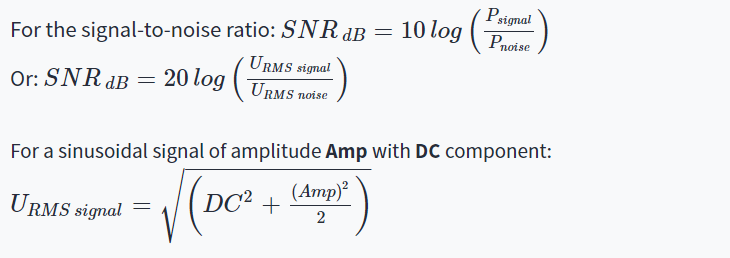
**For the octave online**

**Generate a P period of a sinusoidal voltage signal of frequency Fsig, the sampling frequency will be Fs, the amplitude of Amp with the same parameters is the DC component. Superimpose normal distribution noise with zero mean on this signal so that the signal-to-noise ratio is SNR dB. See below for specific values.**



**Generate a noise signal with normal distribution, zero mean, whose rms value (standard deviation) will be equal to the calculated value of URMS noise**

**(use the "randn" function in octave online).**

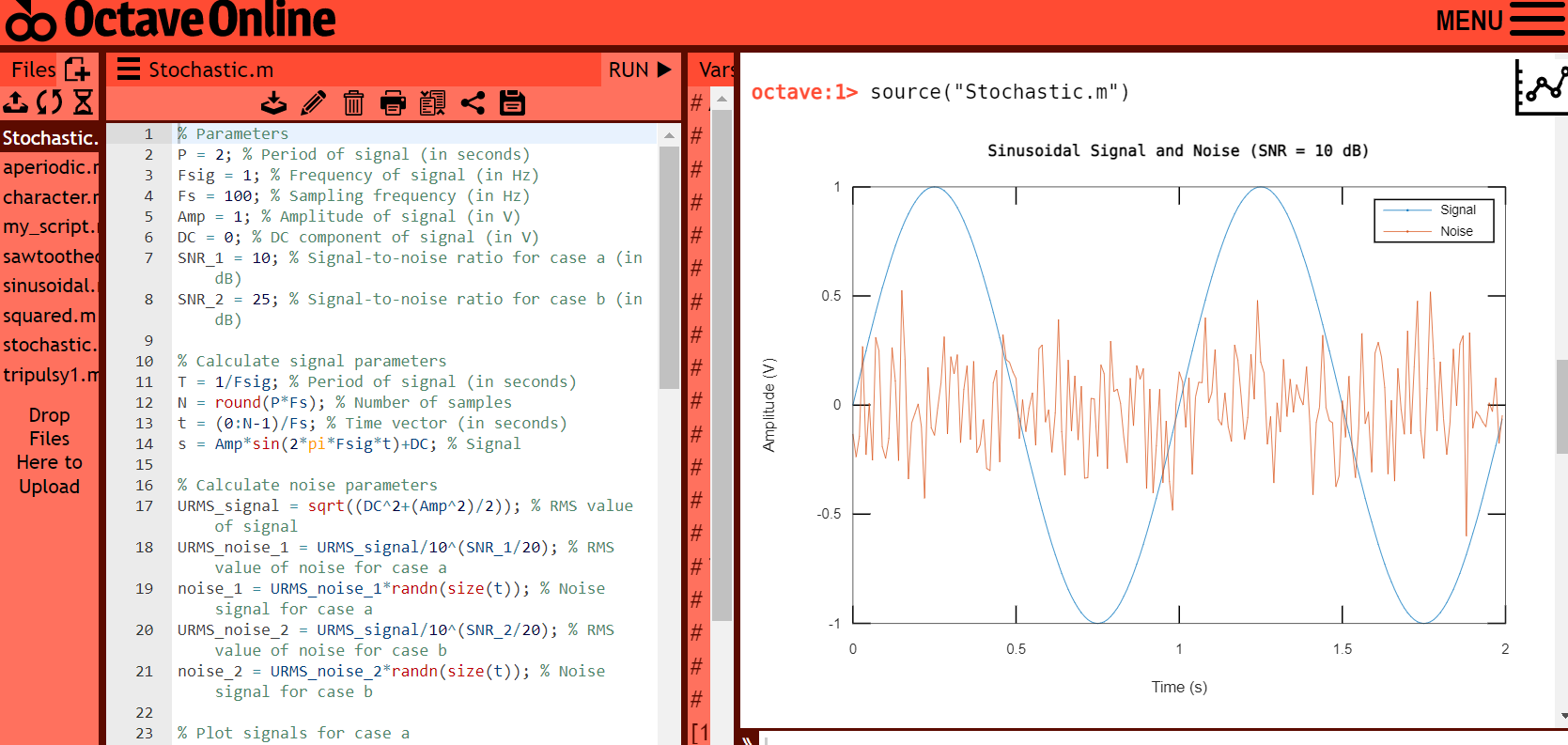
**P = 2, Fsig = 1 Hz, Fs = 100 Hz, Amp = 1 V, DC = 0 V,**

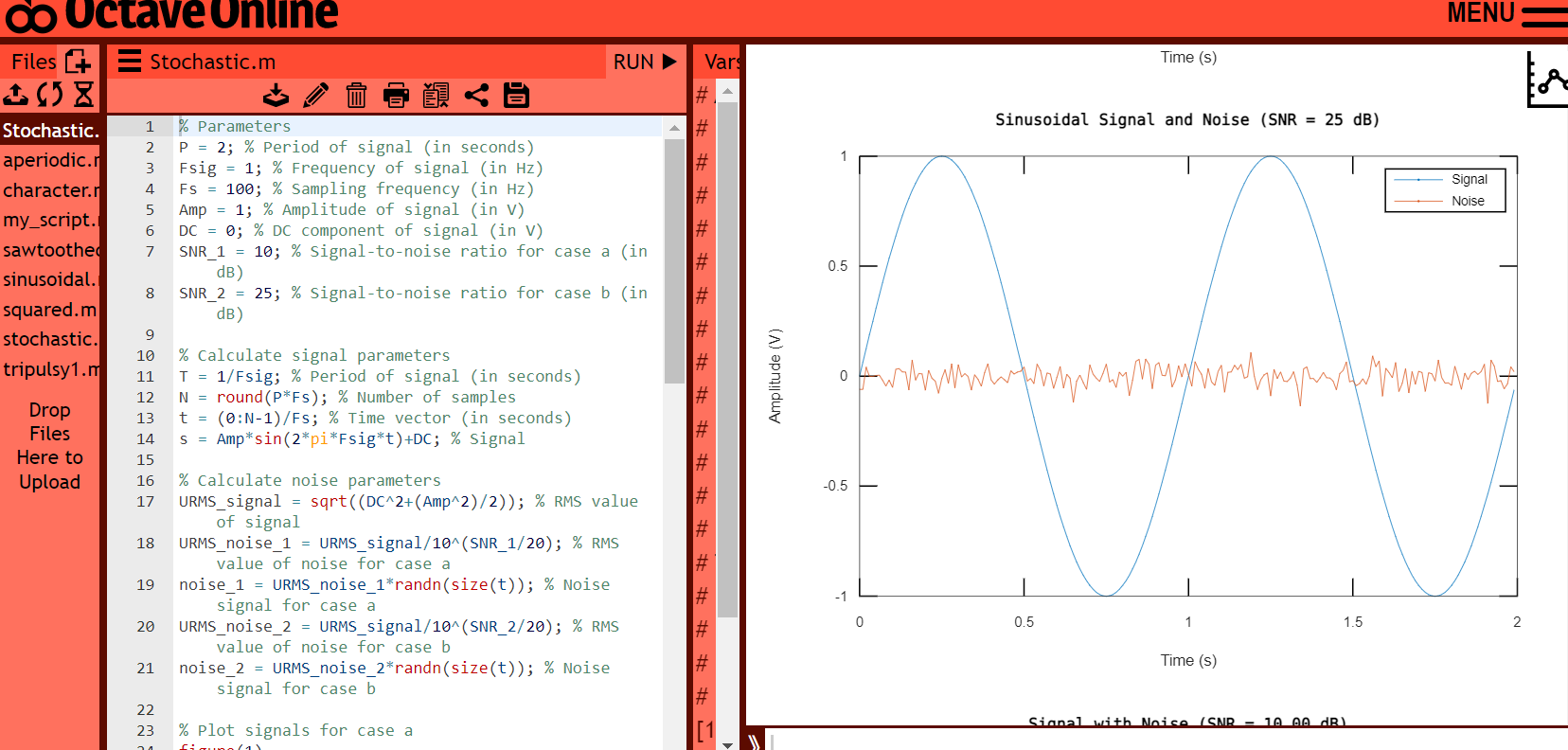
**a) SNR = 10 dB**

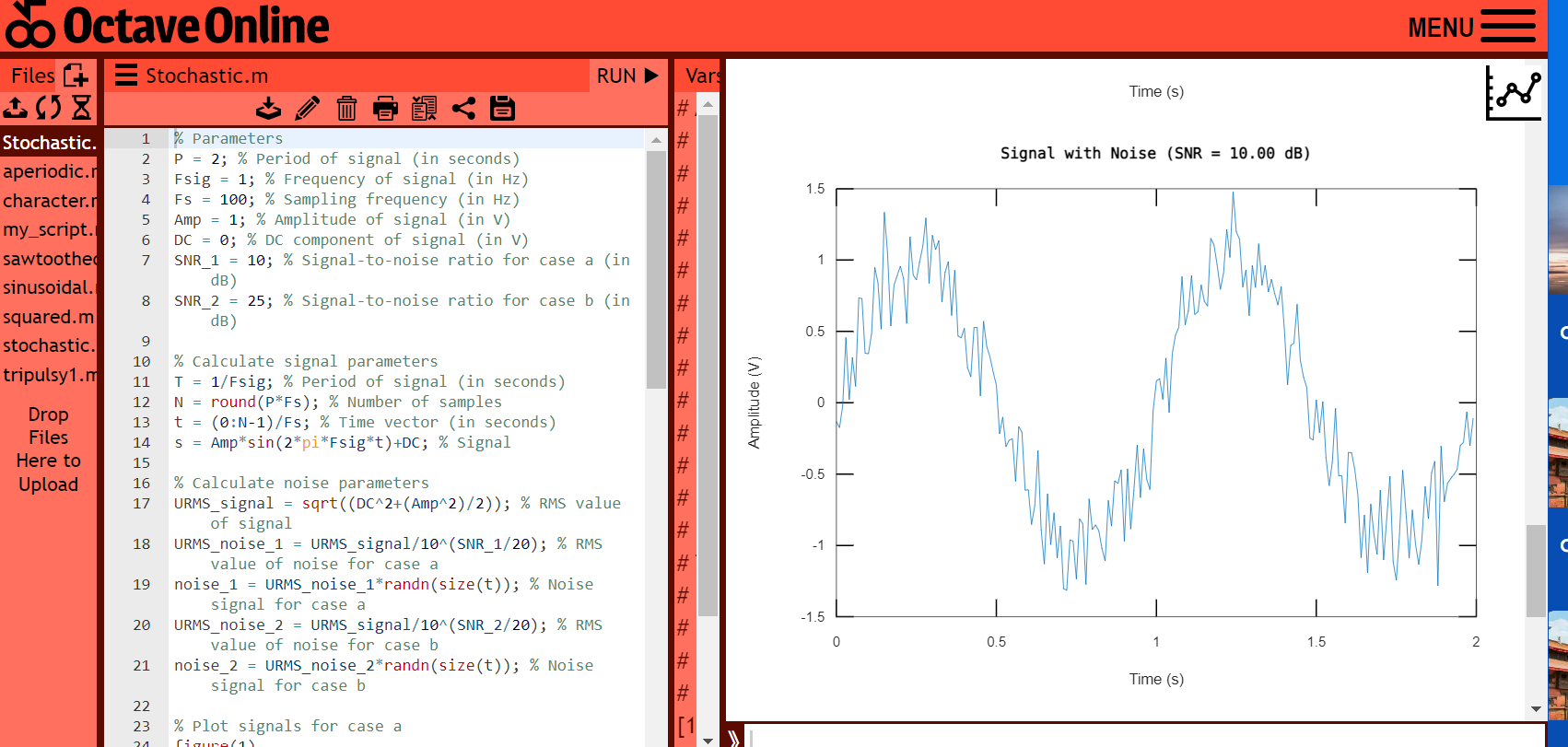
**b) SNR = 25 dB**

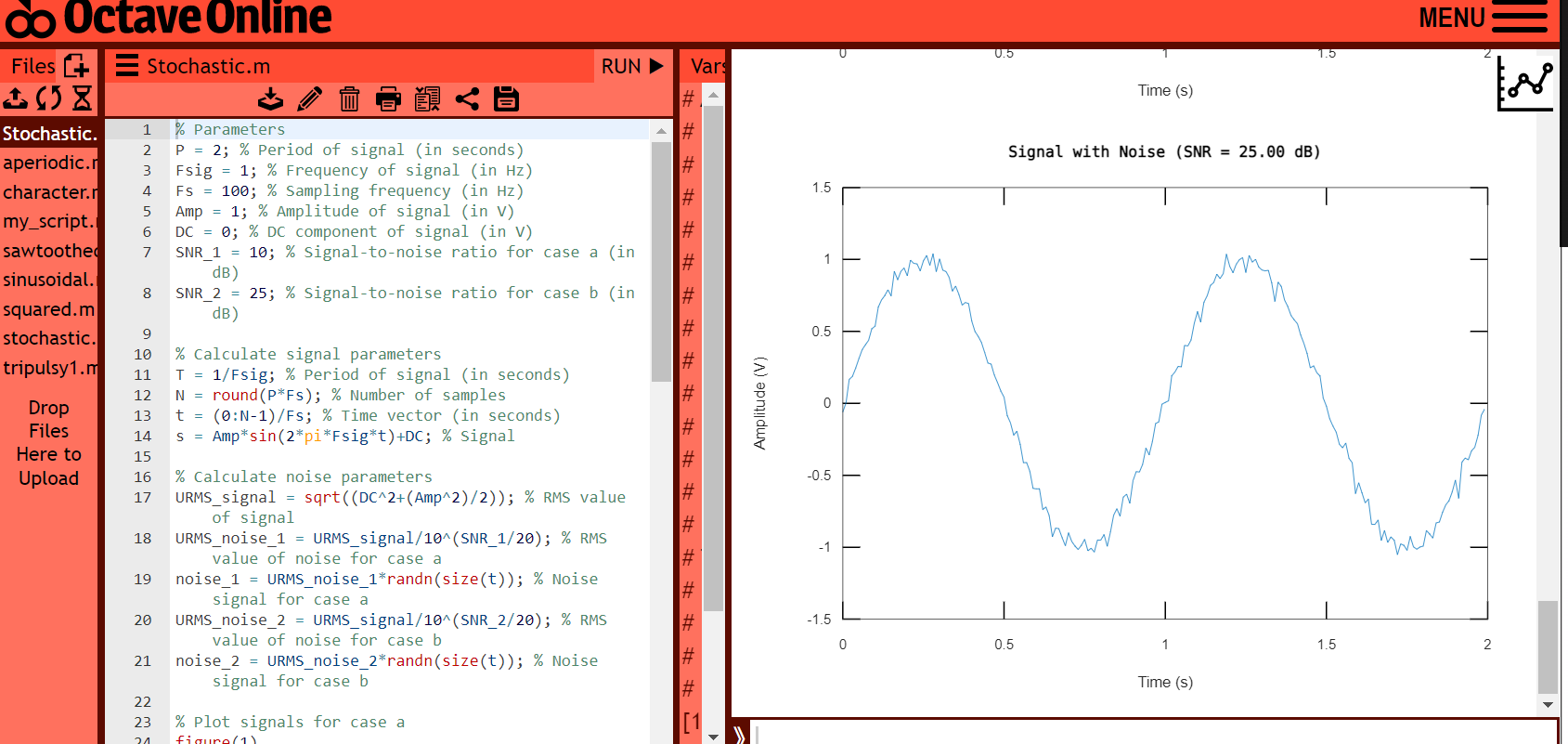
**For each case (a and b) create two graphs, in the first graph show the sine signal and the noise signal separately, in the second graph superimpose the noise on the sine signal and calculate the actual signal-to-noise ratio, see below.**

**Screenshots:**









**Code in Octave online:**

% Parameters

P = 2; % Period of signal (in seconds)

Fsig = 1; % Frequency of signal (in Hz)

Fs = 100; % Sampling frequency (in Hz)

Amp = 1; % Amplitude of signal (in V)

DC = 0; % DC component of signal (in V)

SNR\_1 = 10; % Signal-to-noise ratio for case a (in dB)

SNR\_2 = 25; % Signal-to-noise ratio for case b (in dB)

% Calculate signal parameters

T = 1/Fsig; % Period of signal (in seconds)

N = round(P\*Fs); % Number of samples

t = (0:N-1)/Fs; % Time vector (in seconds)

s = Amp\*sin(2\*pi\*Fsig\*t)+DC; % Signal

% Calculate noise parameters

URMS\_signal = sqrt((DC^2+(Amp^2)/2)); % RMS value of signal

URMS\_noise\_1 = URMS\_signal/10^(SNR\_1/20); % RMS value of noise for case a

noise\_1 = URMS\_noise\_1\*randn(size(t)); % Noise signal for case a

URMS\_noise\_2 = URMS\_signal/10^(SNR\_2/20); % RMS value of noise for case b

noise\_2 = URMS\_noise\_2\*randn(size(t)); % Noise signal for case b

% Plot signals for case a

figure(1)

plot(t, s)

hold on

plot(t, noise\_1)

title('Sinusoidal Signal and Noise (SNR = 10 dB)')

xlabel('Time (s)')

ylabel('Amplitude (V)')

legend('Signal', 'Noise')

% Plot signals for case b

figure(2)

plot(t, s)

hold on

plot(t, noise\_2)

title('Sinusoidal Signal and Noise (SNR = 25 dB)')

xlabel('Time (s)')

ylabel('Amplitude (V)')

legend('Signal', 'Noise')

% Add noise to signal for case a

x\_1 = s + noise\_1;

% Calculate actual SNR for case a

Psignal\_1 = Amp^2/2;

Pnoise\_1 = URMS\_noise\_1^2;

SNR\_actual\_1 = 10\*log10(Psignal\_1/Pnoise\_1);

% Plot signal with noise for case a

figure(3)

plot(t, x\_1)

title(sprintf('Signal with Noise (SNR = %.2f dB)', SNR\_actual\_1))

xlabel('Time (s)')

ylabel('Amplitude (V)')

% Add noise to signal for case b

x\_2 = s + noise\_2;

% Calculate actual SNR for case b

Psignal\_2 = Amp^2/2;

Pnoise\_2 = URMS\_noise\_2^2;

SNR\_actual\_2 = 10\*log10(Psignal\_2/Pnoise\_2);

% Plot signal with noise for case b

figure(4)

plot(t, x\_2)

title(sprintf('Signal with Noise (SNR = %.2f dB)', SNR\_actual\_2))

xlabel('Time (s)')

ylabel('Amplitude (V)')

**Comment**: This code generates two sinusoidal signals with different levels of noise and plots them. The first signal has a signal-to-noise ratio (SNR) of 10 dB, while the second signal has an SNR of 25 dB. The code also calculates the actual SNR for each signal after adding noise and plots the resulting signals. The use of random noise ensures that each time the code is run, the noise signals will be different, allowing for different noise scenarios to be tested. The code is well-commented and uses variable names that are easy to understand, making it easy to follow and modify for different applications. Overall, this code is a good example of how to generate and analyze signals with different levels of noise.