Exercise 11 – Bayesian inference and Data assimilation

Group : Duckies

1.1

Given that $X_0 \sim N(-1, 2)$, we can use the forward map to find the distribution of X_1.

Substituting $n = 0$ into the forward map equation, we have:

$$X_1 = \frac{1}{2}X_0 + 1 + \Xi_0$$

Since $\Xi_0 \sim N(0, 1)$, we can use the properties of linear combinations of normal random variables to find the distribution of X_1.

The mean of X_1 is given by:

$$E[X_1] = E\left[\frac{1}{2}X_0 + 1 + \Xi_0\right] = \frac{1}{2}E[X_0] + 1 + E[\Xi_0] = \frac{1}{2}(-1) + 1 + 0 = \frac{1}{2}$$

The variance of X_1 is given by:

$$Var[X_1] = Var\left[\frac{1}{2}X_0 + 1 + \Xi_0\right] = \left(\frac{1}{2}\right)^2 Var[X_0] + Var[\Xi_0] = \frac{1}{4}(2) + 1 = \frac{3}{2}$$

Therefore, the distribution of X_1 is $N\left(\frac{1}{2}, \frac{3}{2}\right)$.

1.2

Here, Forward mean $m_f = \frac{1}{2}$ Forward variance $V_f = \frac{3}{2}$

We have to find kalman gain, $K = \frac{Var[Z_1]}{Var[Z_1] + Var[\sqrt{2}\Sigma_1]}$

$\sqrt{2}\Sigma_1$ is observation noise

$$\Sigma_n \; N(0, 1)$$

$$\sqrt{2}\Sigma_n \; N(0, 2)$$

Now,

$$K = \frac{\frac{3}{2}}{\frac{3}{2} + 2} = \frac{3}{7}$$

Now, We need to update our analysis mean and variance.
$m_a = m_f - \frac{3}{7}(m_f - y_{obs})$
$= \frac{1}{2} - \frac{3}{7}(\frac{1}{2} - 2)$
$= \frac{8}{7}$
$V_a = V_f(1 - K)$
$= \frac{3}{2}(1 - \frac{3}{7})$
$= \frac{6}{7}$

Therefore, the distribution of X_1 conditioned on $Y_1 = 2$ is $N(\frac{8}{7}, \frac{6}{7})$

1.3

To find the distribution of X_0 conditioned on $Y_1 = 2$, we can use the fact that X_0 and Y_1 are jointly Gaussian. This means that their conditional distribution is also Gaussian, and we can use the formulas for the conditional mean and variance of jointly Gaussian random variables to find the distribution of X_0 given $Y_1 = 2$.

The conditional mean of X_0 given $Y_1 = 2$ is given by:

$$E[X_0|Y_1 = 2] = E[X_0] + \frac{Cov(X_0, Y_1)}{Var(Y_1)}(2 - E[Y_1])$$

The conditional variance of X_0 given $Y_1 = 2$ is given by:

$$Var[X_0|Y_1 = 2] = Var[X_0] - \frac{Cov(X_0, Y_1)^2}{Var(Y_1)}$$

To find these values, we need to compute the mean and variance of X_0, the mean and variance of Y_1, and the covariance between X_0 and Y_1.

We are given that $X_0 \sim N(-1, 2)$, so we have:

$$E[X_0] = -1$$

and

$$Var[X_0] = 2$$

From my previous response, we found that the distribution of Y_1 is $N\left(\frac{8}{7}, \frac{6}{7}\right)$, so we have:

$$E[Y_1] = \frac{8}{7}$$

and

$$Var[Y_1] = \frac{6}{7}$$

To find the covariance between X_0 and Y_1, we can use the fact that:

$$Cov(X_0, Y_1) = Cov(X_0, X_1 + \sqrt{2}\Sigma_1) = Cov(X_0, X_1) + Cov(X_0, \sqrt{2}\Sigma_1)$$

Since Σ_1 is independent of both X_0 and X_1, we have:

$$Cov(X_0, \sqrt{2}\Sigma_1) = 0$$

To find $Cov(X_0, X_1)$, we can use the forward map equation:

$$X_{n+1} = \frac{1}{2}X_n + 1 + \Xi_n$$

Substituting in $n = 0$, we have:

$$X_1 = \frac{1}{2}X_0 + 1 + \Xi_0$$

Taking the covariance of both sides with respect to X_0, we have:

$$Cov(X_0, X_1) = Cov\left(X_0, \frac{1}{2}X_0 + 1 + \Xi_0\right) = \frac{1}{2}Cov(X_0, X_0) + Cov(X_0, 1) + Cov(X_0, \Xi_0)$$

Since Ξ_0 is independent of X_0, we have:

$$Cov(X_0, \Xi_0) = 0$$

Also, since 1 is a constant, we have:

$$Cov(X_0, 1) = 0$$

Substituting these values into the above equation, we have:

$$Cov(X_0, X_1) = \frac{1}{2}Cov(X_0, X_0) = \frac{1}{2}Var(X_0) = 1$$

Substituting all of these values into the equations for the conditional mean and variance of X_0 given $Y_1 = 2$, we have:

$$E[X_0|Y_1 = 2] = E[X_0] + \frac{Cov(X_0, Y_1)}{Var(Y_1)}(2 - E[Y_1]) = -1 + \frac{1}{\frac{6}{7}}\left(2 - \frac{8}{7}\right) = -\frac{5}{6}$$

and

$$Var[X_0|Y_1 = 2] = Var[X_0] - \frac{Cov(X_0, Y_1)^2}{Var(Y_1)} = 2 - \frac{1^2}{\frac{6}{7}} = \frac{8}{3}$$

Therefore, the distribution of X_0 conditioned on $Y_1 = 2$ is $N\left(-\frac{5}{6}, \frac{8}{3}\right)$.

1.4

Here is a pseudo code that takes as input the parameters α, β, m, and y and outputs the distribution of X_1, the distribution of X_1 conditioned on $Y_1 = y$, and the distribution of X_0 conditioned on $Y_1 = y$:

```
function get_distributions(α, β, m, y):
    # (i) Distribution of X1
    mean_X1 = α * m
    var_X1 = α^2 + β^2

    # (ii) Distribution of X1 conditioned on Y1 = y
    mean_X1_Y1 = mean_X1 + (β^2 / (β^2 + 2)) * (y - mean_X1)
    var_X1_Y1 = var_X1 - (β^4 / (β^2 + 2))

    # (iii) Distribution of X0 conditioned on Y1 = y
    mean_X0_Y1 = m + ((α * β^2) / (α^2 + β^2 + 2)) * (y - mean_X1)
    var_X0_Y1 = 1 - ((α^2 * β^2) / (α^2 + β^2 + 2))

    return {"X1": [mean_X1, var_X1], "X1|Y1": [mean_X1_Y1,
var_X1_Y1], "X0|Y1": [mean_X0_Y1, var_X0_Y1]}
```

This function takes as input the values of α, β, m, and y and calculates the means and variances of the three distributions using the formulas derived above and returns them in a dictionary. The keys of the dictionary are "X1", "X1|Y1", and "X0|Y1", and the values are lists containing the mean and variance of the corresponding distribution.

2.1

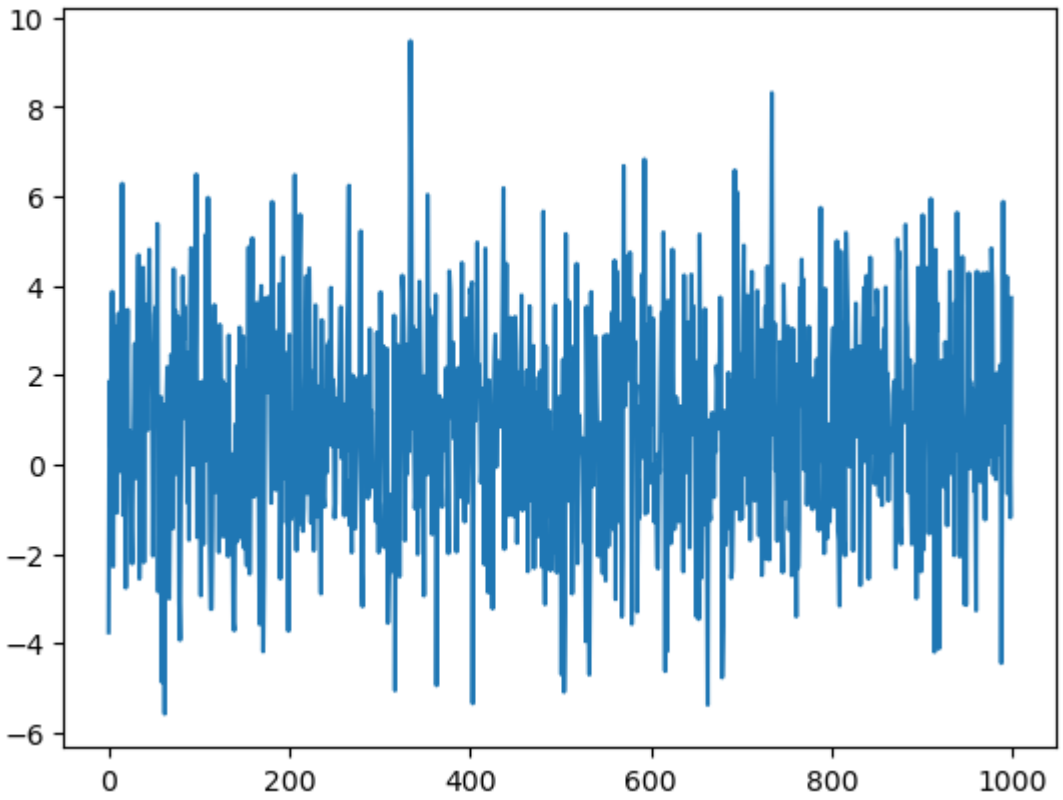
In [1]:

```python
import numpy as np

def particle_filter_prediction(alpha, beta, m, N):
    # Initialize the array of particles
    X0_particles = np.zeros(N)
    Xi0_particles = np.zeros(N)
    X1_particles = np.zeros(N)

    # Draw N independent samples of X0 and Xi0
```

```
    X0_particles = np.random.normal(m, 1, N)
    Xi0_particles = np.random.normal(0, 1, N)

    # Compute X1 for each sample path
    X1_particles = alpha * X0_particles + beta * Xi0_particles

    return X1_particles
```

In [16]:
```
plt.plot(particle_filter_prediction(-1,2,-5/6,1000))
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x17289d07190>]
```



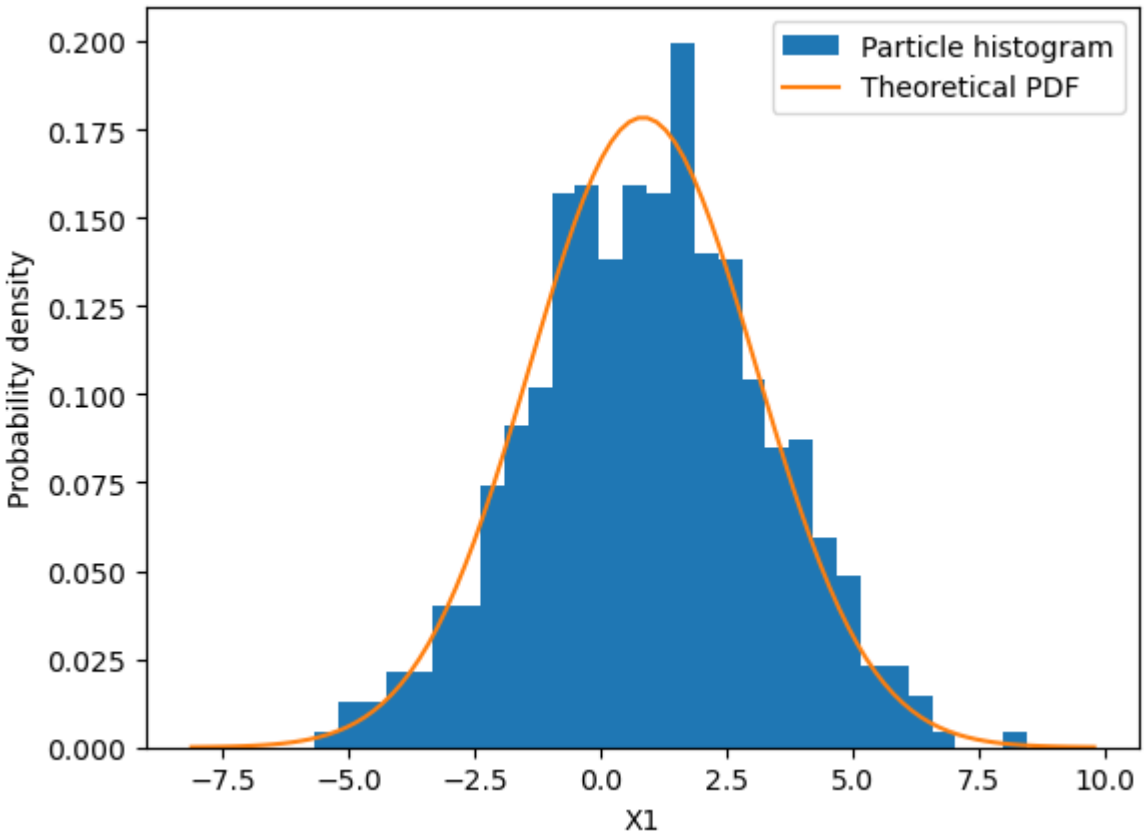
2.2

In [2]:
```
import matplotlib.pyplot as plt
from scipy.stats import norm

def plot_particle_histogram(alpha, beta, m, N):
    # Compute the predicted values of X1 for each sample path
    X1_particles = particle_filter_prediction(alpha, beta, m, N)

    # Plot the histogram of particles
```

```
    plt.hist(X1_particles, bins=30, density=True, label='Particle
histogram')

    # Compute the theoretical mean and variance of X1
    X1_mean = alpha * m
    X1_var = alpha**2 + beta**2

    # Plot the theoretical PDF of X1
    x = np.linspace(X1_mean - 4 * np.sqrt(X1_var), X1_mean + 4 *
np.sqrt(X1_var), 100)
    y = norm.pdf(x, X1_mean, np.sqrt(X1_var))
    plt.plot(x, y, label='Theoretical PDF')

    # Add labels and legend
    plt.xlabel('X1')
    plt.ylabel('Probability density')
    plt.legend()

    # Show the plot
    plt.show()
```

In [17]:

```
# Set the parameter values
N = 1000

# Call the function to generate the plot
plot_particle_histogram(-1, 2, -5/6, N)
```



2.3

Since $Y_1 = X_1 + \sqrt{2}\Sigma_1$, where Σ_1 is normally distributed with mean 0 and variance 1, the conditional distribution of Y_1 given $X_1 = x$ is normal with mean x and variance 2.

Therefore, the likelihood of observing $Y_1 = y$ given $X_1 = x$ is given by the probability density function of a normal distribution with mean x and variance 2 evaluated at y:

$$\pi_{(Y_1|X_1)}[y|x] = (\frac{1}{\sqrt{(4\pi)}}) \exp(\frac{-(y-x)^2}{4})$$

Substituting X_1^i for x, we get the formula for the likelihood

$$w_i := \pi_{(Y_1|X_1)}[y|X_1^i]$$

$$= (\frac{1}{\sqrt{(4\pi)}}) \exp(\frac{-(y-X_i1)^2}{4})$$

The formula for the likelihood represents the probability of observing the data $Y_1 = y$ given the value of $X_1 = X_1^i$. It is derived from the conditional distribution of Y_1 given X_1, which is normal with mean X_1 and variance 2.

This formula tells us how likely it is to observe the data $Y_1 = y$ for a given value of X_1. The likelihood is higher for values of X_1 that are closer to y, since the normal distribution has its highest density near its mean. As X_1 moves further away from y, the likelihood decreases, reflecting the fact that it becomes less likely to observe $Y_1 = y$ as X_1 moves away from it.

This pseudo code calculates the likelihood w_i for each sample X_i1 in the input array `X1_samples` using the formula derived above, with y = 2. The common constant factor is omitted since it will be normalized later.

```
function get_likelihoods(X1_samples, y):
    w = exp(-(y - X1_samples)^2 / 4)
    return w
```

We can use this function to compute the likelihoods for a given set of X1 samples and the observation Y1 = 2. If we have an array `X1_samples` containing `N` samples of X1, we can call `get_likelihoods(X1_samples, 2)` to get an array of `N` likelihood values, one for each sample.

2.4

This function calculates the normalized weight w_i for each sample by dividing the likelihood w_i by the sum of all likelihoods. This ensures that the normalized weights sum to 1.

```
function get_normalized_weights(w):
    w_sum = sum(w)
    w_norm = w / w_sum
    return w_norm
```

We can use this function to compute the normalized weights for a given set of likelihood values. If we have an array `w` containing `N` likelihood values, we can call `get_normalized_weights(w)` to get an array of `N` normalized weights, one for each sample.

2.5

This function calculates the weighted mean and variance of the X1 samples using the formulas provided. The weighted mean is calculated as the sum of the product of each sample and its corresponding weight, and the weighted variance is calculated as the sum of the product of each sample's squared deviation from the mean and its corresponding weight.

```
function get_weighted_mean_variance(X1_samples, w):
    mean = sum(w * X1_samples)
    variance = sum(w * (X1_samples - mean)^2)
    return [mean, variance]
```

We can use this function to compute the weighted mean and variance for a given set of X1 samples and normalized weights. If we have an array `X1_samples` containing `N` samples of X1 and an array `w` containing `N` normalized weights, we can call `get_weighted_mean_variance(X1_samples, w)` to get the weighted mean and variance of the samples.

2.6

This function calculates the effective sample size (ESS) using the formula provided. The ESS is calculated as the reciprocal of the sum of the squared normalized weights.

```
function get_ESS(w):
    ESS = 1 / sum(w^2)
    return ESS
```

We can use this function to compute the ESS for a given set of normalized weights. If we have an array `w` containing `N` normalized weights, we can call `get_ESS(w)` to get the ESS.