**Problem 1**

Copilot

**(a) Prediction step: compute the distribution of X1.** The forward map is given by

,

where

and

Therefore,

**(b) Filtering step: compute the distribution of X1 conditioned on Y1 = 2.**

The observation operator is given by

,

where

.

Therefore,

.

From Bayes’ theorem,

**(c) Smoothing step: compute the distribution of X0 conditioned on Y1 = 2.**

We have

Therefore,

**(d) Pseudocode for the distribution of X1, X1|Y1, and X0|Y1.**

Given the inputs a, B, m and y, the pseudocode that will return as output the distribution of X1, X1 conditioned on Y1 = y, and X0 conditioned on Y1 = y is as follows:

# Inputs: a, B, m, y

# Outputs: X1, X1|Y1, X0|Y1

# X1 = a\*X0 + B\*E1

# X1|Y1 = ((a^2\*m + B^2)\*y + B^2\*m)/(a^2 + B^2)

# X0|Y1 = ((X1|Y1 - B\*E1)/a, m/a^2)

**Problem 2**

import numpy as np

import matplotlib.pyplot as plt

# Constants

N = 1000

a = 7

B = 1

m = -1

y = 2

# (a) Prediction step

E1 = np.random.normal(0, 1, N)

X0 = np.random.normal(m, np.sqrt(2), N)

X1 = a \* X0 + B \* E1

# (b) Plot the histogram of particles

plt.hist(X1, bins=30, density=True)

plt.title('Histogram of particles')

plt.xlabel('X1')

plt.ylabel('Density')

plt.show()

# (c) Compute the likelihood Wi

Wi = (1 / np.sqrt(2 \* np.pi)) \* np.exp(-0.5 \* (y - X1)\*\*2)

# (d) Filtering step

wi = Wi / np.sum(Wi)

# (e) Compute the weighted mean and the variance

m1 = np.sum(wi \* X1)

V1 = np.sum(wi \* (X1 - m1)\*\*2)

print('Weighted mean:', m1)

print('Variance:', V1)

# (f) Compute the effective sample size

ESS = 1 / np.sum(wi\*\*2)

print('Effective sample size:', ESS)

This code first sets up the model parameters and generates the particles for X1. It then computes the likelihood Wi for each particle and normalizes these to obtain the weights wi. The weighted mean and variance are then computed using these weights. Finally, the effective sample size is computed as the reciprocal of the sum of the squares of the weights.

(c) by hand:  
the likelihood

by hand for the observation Y1​=2Y1​=2

Given that

where

we know that

follows a normal distribution

Therefore, the likelihood of observing

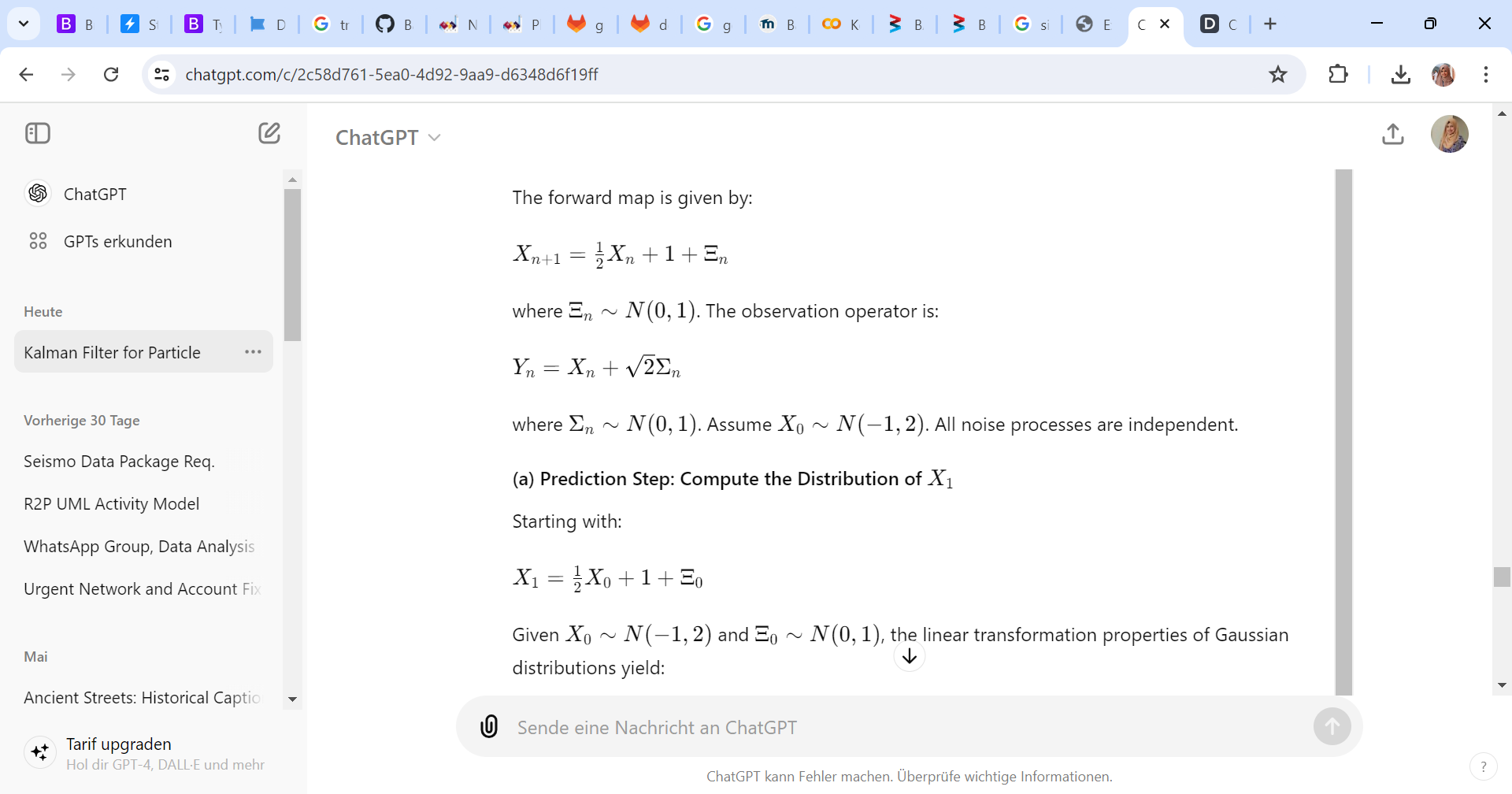
given X1​=xi​

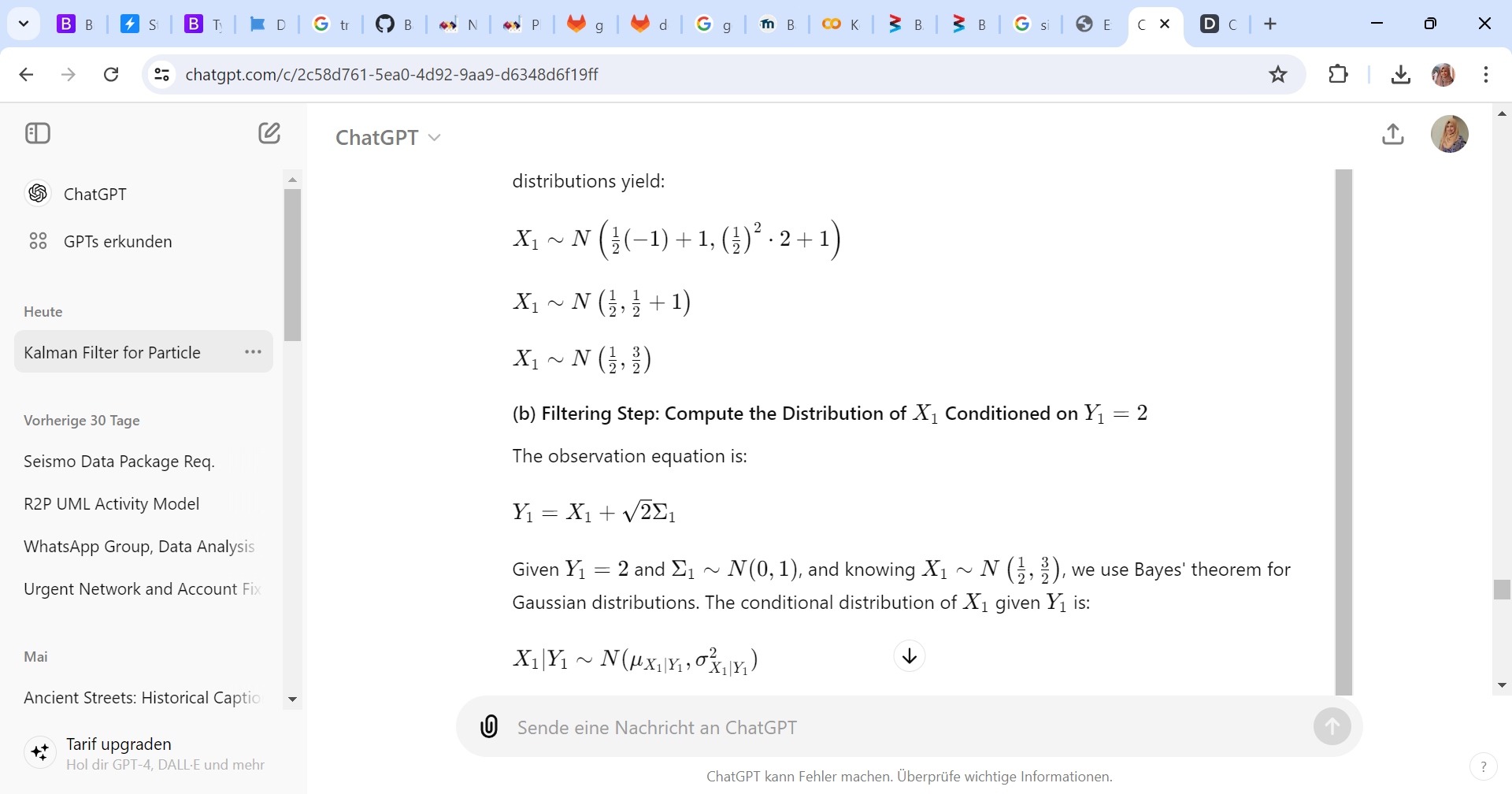
is given by the PDF of a normal distribution:

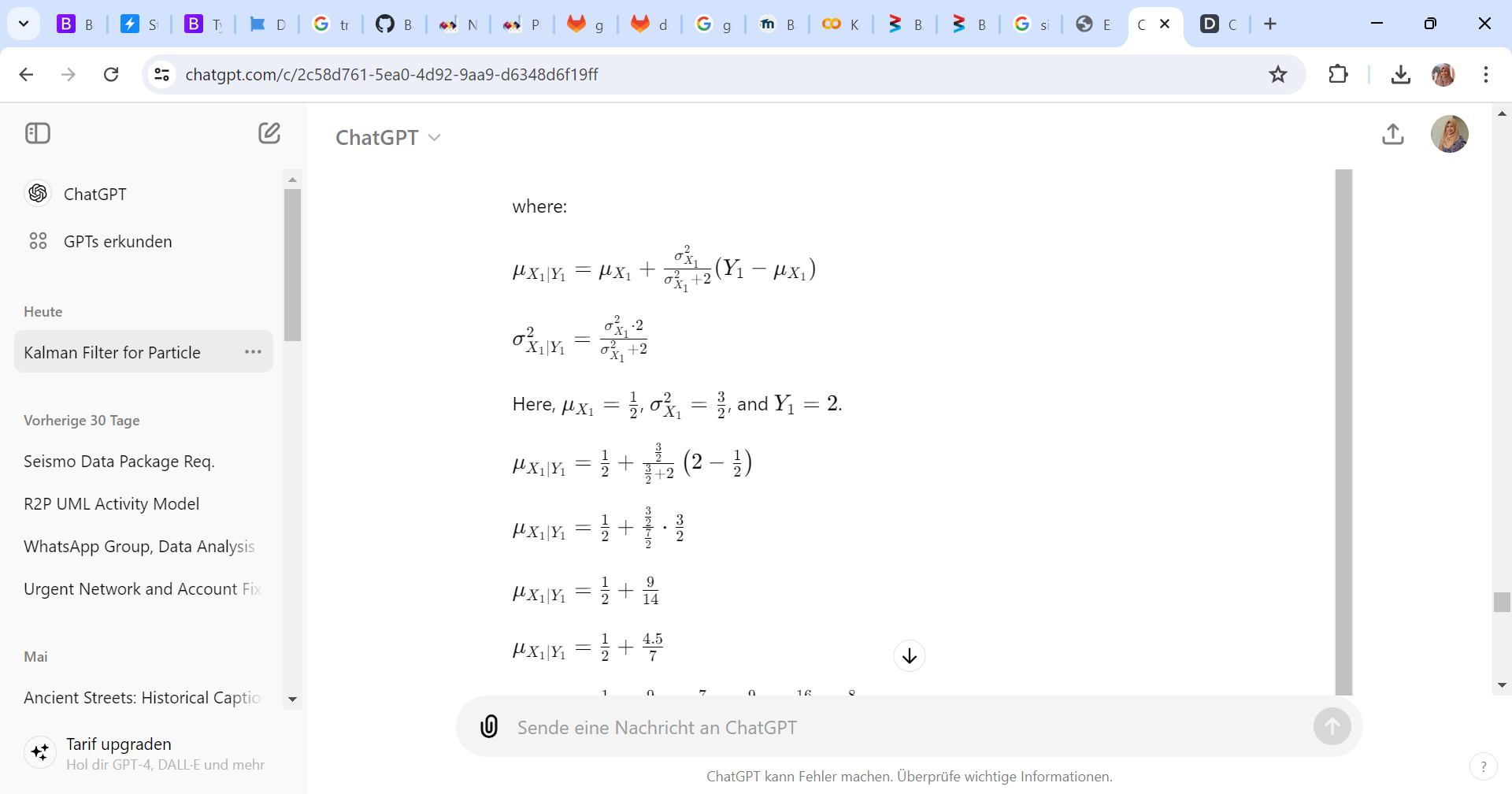
This is the formula for the likelihood W\_iWi​. You can substitute each x\_ixi​ into this formula to get the corresponding W\_iWi​.

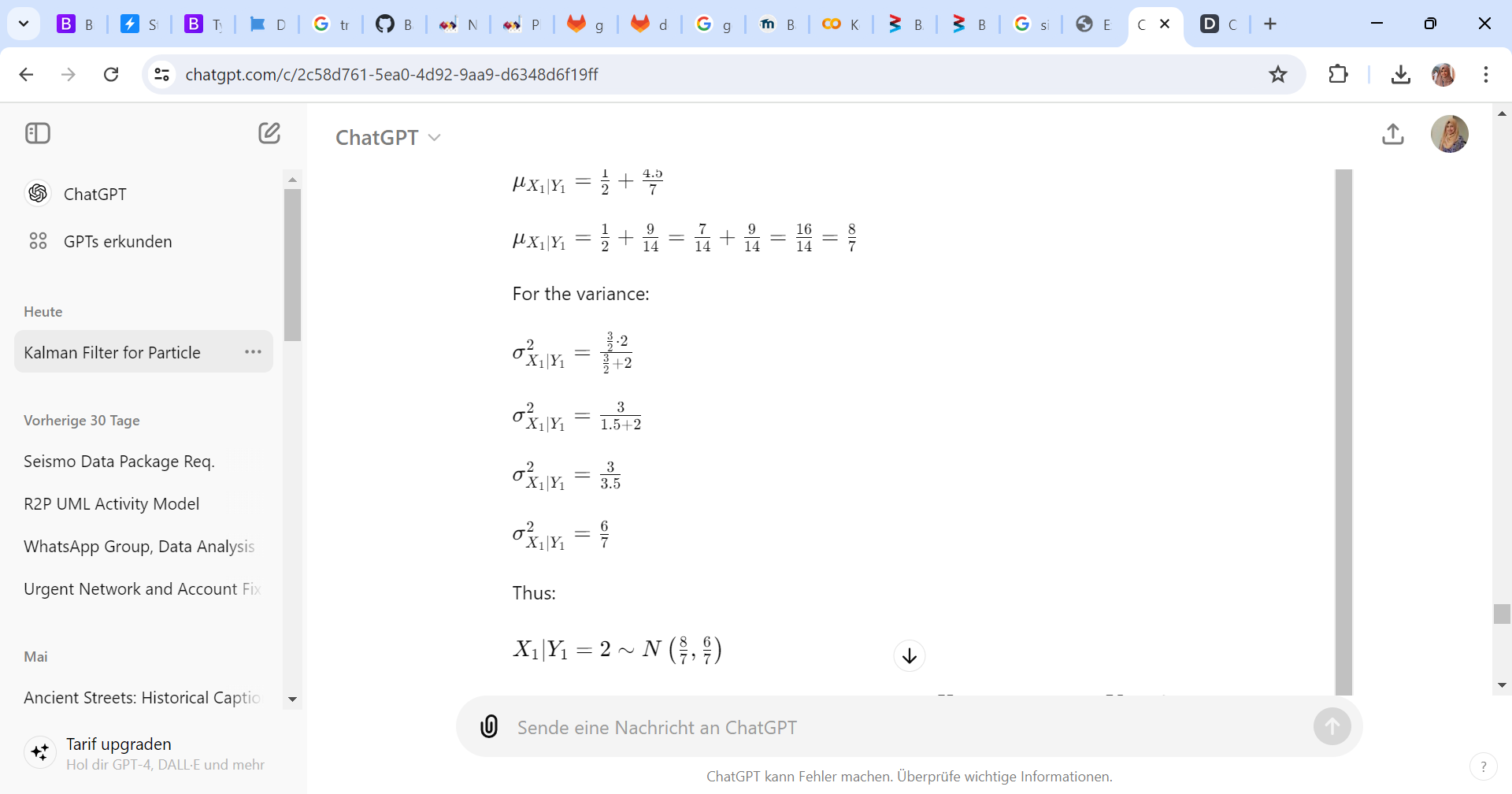
ChatGPT

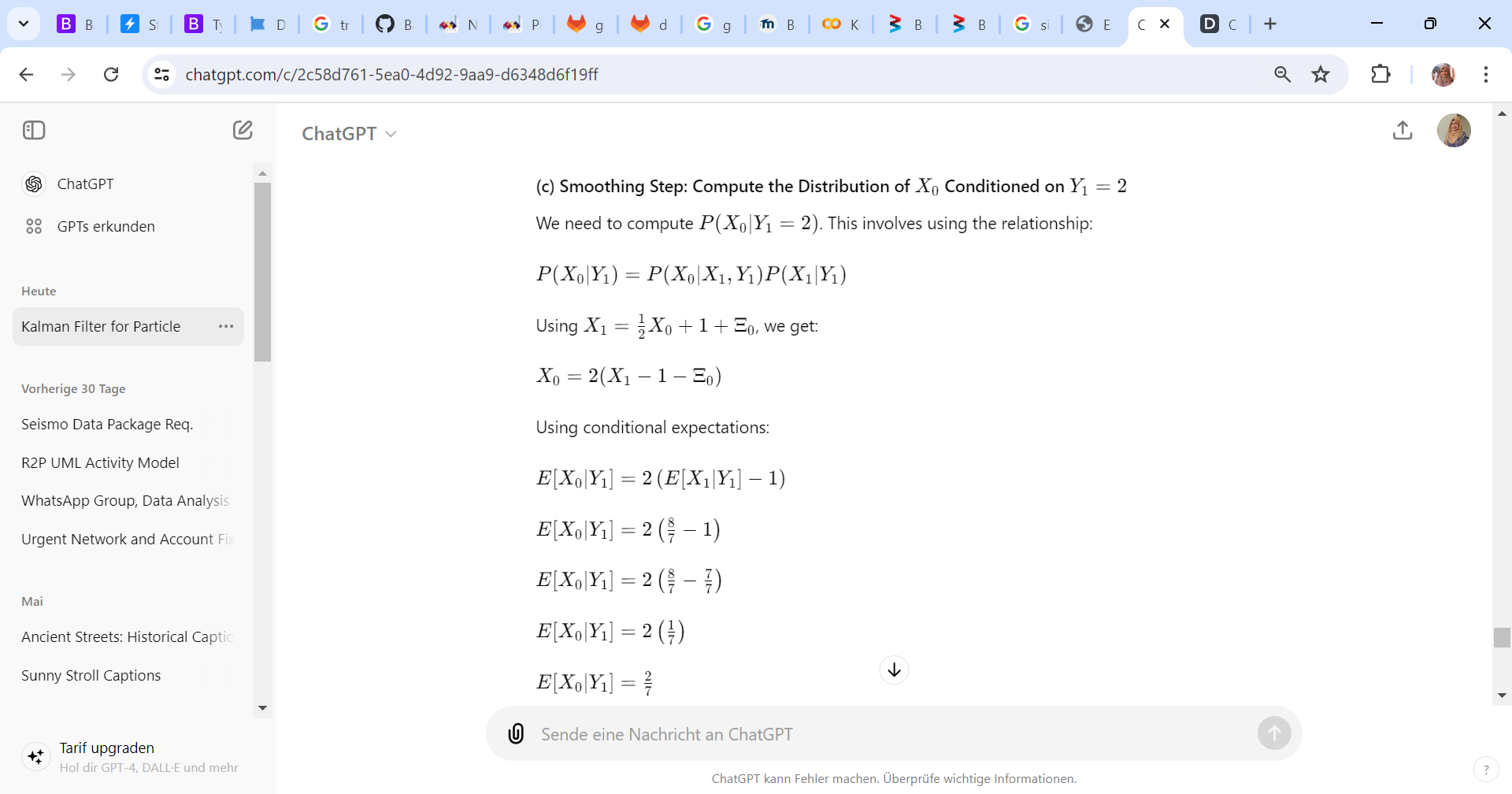
**Problem 1**

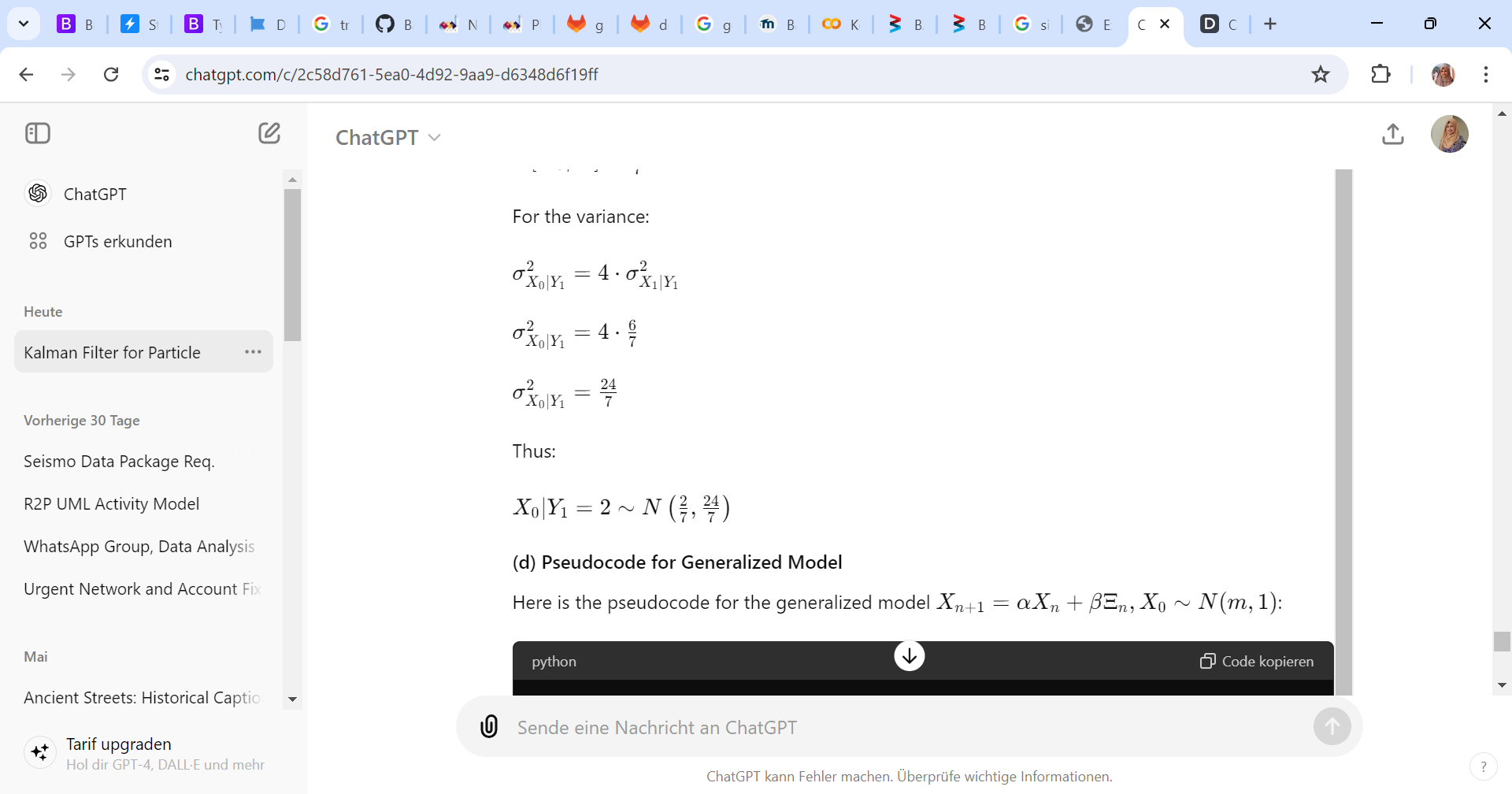












(d) Pseudocode for Generalized Model

Here is the pseudocode for the generalized model \(X\_{n+1} = \alpha X\_n + \beta \Xi\_n, X\_0 \sim N(m, 1)\):

```python

def kalman\_filter(alpha, beta, m, y):

# Prediction Step

mu\_X1 = alpha \* m + 1

sigma\_X1 = (alpha\*\*2) \* 1 + beta\*\*2

# Filtering Step

mu\_Y1 = mu\_X1

sigma\_Y1 = sigma\_X1 + 2

K = sigma\_X1 / sigma\_Y1

mu\_X1\_given\_Y1 = mu\_X1 + K \* (y - mu\_Y1)

sigma\_X1\_given\_Y1 = sigma\_X1 - K \* sigma\_X1

# Smoothing Step

mu\_X0\_given\_Y1 = 2 \* (mu\_X1\_given\_Y1 - 1)

sigma\_X0\_given\_Y1 = 4 \* sigma\_X1\_given\_Y1

return {

'X1': (mu\_X1, sigma\_X1),

'X1\_given\_Y1': (mu\_X1\_given\_Y1, sigma\_X1\_given\_Y1),

'X0\_given\_Y1': (mu\_X0\_given\_Y1, sigma\_X0\_given\_Y1)

}

# Example usage

results = kalman\_filter(0.5, 1, -1, 2)

print("X1:", results['X1'])

print("X1 given Y1:", results['X1\_given\_Y1'])

print("X0 given Y1:", results['X0\_given\_Y1'])

```

**Problem 2**

**(a) Prediction Step**

```python

import numpy as np

import matplotlib.pyplot as plt

# Parameters

N = 1000 # Number of particles

# Draw samples

X0\_samples = np.random.normal(-1, np.sqrt(2), N)

Xi0\_samples = np.random.normal(0, 1, N)

# Compute X1 for each sample

X1\_samples = 0.5 \* X0\_samples + 1 + Xi0\_samples

# Plot histogram of particles

plt.hist(X1\_samples, bins=30, density=True, alpha=0.5, label='Histogram of particles')

plt.title('Histogram of Particles for X1')

plt.xlabel('X1')

plt.ylabel('Frequency')

plt.legend()

plt.show()

```

**(b)** The theoretical PDF for \(X\_1\) obtained from Problem 1 is:

\[ X\_1 \sim N\left(\frac{1}{2}, \frac{3}{2}\right) \]

We can overlay the theoretical PDF on the histogram:

```python

from scipy.stats import norm

# Plot histogram of particles

plt.hist(X1\_samples, bins=30, density=True, alpha=0.5, label='Histogram of particles')

# Plot theoretical PDF

x = np.linspace(min(X1\_samples), max(X1\_samples), 100)

plt.plot(x, norm.pdf(x, 0.5, np.sqrt(1.5)), 'r-', label='Theoretical PDF')

plt.title('Histogram of Particles for X1')

plt.xlabel('X1')

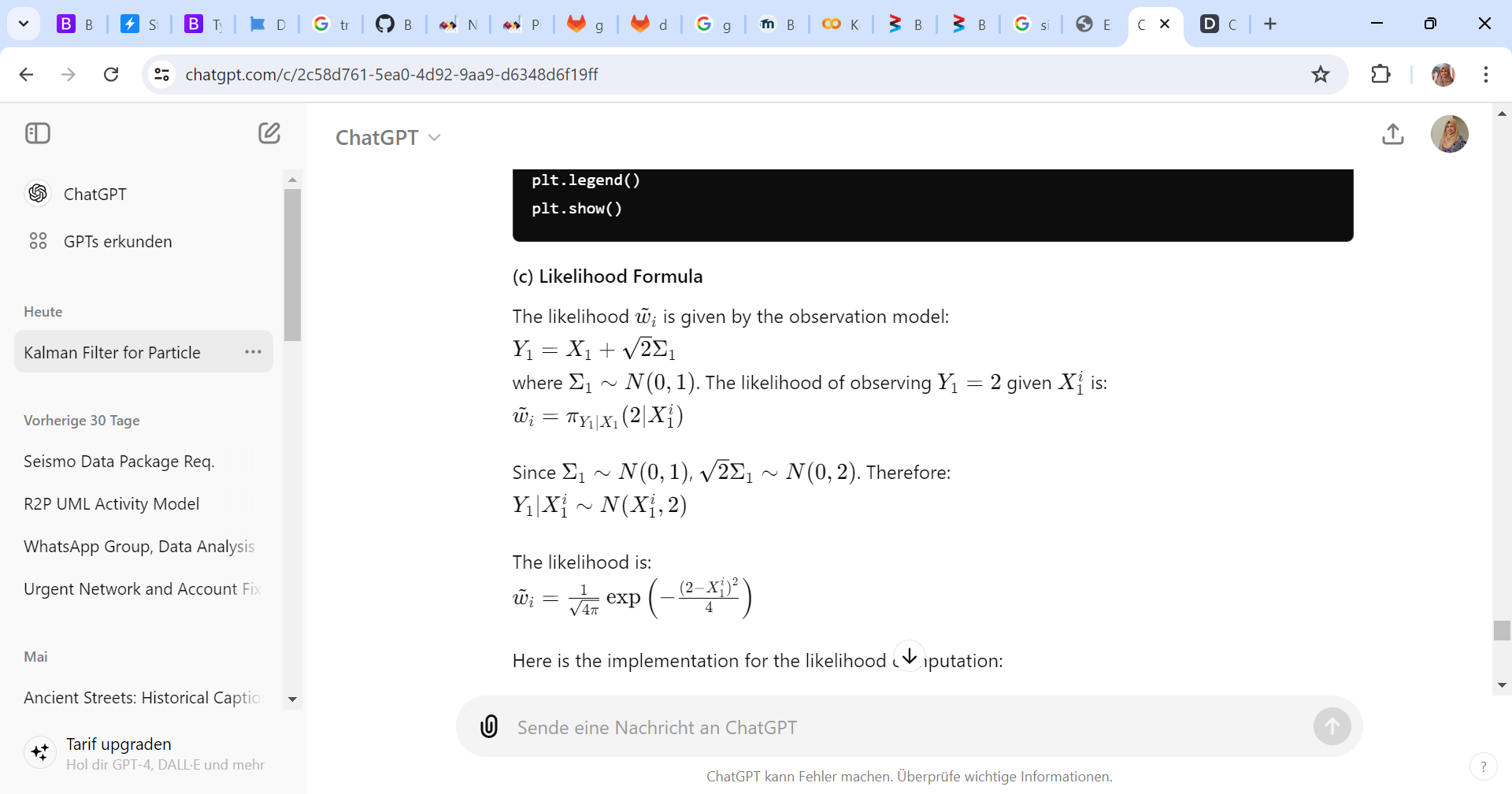
plt.ylabel('Density')

plt.legend()

plt.show()

```

**(c) Likelihood Formula**



Here is the implementation for the likelihood computation:

```python

# Observation

Y1 = 2

# Compute likelihood

likelihoods = (1 / np.sqrt(4 \* np.pi)) \* np.exp(-((Y1 - X1\_samples)\*\*2) / 4)

```

**(d) Compute normalized weights** \(w\_i\):

\[ w\_i = \frac{\tilde{w}\_i}{\sum\_{i=1}^N \tilde{w}\_i} \]

```python

# Compute normalized weights

weights = likelihoods / np.sum(likelihoods)

```

**(e) Compute the weighted mean and variance**

\[ \bar{m}\_1 = \sum\_{i=1}^N w\_i X\_1^i \]

\[ \bar{V}\_1 = \sum\_{i=1}^N w\_i (X\_1^i - \bar{m}\_1)^2 \]

```python

# Compute weighted mean

m\_bar = np.sum(weights \* X1\_samples)

# Compute weighted variance

V\_bar = np.sum(weights \* (X1\_samples - m\_bar)\*\*2)

print(f"Weighted Mean: {m\_bar}")

print(f"Weighted Variance: {V\_bar}")

```

**(f) Effective Sample Size**

Compute the Effective Sample Size (ESS):

\[ ESS = \frac{1}{\sum\_{i=1}^N w\_i^2} \]

```python

# Compute ESS

ESS = 1 / np.sum(weights\*\*2)

print(f"Effective Sample Size: {ESS}")

```