To simulate bank and swap rates using the Hull-White model with the given data, you can follow these steps:

1. Preprocess the data: First, you need to preprocess the historical bank rates and swap rates data to ensure they have consistent time intervals. Since bank rates have a monthly frequency and swap rates are published daily, you should align them to a common time grid, such as daily or a finer granularity.

2. Calibrate the Hull-White model: Next, you'll need to calibrate the Hull-White model using the historical data to estimate the model parameters `kappa`, `theta`, and `sigma` for each tenor. This calibration process involves finding the best-fit parameters that minimize the difference between the model's simulated rates and the historical rates.

3. Simulate bank rates: Once you have calibrated the model, you can simulate the paths of bank rates using the Hull-White model with the time-dependent `theta(t)` and the estimated parameters.

4. Simulate swap rates: To simulate swap rates, you can use the simulated bank rates as inputs to a suitable interpolation method. For each swap tenor (3M, 6M, 9M, 1Y, 15M, 18M, 2Y, 3Y, 4Y, 5Y, 7Y, 10Y, 15Y, 20Y), you'll need to interpolate the corresponding bank rate from the simulated bank rate paths. One common interpolation method used for this purpose is linear interpolation.

Here's a high-level outline of how to proceed with these steps:

```python

# Step 1: Preprocess the data (align to a common time grid)

# You can interpolate the bank rates to daily frequency or the swap rates to monthly frequency.

# Step 2: Calibrate the Hull-White model

# For each tenor, calibrate the Hull-White model to estimate kappa, theta, and sigma using historical data.

# Step 3: Simulate bank rates

# Using the Hull-White model with time-dependent theta(t) and the calibrated parameters,

# simulate multiple paths of bank rates for each tenor.

# Step 4: Simulate swap rates

# For each swap tenor, use the simulated bank rates to interpolate the corresponding bank rate.

# Then, apply the interpolation result as the simulated swap rate.

# You can repeat the simulation process for multiple paths to account for uncertainty.

# Finally, you can plot the simulated bank and swap rates over time for visualization.

```

Keep in mind that the accuracy of the simulations heavily relies on the quality of historical data and the calibration process. Calibration is an essential step to ensure that the model reasonably replicates the historical rate movements.

Additionally, as mentioned earlier, the Hull-White model is a simplified model, and there are more sophisticated approaches like the LIBOR Market Model (LMM) or the SABR model that can be used for modeling swap rates. However, the Hull-White model should give you a good starting point for understanding the dynamics of interest rates and their impact on both bank and swap rates.

import numpy as np

import matplotlib.pyplot as plt

# Example data: Bank rates (1, 5, 10, 20 years) and Swap rates (3M, 6M, 1Y, ..., 20Y)

bank\_rate\_tenors = [1, 5, 10, 20]

swap\_rate\_tenors = [0.25, 0.5, 1, 1.25, 1.5, 1.75, 2, 3, 4, 5, 7, 10, 15, 20]

historical\_bank\_rates = np.array([[0.02, 0.025, 0.03, 0.035], # Monthly bank rates for 1-year tenor

[0.03, 0.032, 0.035, 0.038], # Monthly bank rates for 5-year tenor

[0.04, 0.042, 0.045, 0.048], # Monthly bank rates for 10-year tenor

[0.05, 0.052, 0.055, 0.058]]) # Monthly bank rates for 20-year tenor

historical\_swap\_rates = np.array([[0.01, 0.015, 0.018, 0.02, 0.025, 0.028, 0.03, 0.035, 0.04, 0.045, 0.05, 0.055, 0.06, 0.065]]) # Daily swap rates

# Hull-White model parameters

kappa = 0.1

sigma = 0.015

num\_paths = 1000

num\_steps = 252 # Assuming daily frequency

# Simulate bank rates using Hull-White model

def simulate\_bank\_rates(tenor\_idx):

theta = historical\_bank\_rates[tenor\_idx]

bank\_rates = np.zeros((num\_steps + 1, num\_paths))

bank\_rates[0, :] = theta[0] # Initial rate

dt = 1 / num\_steps

for i in range(1, num\_steps + 1):

dW = np.random.normal(0, np.sqrt(dt), (num\_paths,))

for j in range(num\_paths):

drift = kappa \* (theta[i - 1] - bank\_rates[i - 1, j]) \* dt

volatility = sigma \* np.sqrt(dt) \* dW[j]

bank\_rates[i, j] = bank\_rates[i - 1, j] + drift + volatility

return bank\_rates

# Simulate swap rates using interpolated bank rates

def simulate\_swap\_rates(bank\_rates, tenor\_idx):

simulated\_swap\_rates = np.zeros\_like(historical\_swap\_rates)

for i in range(len(historical\_swap\_rates)):

interpolated\_bank\_rate = np.interp(swap\_rate\_tenors[i], bank\_rate\_tenors, bank\_rates[tenor\_idx])

simulated\_swap\_rates[i] = interpolated\_bank\_rate

return simulated\_swap\_rates

# Main simulation loop

for tenor\_idx, tenor\_years in enumerate(bank\_rate\_tenors):

bank\_rates = simulate\_bank\_rates(tenor\_idx)

swap\_rates = simulate\_swap\_rates(bank\_rates, tenor\_idx)

# Plotting

plt.figure(figsize=(10, 6))

plt.plot(swap\_rate\_tenors, historical\_swap\_rates[0], 'o-', label='Historical Swap Rates')

plt.plot(swap\_rate\_tenors, swap\_rates[0], 'x-', label='Simulated Swap Rates')

plt.xlabel('Maturity (Years)')

plt.ylabel('Rate')

plt.title(f'Swap Rate Simulation for {tenor\_years}-Year Bank Rate')

plt.legend()

plt.grid(True)

plt.show()

import numpy as np

def hull\_white\_model(r0, kappa, theta, sigma, T, num\_steps, num\_paths, correlation\_matrix):

dt = T / num\_steps

num\_tenors = correlation\_matrix.shape[0]

# Initialize zero curve rates for all tenors

rates = np.zeros((num\_steps + 1, num\_tenors, num\_paths))

rates[0, :, :] = r0.reshape(num\_tenors, 1)

for i in range(1, num\_steps + 1):

# Generate correlated random increments for all tenors and paths

dW = np.random.normal(0, np.sqrt(dt), (num\_tenors, num\_paths))

dW\_corr = np.linalg.cholesky(correlation\_matrix) @ dW

# Update rates for all tenors and paths using the Hull-White model dynamics

for j in range(num\_tenors):

rates[i, j, :] = (

rates[i - 1, j, :]

+ kappa \* (theta[j] - rates[i - 1, j, :]) \* dt

+ sigma[j] \* np.sqrt(dt) \* dW\_corr[j, :]

)

return rates

# Sample correlation matrix for three tenors (replace with your desired correlation matrix)

correlation\_matrix = np.array([[1.0, 0.5, 0.3],

[0.5, 1.0, 0.6],

[0.3, 0.6, 1.0]])

# Sample parameters for three tenors (replace with your desired tenor-specific parameters)

r0 = np.array([0.03, 0.035, 0.04])

kappa = np.array([0.1, 0.15, 0.12])

theta = np.array([0.028, 0.033, 0.038])

sigma = np.array([0.01, 0.02, 0.015])

T = 1.0

num\_steps = 100

num\_paths = 5

# Simulate interest rate paths with correlations between tenors

rates = hull\_white\_model(r0, kappa, theta, sigma, T, num\_steps, num\_paths, correlation\_matrix)

# Print the simulated rates for each tenor

print("Simulated Short-Term Interest Rate Paths:")

print(rates)