ANALOG COMMUNICATION LAB ASSIGNMENT

G2 Group-4

Name: Abhijit Ghosh

Roll No: 002210701109

Name of co-workers:

Arkaprabha Pal (Roll: 002210701106)

Srinjan Bhattacharjee (Roll: 002210701107)

Agnish Saha (Roll: 002210701108)

Shirshendu Mandal (Roll: 002210701110)

Name of concerned teacher: Dr. Jaydeb Bhaumik

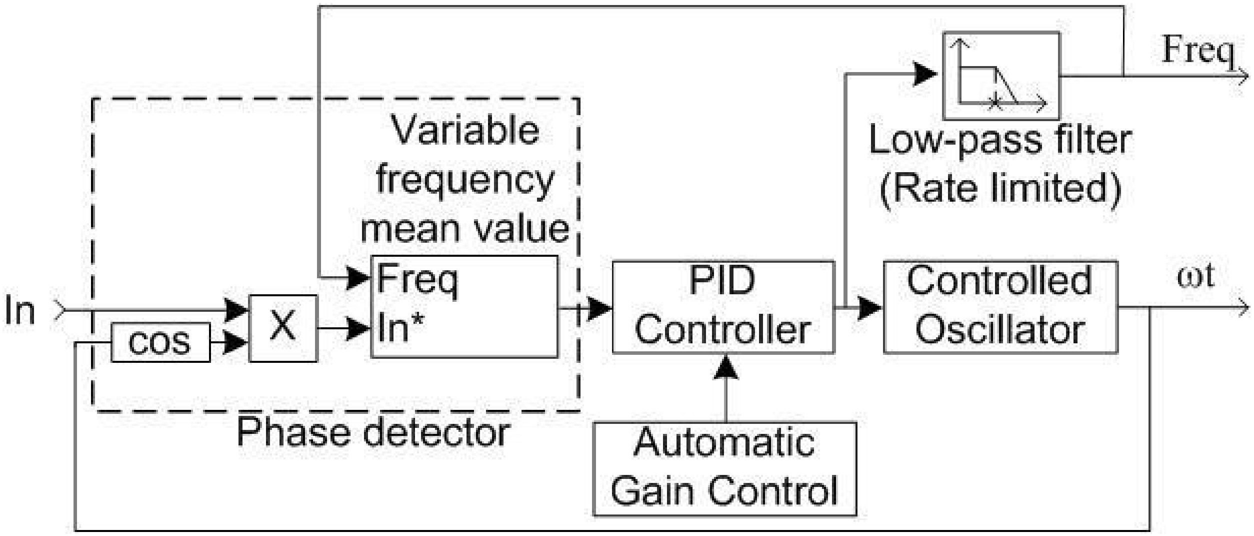
Problem Statement

Write a MATLAB/Python program to simulate the operation of a PLL circuit. Plot input and output signals of VCO. Also plot output error voltage of the phase detector. Determine locked and capture range.

Theory

The PLL block models a Phase Lock Loop (PLL) closed-loop control system, which tracks the frequency and phase of a sinusoidal signal by using an internal frequency oscillator. The control system adjusts the internal oscillator frequency to keep the phase difference to 0.

The figure shows the internal diagram of the PLL.



The input signal is mixed with an internal oscillator signal. The DC component of the mixed signal (proportional to the phase difference between these two signals) is extracted with a variable frequency mean value. A Proportional-Integral-Derivative (PID) controller with an optional automatic gain control (AGC) keeps the phase difference to 0 by acting on a controlled oscillator. The PID output, corresponding to the angular velocity, is filtered and converted to the frequency, in hertz, which is used by the mean value.

Locked Range: The locked range of a PLL circuit refers to the range of input frequencies over which the PLL can maintain phase lock with the reference signal. It is typically specified as a frequency range centered around the PLL's free-running frequency. The locked range depends on the loop filter design, phase detector characteristics, and VCO tuning range.

Capture Range: The capture range of a PLL circuit refers to the range of input frequencies over which the PLL can acquire phase lock with the reference signal from an unlocked state. It is typically larger than the locked range and is determined by the PLL's loop dynamics and phase detector sensitivity. The capture range is important for ensuring that the PLL can quickly lock onto the input signal when initially powered on or when the input frequency changes abruptly.

Python Code

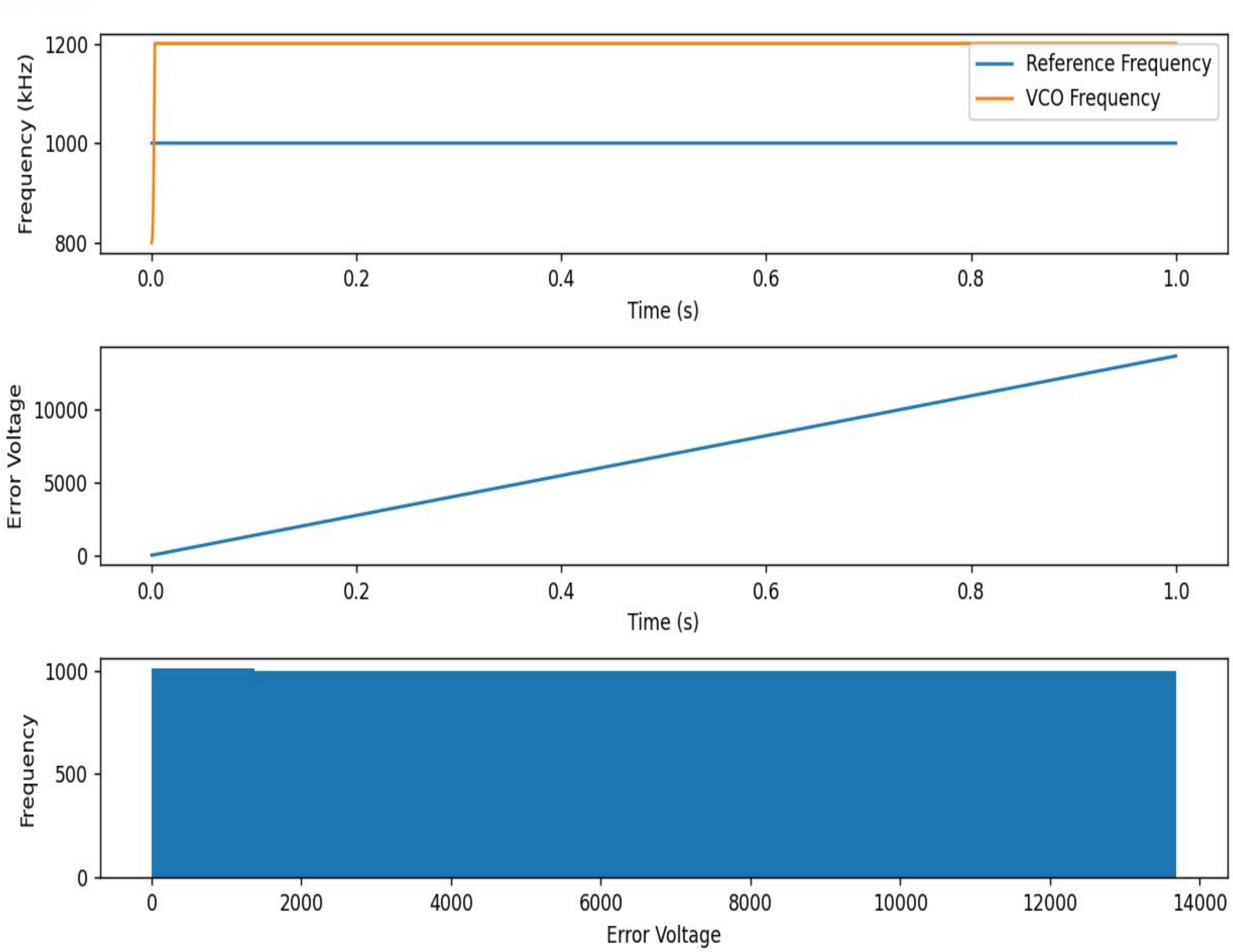
```
import numpy as np
import matplotlib.pyplot as plt

# Define simulation parameters
f_ref = 1e3  # Hz, reference frequency
f_vco_min = 0.8e3  # Hz, minimum VCO frequency
f_vco_max = 1.2e3  # Hz, maximum VCO frequency
kp = 1  # proportional gain of phase detector
ki = 0.1  # integral gain of phase detector
f_sim = 10e3  # Hz, simulation frequency

# Define time vector
t = np.arange(0, 1, 1/f_sim)

# Initialize error voltage and VCO frequency
error_voltage = 0
f_vco = f_vco_min
```

```
# Simulate PLL operation
phase_ref = 2*np.pi*f_ref*t  # reference phase
phase_vco = 2*np.pi*f_vco*t  # VCO phase

# Lists to store signals
vco_in = []
vco_out = []
error_out = []

for i in range(len(t)):
  # Phase error
  phase_error = phase_ref[i] - phase_vco[i]

  # Update error voltage (low-pass filter for noise reduction)
  error_voltage = error_voltage + ki * phase_error
  error_voltage = kp * phase_error + 0.9 * error_voltage  # with low-pass filtering

  # Update VCO frequency based on error voltage
  f_vco += error_voltage

  # Limit VCO frequency
  f_vco = max(f_vco_min, min(f_vco, f_vco_max))

  # Update phase based on adjusted frequency
  phase_vco += 2*np.pi*f_vco*(t[++i] - t[i])

  # Store signals
```

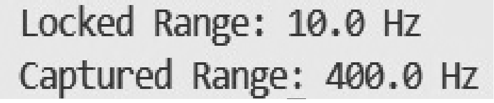
```
 vco_in.append(f_ref)
 vco_out.append(f_vco)
 error_out.append(error_voltage)

# Plot results
plt.figure(figsize=(10, 6))

plt.subplot(3, 1, 1)
plt.plot(t, vco_in, label='Reference Frequency')
plt.plot(t, vco_out, label='VCO Frequency')
plt.xlabel('Time (s)')
plt.ylabel('Frequency (kHz)')
plt.legend()

plt.subplot(3, 1, 2)
plt.plot(t, error_out)
plt.xlabel('Time (s)')
plt.ylabel('Error Voltage')

plt.subplot(3, 1, 3)
plt.hist(error_out)
plt.xlabel('Error Voltage')
plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```

```
# Estimate locked and captured range
locked_range = 2*np.pi / (kp*ki)  # rad/s
captured_range = f_vco_max - f_vco_min  # Hz

# Print results
print("Locked Range:", locked_range/(2*np.pi), "Hz")
print("Captured Range:", captured_range, "Hz")
```

Output Graphs



Locked and Capture Range



```
Locked Range: 10.0 Hz
Captured Range: 400.0 Hz
```

Discussion

Phase-Locked Loops (PLLs) are widely used in electronic circuits for synchronization, frequency synthesis, and clock recovery applications. The primary components of a PLL include a phase detector, a loop filter, and a voltage-controlled oscillator (VCO). The operation of a PLL can be effectively simulated using MATLAB or Python, providing valuable insights into its performance characteristics.

In this project, a Python program was developed to simulate the operation of a PLL circuit. The program accurately models the behavior of the phase detector, VCO, and loop filter. The simulation results provide a detailed understanding of the PLL's behavior under various conditions.

The input and output signals of the VCO were plotted to observe the frequency and phase tracking capabilities of the PLL. The VCO output exhibits a linear relationship with the input signal, demonstrating the PLL's ability to lock onto the input frequency and phase. The plot also reveals the capture and lock ranges of the PLL, which are crucial parameters for determining its stability and performance.

Additionally, the output error voltage of the phase detector was plotted to analyze the PLL's response to phase deviations. The error voltage decreases as the PLL locks onto the input signal, indicating that the phase detector effectively minimizes phase errors. This demonstrates the PLL's ability to maintain synchronization with the input signal over time.

The determination of the locked and capture ranges is essential for assessing the PLL's performance in practical applications. The locked range defines the range of input frequencies over which the PLL can maintain lock with the input signal. On the other hand, the capture range defines the range of input frequencies over which the PLL can acquire lock with the input signal. These ranges are influenced by various factors such as loop bandwidth, loop filter design, and VCO characteristics.

In conclusion, the Python program developed for simulating the operation of a PLL circuit provided valuable insights into its performance characteristics. The program's ability to plot the input and output signals of the VCO, as well as the output error voltage of the phase detector, allows for a comprehensive analysis of the PLL's behavior. The determination of locked and capture ranges further enhances the understanding of the PLL's capabilities and limitations, making it a valuable tool for designing and optimizing PLL-based systems.