import numpy as np import matplotlib.pyplot as plt import scipy as sp

def add_noise(s, SNR): var_s = np.cov(s) var_noise = var_s/(10**(SNR/10)) noise = var_noise**0.5 * np.random.randn(len(s)) return s + noise

def add_noise_db(s, noise_db): var_noise = 10**(noise_db/10) noise = var_noise**0.5 * np.random.randn(len(s)) return s + noise

def lowpass(data, f_cutoff, f_s): nyq = 0.5*f_s normal_cutoff = f_cutoff/nyq b, a = sp.signal.butter(2, normal_cutoff, btype="low", analog=False) y = sp.signal.lfilter(b, a, data) return y

def bandpass(data, f_pass, f_cutoff, f_s): nyq = 0.5*f_s

```
normal_pass = f_pass/nyq
normal_cutoff = f_cutoff/nyq

b, a = sp.signal.butter(2, (normal_pass, normal_cutoff), btype="bandpass", analog=False)
y = sp.signal.lfilter(b, a, data)
return y
```

def fm(m, f_c, beta_f, t): omega_c = 2*np.pi*f_c return np.cos(omega_c*t + beta_f*m)

def product_demod(s, f_baseband, f_c, f_s, t): omega_c = 2*np.pi*f_c mixed = s*np.cos(omega_c*t) return lowpass(mixed, f_baseband, f_s)

def fm_demod(s_fm, f_c, f_s, beta_f, t): diff_t = np.gradient(s_fm, t) envelope = np.abs(sp.signal.hilbert(diff_t))

```
omega_c = 2*np.pi*f_c
dt = 1/f_s
m_demod = np.empty_like(envelope)

err1 = dt*(envelope[0] + envelope[1] - 2*omega_c)/beta_f**2
err2 = dt*(dm_dt[0] + dm_dt[1])/beta_f

for i in range(len(envelope)):
    m_demod[i] = np.sum((envelope[:i+1] - omega_c)/(beta_f*f_s))

return m_demod
```

def main(): T_s = 0.001 f_s = 1/T_s f_m = 10 omega_m = 2*np.pi*f_m

```
f_c = 100

beta_f = 2

t = np.arange(0,1,T_s)
f = np.linspace(0,f_s,len(t))
```

```
m = 0.1*np.sin(omega_m*t)# + np.sin(omega_m/2*t)
s_fm = fm(m, f_c, beta_f, t)

m_fm_demod = fm_demod(s_fm, f_c, f_s, beta_f, t)

plt.subplot(211)
plt.plot(t, m, label="Original Message Signal")
plt.plot(t, m_fm_demod, label="Demodulated Message")
plt.legend()

plt.subplot(212)
plt.plot(t, m-m_fm_demod)

plt.show()
```

if **name** == "**main**": main()