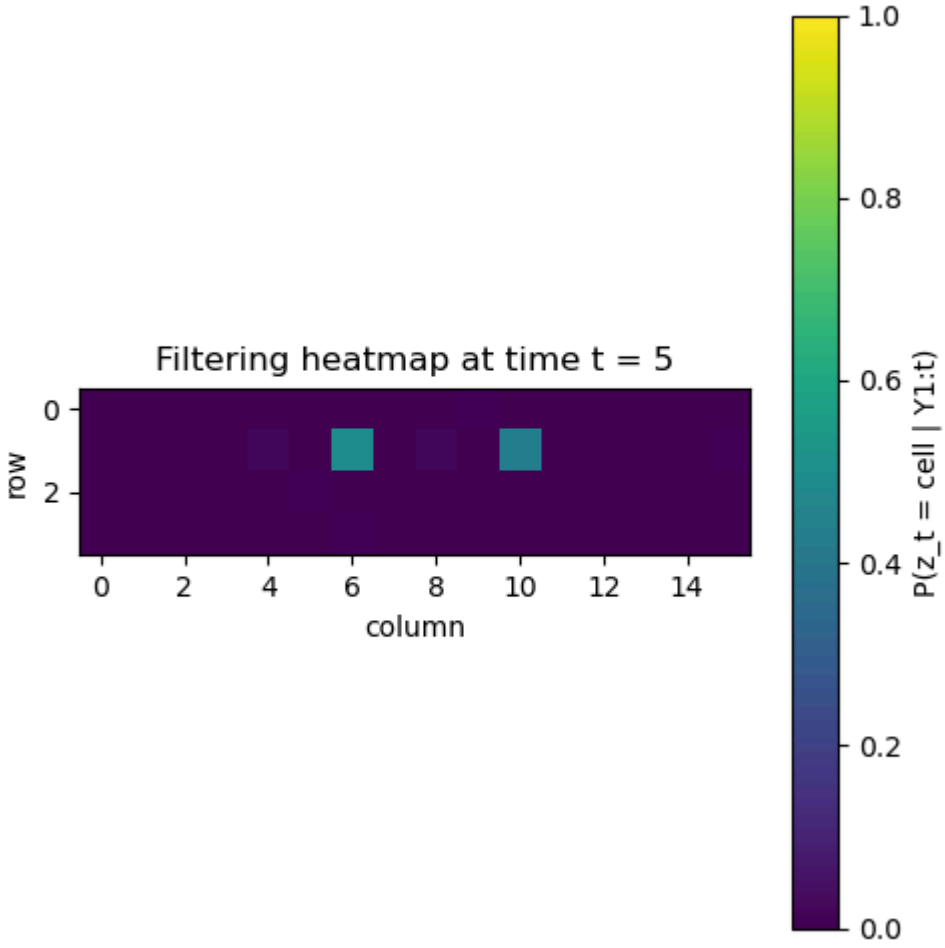


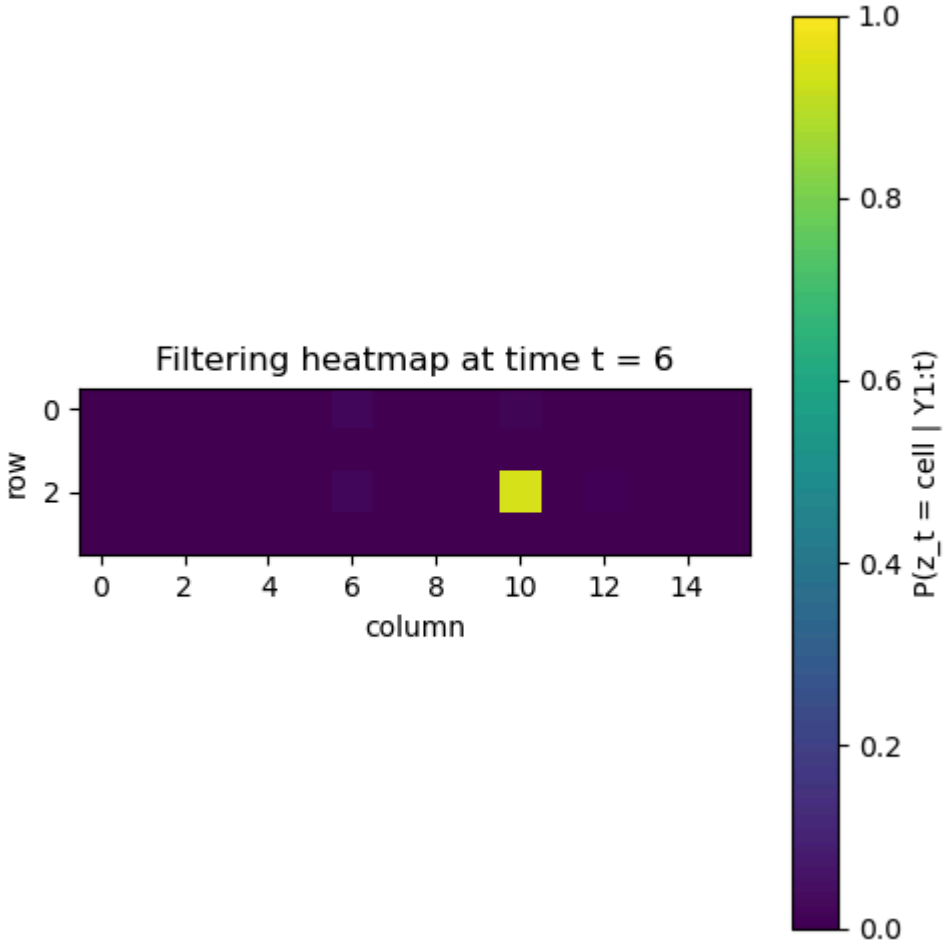

Filtering heatmap at time t = 1
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

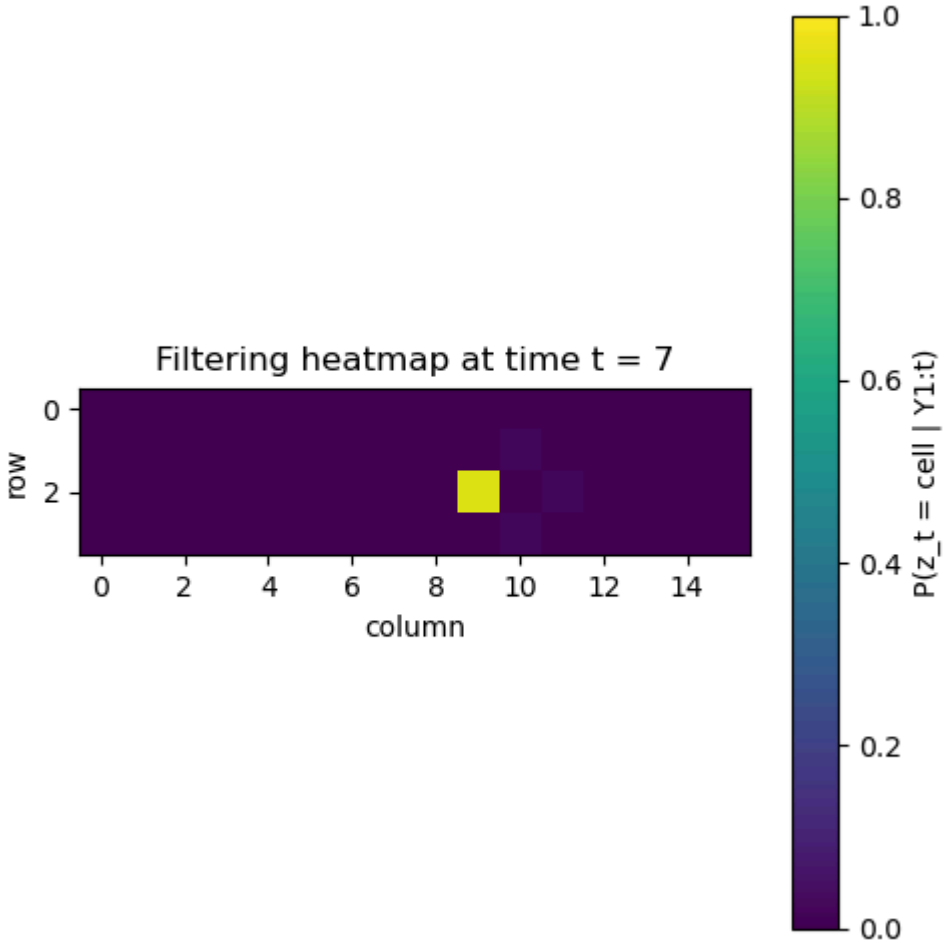

Filtering heatmap at time t = 2
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

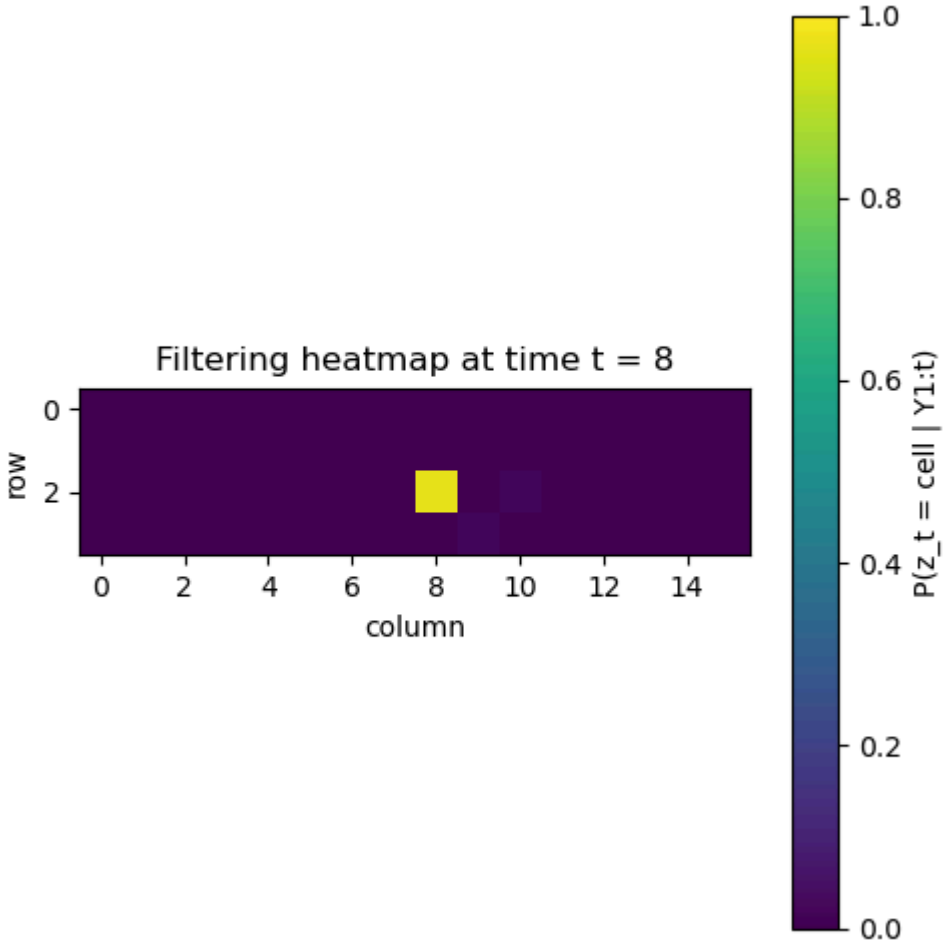

Filtering heatmap at time t = 3
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

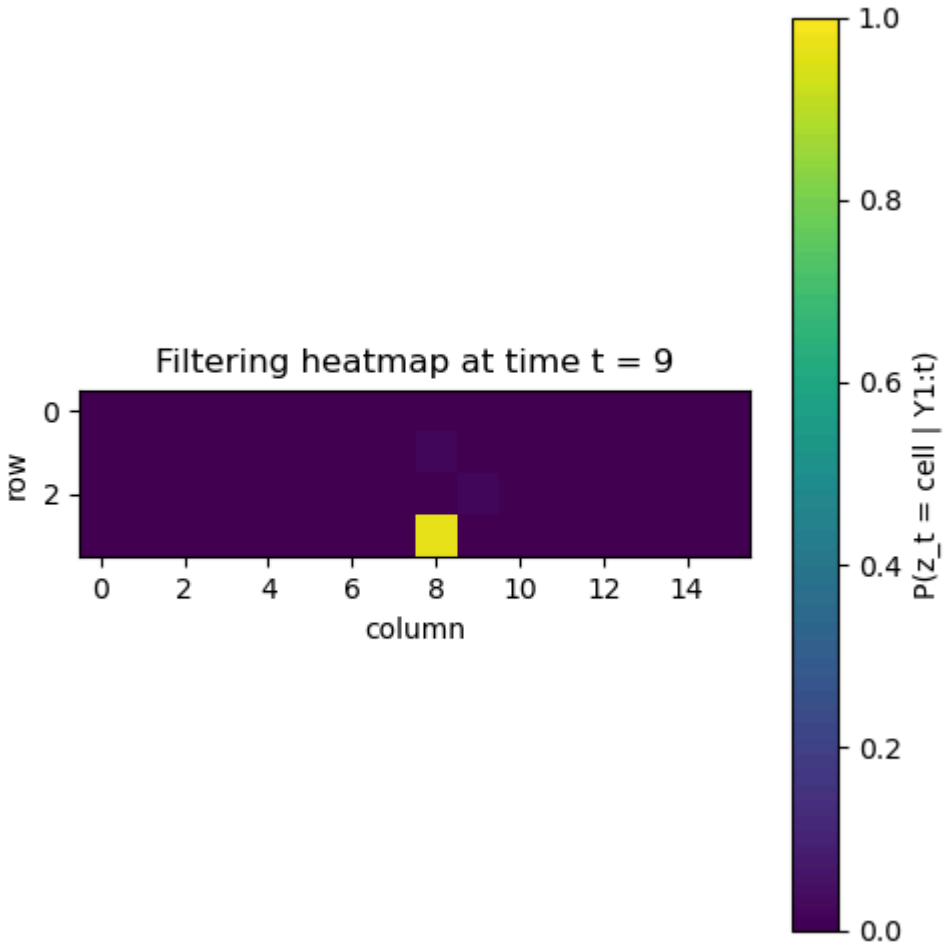

Filtering heatmap at time t = 4
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

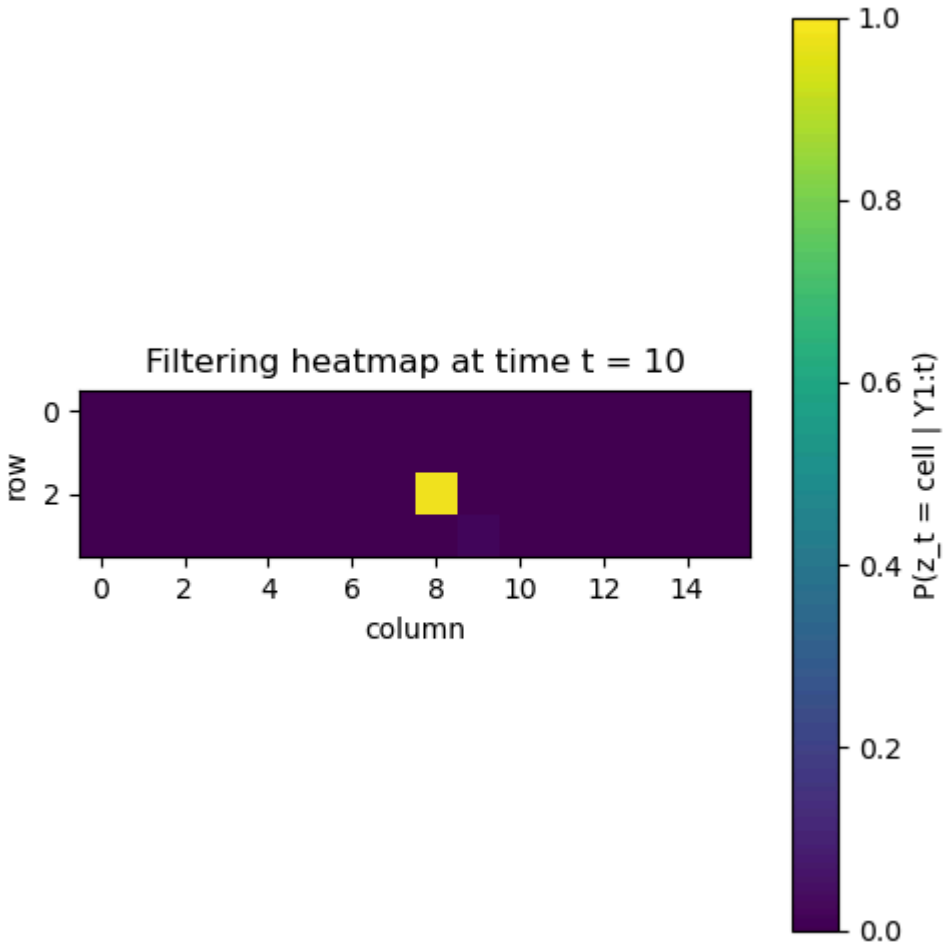

Filtering heatmap at time t = 5
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

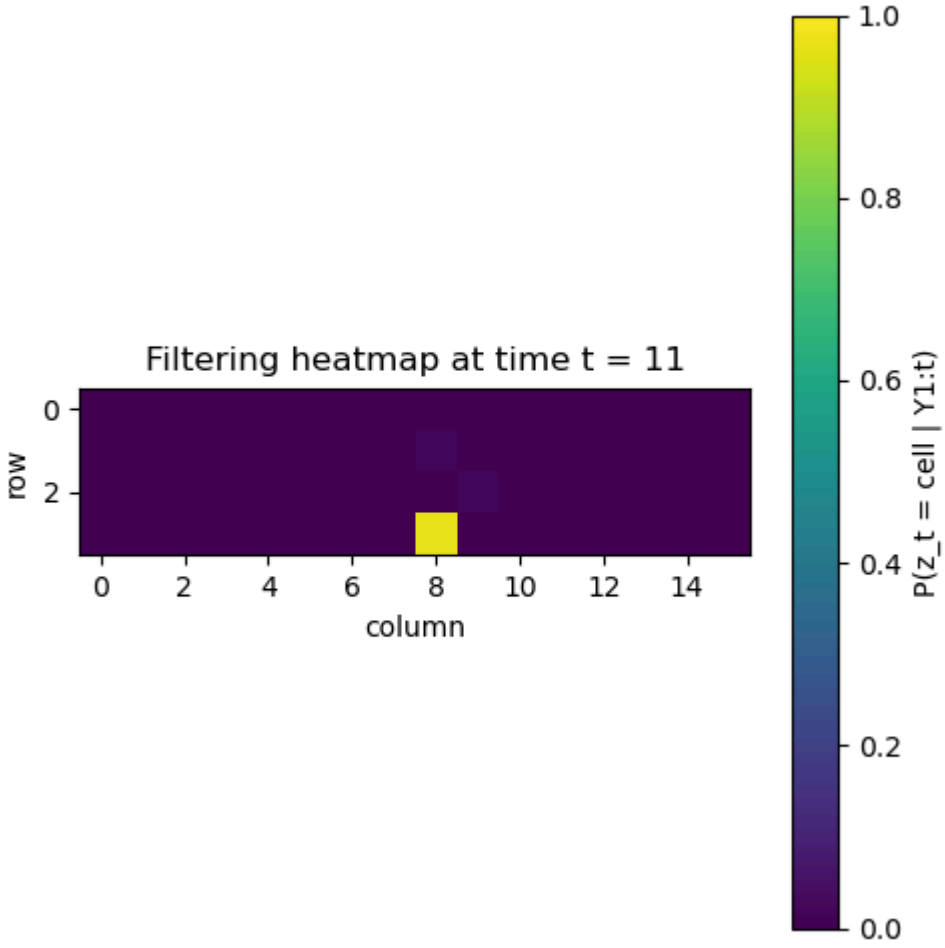

Filtering heatmap at time t = 6
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

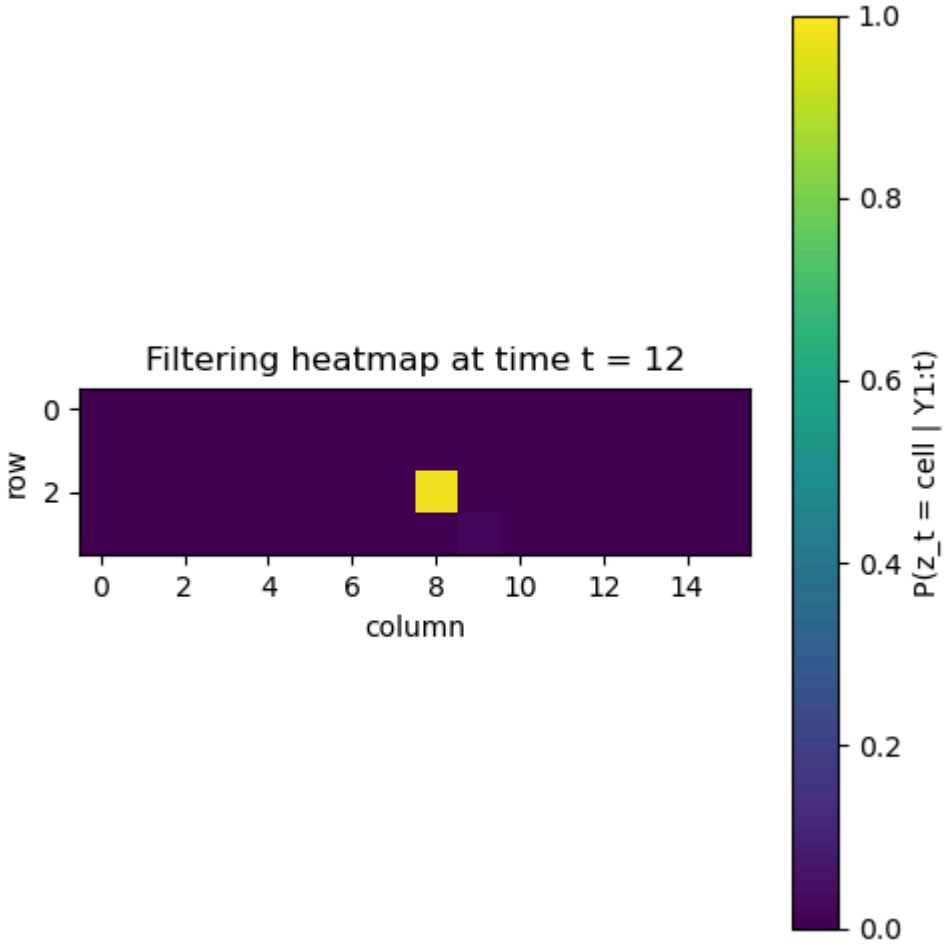

Filtering heatmap at time t = 7
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

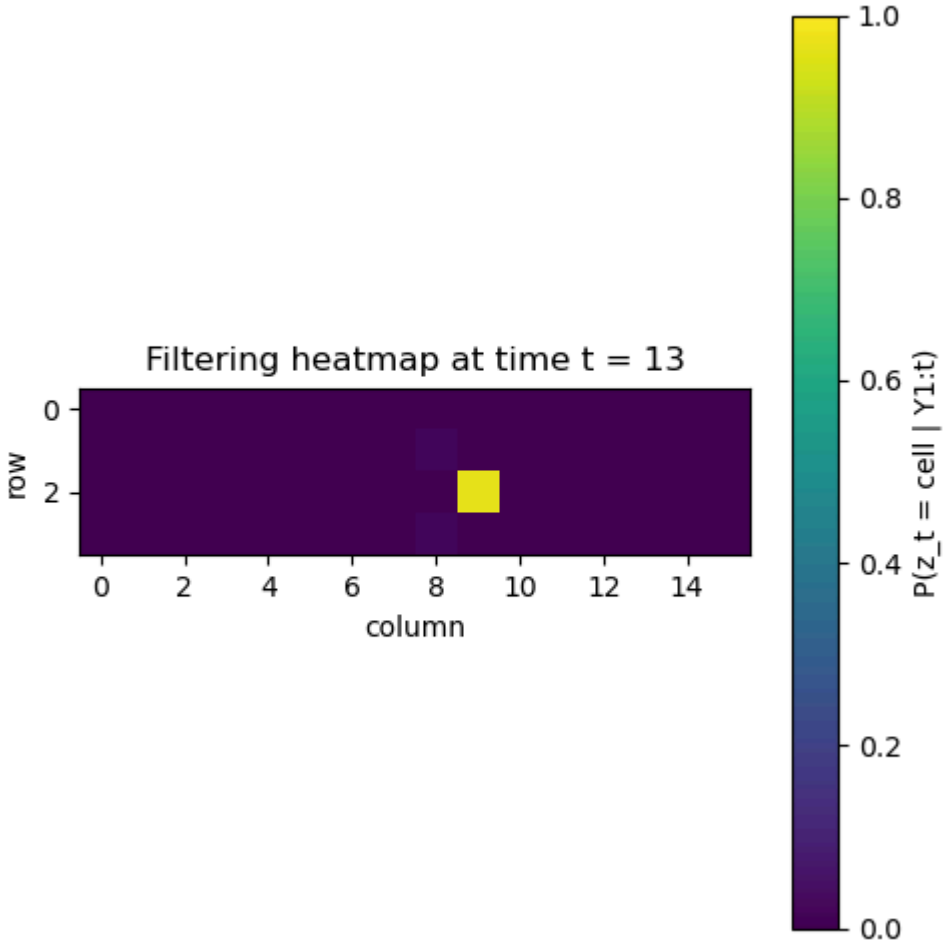

Filtering heatmap at time t = 8
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

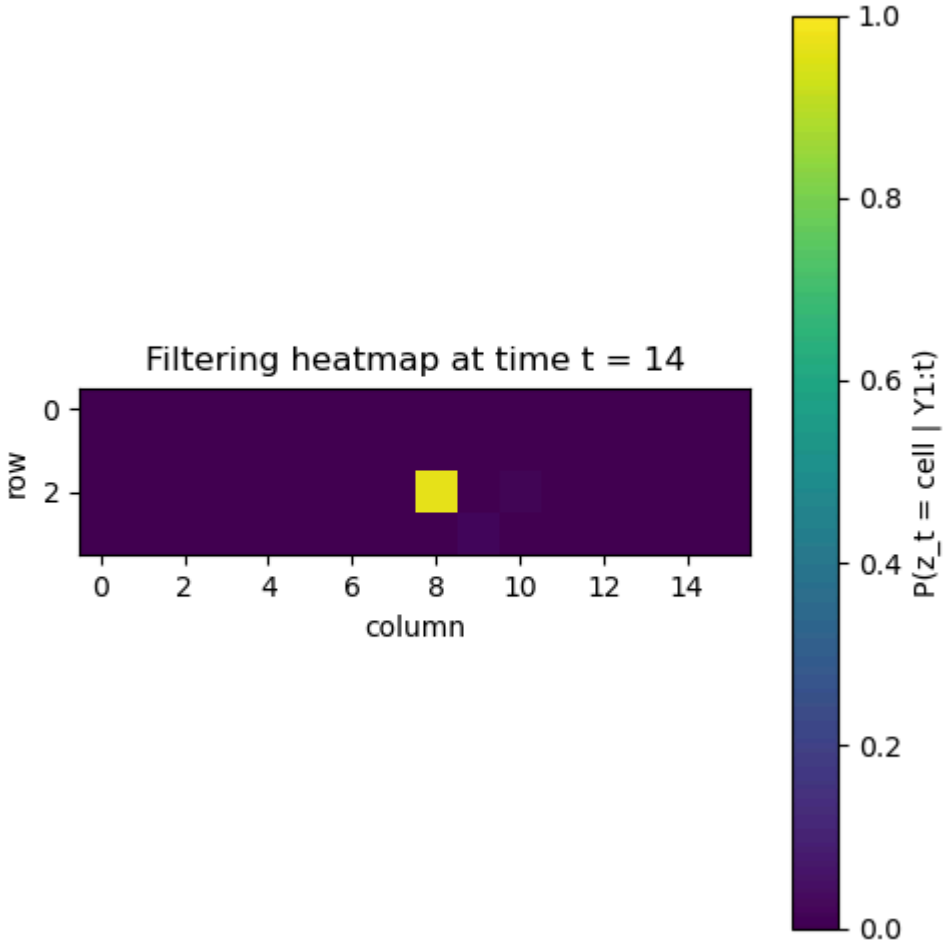

Filtering heatmap at time t = 9
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

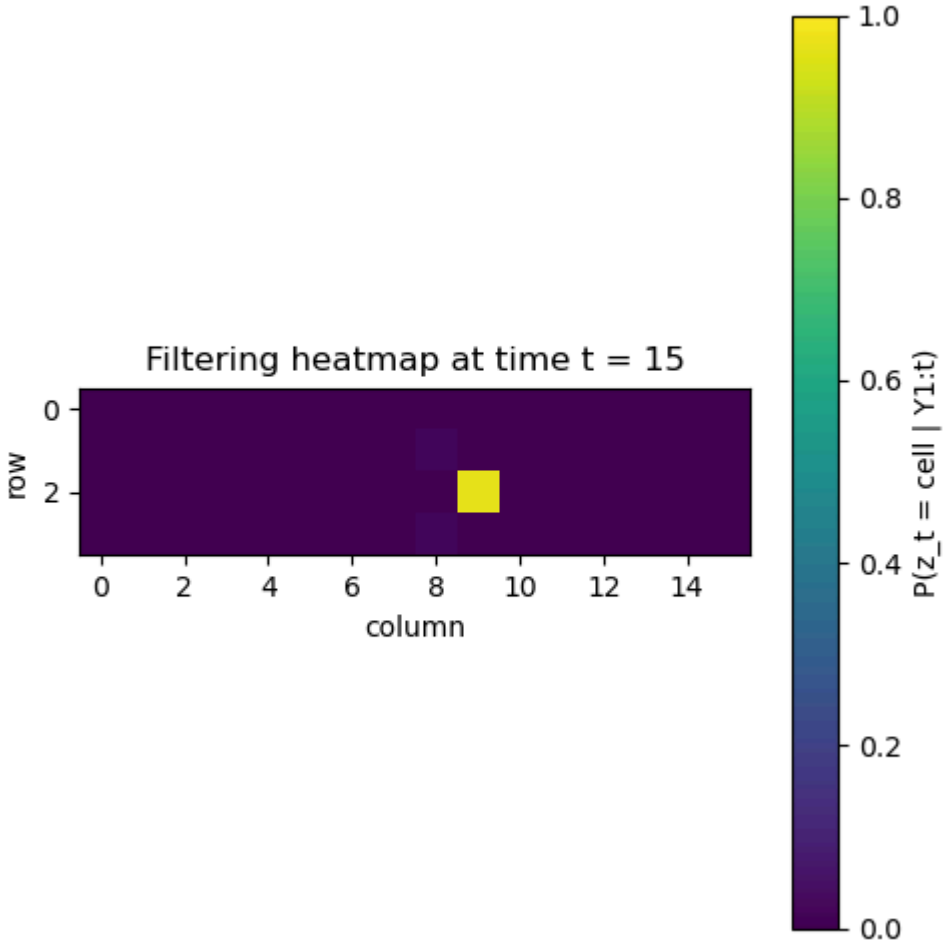

Filtering heatmap at time t = 10
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

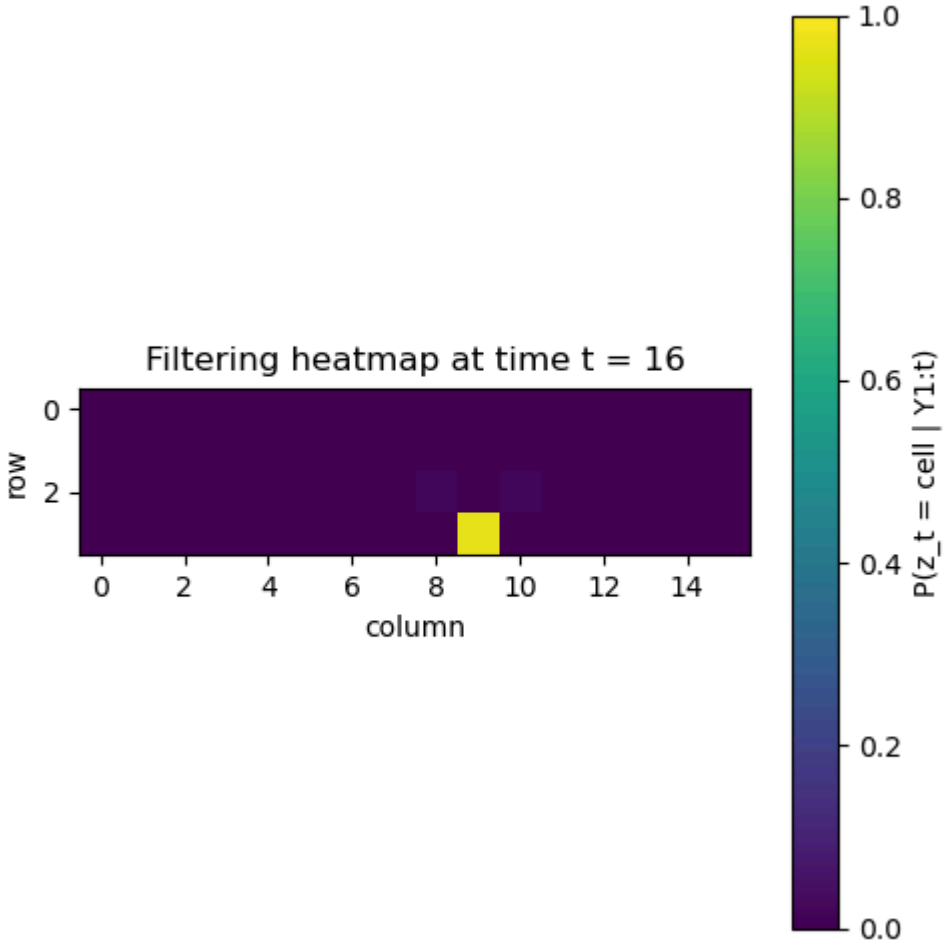

Filtering heatmap at time t = 11
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

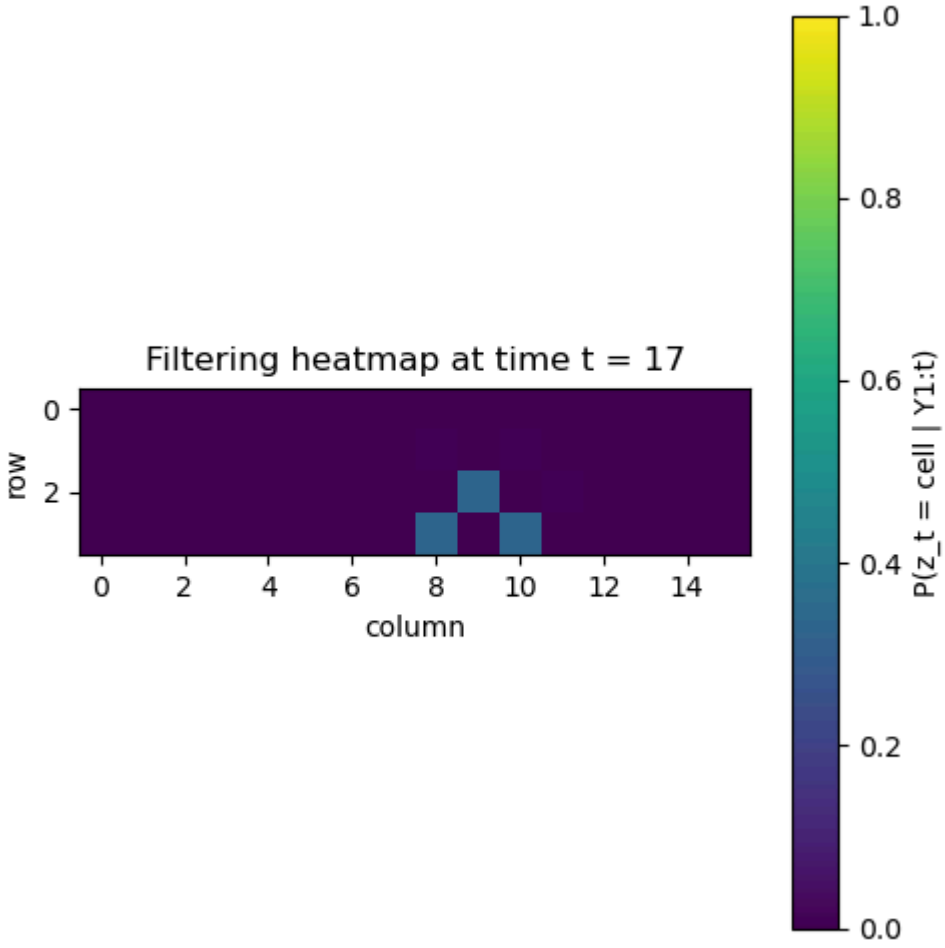

Filtering heatmap at time t = 12
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

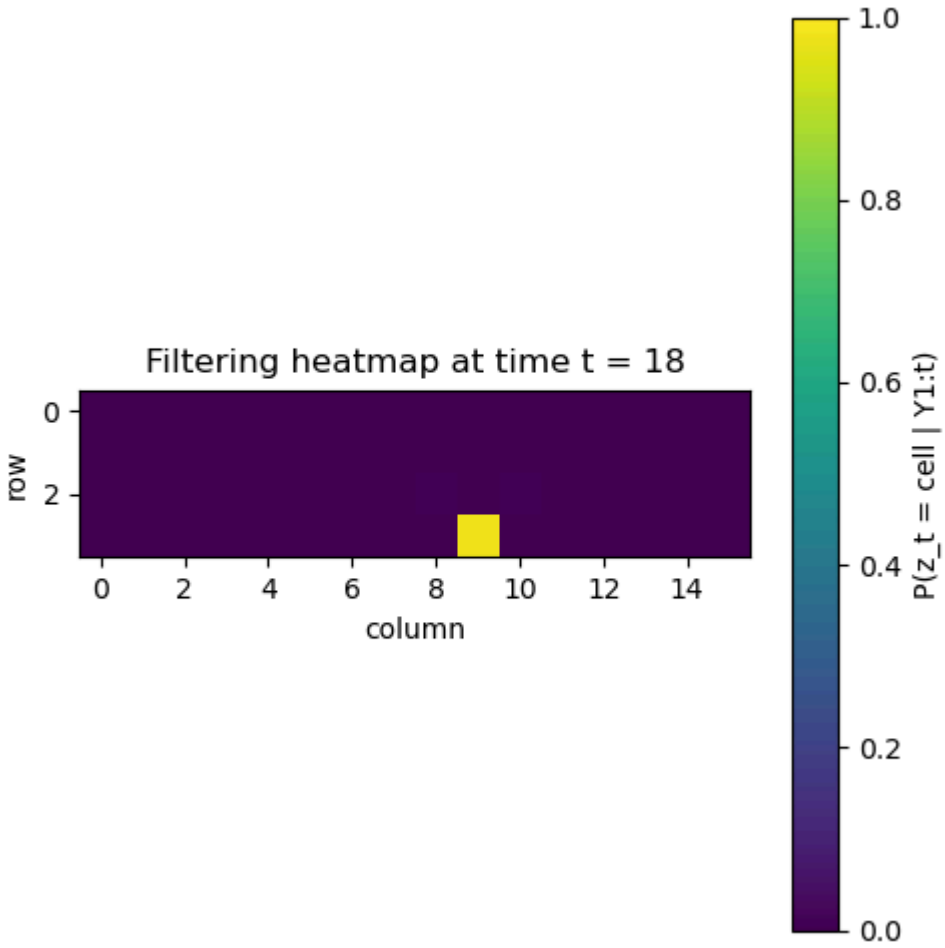

Filtering heatmap at time t = 13
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0

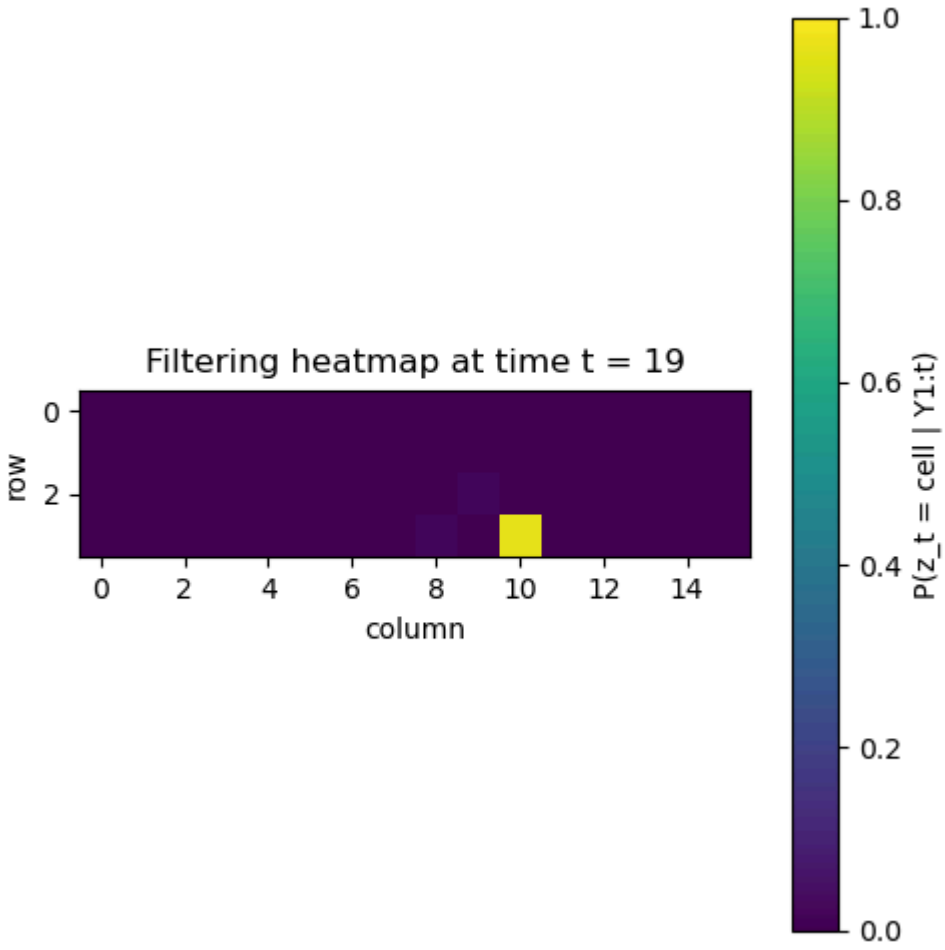

Filtering heatmap at time t = 14
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0


Filtering heatmap at time t = 15
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0


Filtering heatmap at time t = 16
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0


Filtering heatmap at time t = 17
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0


Filtering heatmap at time t = 18
row
column
0
2
0
2
4
6
8
10
12
14
P(z_t = cell | Y1:t)
0.0
0.2
0.4
0.6
0.8
1.0



Noise in observations eliminates the possibility of having complete certainty about a reading.
In our case, we assume a noise level of $\epsilon = 0.2$, which means that the probability of receiving the *correct* observation is $1 - \epsilon$.

The remaining probability (0.2) is **divided equally** among all possible *incorrect* observations.

This introduces uncertainty into the observation model (the matrix B), as the agent may receive an incorrect reading. As explained above, the value of (Z[t]) is sampled "randomly" according to the probabilities of the observation.
This means that the agent may select an incorrect observation, thus influencing all subsequent belief updates.

After modifying the observation matrix (B) to include noise, we recalculate the results of points **2.1** and **2.2** to observe how noise influences the agent's choices.

As before, heatmaps help us visualise the evolution of the probability distribution and understand how the sequence of observations conditions the estimated position.

When a new sequence is generated, that will possibly contain at least one error. The likelihood is recomputed and it will be lower than the previously computed one, than viterbi will esteem as most proable path a path that will be slightly different than the original because errors interfeers with that computation.

Finally with the use of the filtering when an error is committed it can be seen by the creation of a "flower" around the last cell, or the small shadow around some cells that are mostly unprobable due to an improbable error.

3 Analysis and discussion

The code is kept as easy and correlated to the theory as possible. We decided to take the code saw in class and in some cases modify it to stay more aligned with the theory, so without the use of vectors and similar used in faster and more efficient versions of the algorithm. As for the creation of the maze and the instantiation of the problem we have a general code that simply changing the obstacle list can change the maze and use another layout.

For the results of each algorithm, they are as expected. The more interesting results are the Viterbi and the heatmaps created by the filtering. In the cases depicted above there could be mismatch even if the measurement is perfect due to some pattern in the maze, for example a cell neighbouring with 2 cells with the same observation. For the heatmap it can be observed the behaviour created by the noise, when a wrong reading is saved (due to noise) it's created a sort of "flower" around the last cell for example t=17.