Частотная модуляция

Основы

Для запуска скриптов необходимо корневую папку репозитория сделать рабочей папкой Matlab!

1. Введение

Одним из наиболее распространенных видов аналоговой модуляции является частотная модуляция. Ее преимущество относительно амплитудной модуляции состоит в лучшей помехоустойчивости. Однако, это достигается за счет большей полосы сигнала и усложнения приемного устройства. При частотной модуляции мгновенная частота несущей изменяется в соответвии с амплитудой информационного сообщения.

В общем случае передаваемый сигнал можно представить в виде:

$$s(t) = A(t) \cdot \cos(\theta(t)), \qquad (1)$$

где $A(t)$ - амплитуда сигнала, $\theta(t)$ - фаза сигнала.

При амплитудной модуляции значение амплитуды $A(t)$ изменяется пропорционально информационному сообщению $m(t)$, а фаза изменяется линейно, как $\theta(t) = \omega_c t$, где ω_c - частота несущей.

При частотной модуляции амплитуда сигнала остается постоянной $A(t) = A_c$, а пропорционально информационному сообщению $m(t)$ изменяется мгновенная частота несущей, которая выражается через фазу $\theta(t)$ с помощью производной:

$$\frac{d}{\mathrm{dt}}\theta(t) = \omega(t) = 2\pi \cdot f(t).$$

Мгновенная частота в герцах $f(t)$ обычно представляется как сумма значения частоты несущей f_c и информационного сообщения $m(t)$, умноженного на константу K_f, которая называется чувствительностью модулятора (frequency sensetivity):

$$f(t) = f_c + K_f \cdot m(t).$$

Выразим фазу сигнала через его мгновенную частоту с помощью интегрирования:

$$\theta(t) = \int_{-\infty}^{t} 2\pi \cdot f(t) \cdot \mathrm{dt} = 2\pi \cdot \int_{-\infty}^{t} [f_c + K_f \cdot m(t)] \cdot \mathrm{dt} = 2\pi f_c t + 2\pi K_f \cdot \int_{-\infty}^{t} m(t) \cdot \mathrm{dt} + \theta_0, \qquad (2)$$

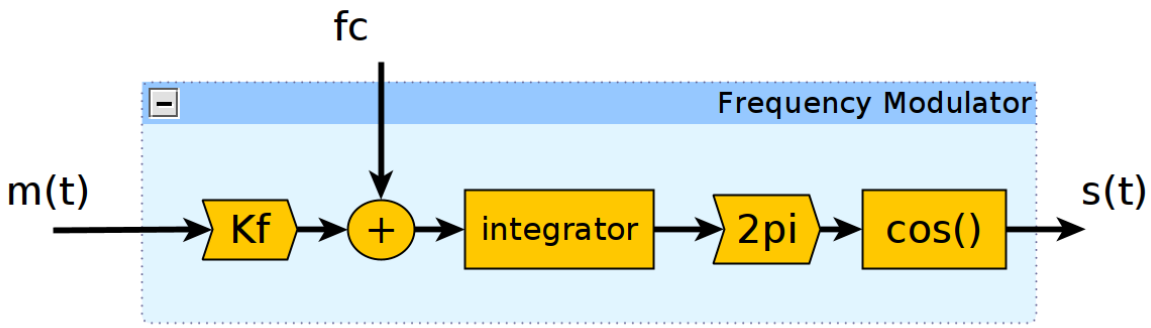
где θ_0 - начальная фаза, которую в дальнейшем будем считать равной нулю.

Из уравнения (2) можно увидеть, что фаза сигнала изменяется пропорционально интегралу от информационного сообщения. Подставляя уравнение (2) в формулу (1), получим общее выражение для сигнала с частотной модуляцией:

$$s(t) = A(t) \cdot \cos(\theta(t)) = A_c \cdot \cos\left(2\pi f_c t + 2\pi K_f \cdot \int_{-\infty}^{t} m(\tau) \cdot d\tau\right) \qquad (3),$$

где A_c - амплитуда несущей, f_c - частота несущей, K_f - чувствительность модулятора.

Схема частотного модулятора представлена ниже:



Введем еще два важных понятия. Мгновенная частота сигнала в герцах равна:

$$f(t) = f_c + K_f \cdot m(t).$$

Максимальное отклонение мгновенной частоты от частоты несущей называют девиацией (frequency deviation):

$$\Delta f = \max|f(t) - f_c| = \max|f_c + K_f \cdot m(t) - f_c| = \max|K_f \cdot m(t)| = K_f \cdot \max|m(t)|.$$

Таким образом, девиация зависит от чувствительности модулятора и максимального значения информационного сообщения.

Индексом модуляции (modulation index) называют отношение девиации к максимальной частоте в спектре информационного сообщения:

$$\beta = \frac{\Delta f}{f_m} = \frac{K_f \cdot \max|m(t)|}{f_m},$$

где f_m - максимальная частота в спектре информационного сообщения.

Как будет показано в дальнейшем, девиация определяется полосу модулированного сигнала. Чем больше девиация, тем шире полоса. Индекс модуляции - это просто девиация, нормированная на полосу сообщения. Индексом модуляции удобно пользоваться для определения является ли модуляция узкополосной или широкополосной.

2. Однотональная модуляция

Рассмотрим наиболее простой случай модуляции однотонольным сообщением. Информационное сообщение представляет из себя гармонический сигнал вида:

$$m(t) = A_m \cos(2\pi f_m t),$$

где A_m - амплитуда тона, f_m - частота в герцах.

Максимальное значение однотонального сообщения равно его амплитуде A_m. Отсюда получаем, что девиация Δf равна произведения $K_f A_m$. Далее, так как спектр сообщения состоит всего из одной гармоники, максмальная

частота в спектре равна частоте этой гармоники f_m. Таким образом, в случае однотональной модуляции индекс модуляции задается как

$$\beta = \frac{\Delta f}{f_m} = \frac{K_f A_m}{f_m}.$$

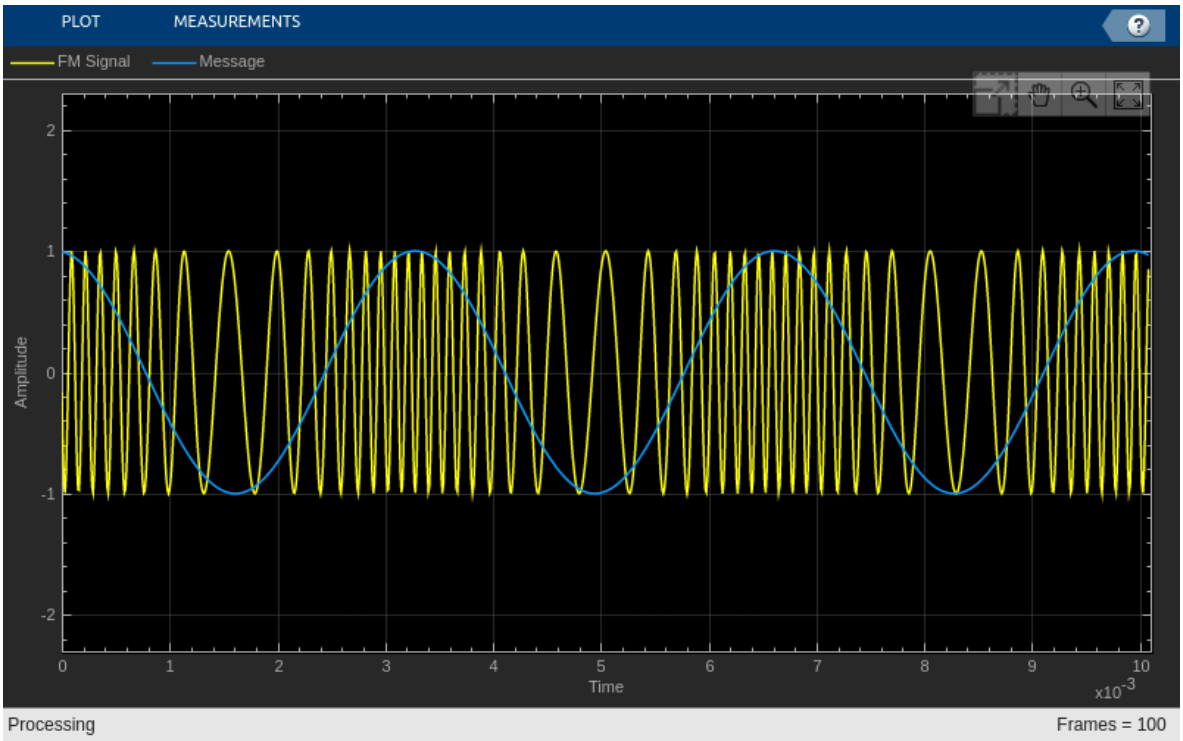
Ниже представлен скрипт, реализующий однотональную частотную модуляцию. Модуляция несущей выполняется с помощью уравнения (3). Сначала рассчитывается мгновенная частота несущей $f(t) = f_c + K_f \cdot m(t)$. Далее, она интегрируется с помощью функции cumsum(), и получается фаза сигнала. Эта фаза используется для формирования передаваемого сигнала FmSignal. Чувствительность модулятора задается с помощью индекса модуляции в виде:

$$K_f = \frac{\beta \cdot f_m}{A_m}.$$

Изменяя индекс модуляции β, заданный с помощью переменной ModIndex, можно увидеть его влияние на временную диаграмму сигнала.

```
clc; clear; close all;

FrameSize = 1011;                % размер обрабатываемой за один раз пачки данных
Fs = 100e3;                      % частота дискретизации (Hz)
Fc = 5e3;                        % частота несущей (Hz)
Am = 1;                          % амплитуда тонального сообщения
Fm = 300;                        % частота тонального сообщения (Hz)
ModIndex = 10;   % индекс модуляции

% вычисление чувствительности модулятора
Kf = ModIndex * Fm / Am;

% генератор однотонального сигнала
Message = dsp.SineWave(...
    'SampleRate',Fs,...
    'SamplesPerFrame',FrameSize,...
    'Frequency', Fm,...
    'Amplitude', Am ...
    );

% объект для отрисовки графиков
Plotter = dsp.ArrayPlot(...
    'PlotType', 'Line', ...
    'YLimits', [-2.3, 2.3], ...
    'SampleIncrement', 1/Fs, ...
    'XLabel', 'Time', ...
    'YLabel', 'Amplitude', ...
    'ChannelNames', {'FM Signal', 'Message'} ...
    );

% начальная фаза интегратора
InitPhase = 0;

% запуск симуляции
```

```
for i = 1:100
    % формирование тонального сигнала
    MessageData = Message();

    % вычисление мгновенной частоты (Hz)
    freq = Fc + Kf * MessageData;

    % вычисление фазы в радианах с помощью интегрирования
    phase = InitPhase + 2*pi/Fs * cumsum(freq);

    % обновление начальной фазы интегратора
    InitPhase = phase(end);

    % формирование модулированного сигнала
    FmSignal = cos(phase);

    % вывод результатов на график
    Plotter([FmSignal, MessageData])

    % задержка в 0.1 секунды для лучшей визуализации
    pause(0.1)
end
```



На временной диаграмме можно увидеть однотональное сообщение (синий цвет) и частотно-модулированный сигнал (желтый цвет). Частота сигнала изменяется пропорционально амплитуде информационного сообщения. Чем больше значение сообщения, тем больше частота и наоборот. Также можно заметить, что при увеличении индекса модуляции изменения частоты несущей становятся более выраженными.

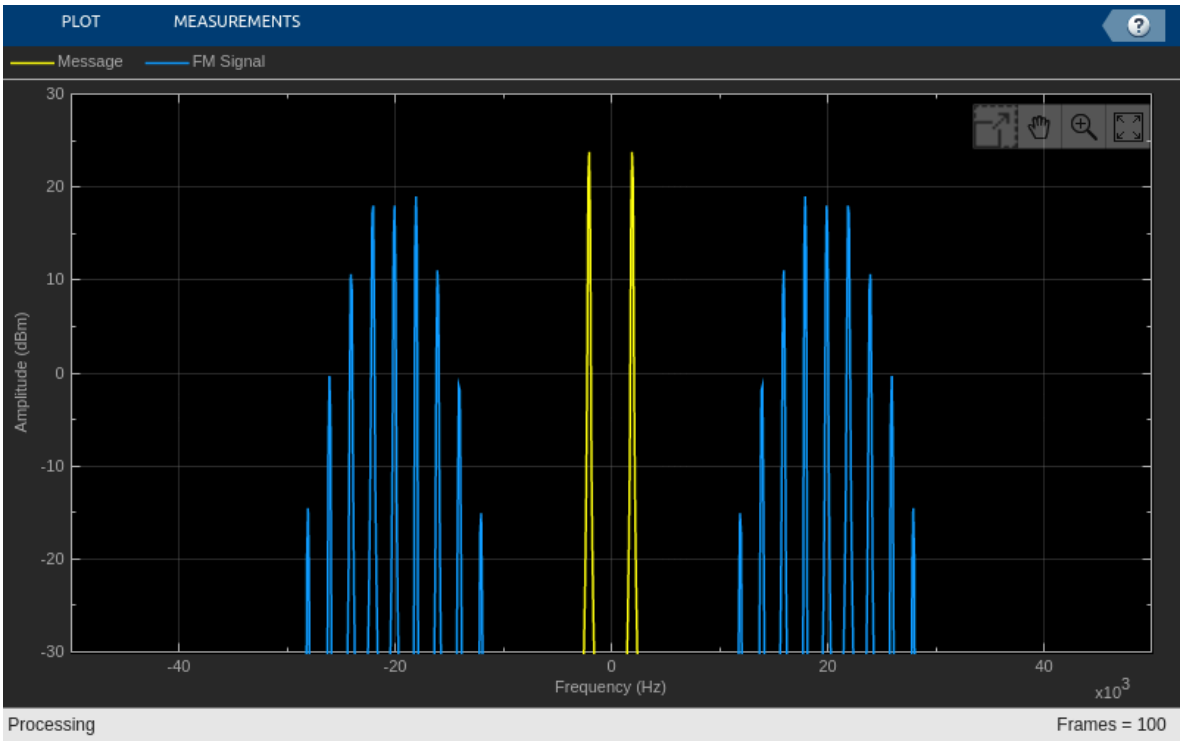
Теперь рассмотрим, как выглядит спектр сигнала при однотональной частотной модуляции. В отличие от амплитудной модуляции, которая является линейной, и при которой спектр информационного сообщения

просто смещается по частоте, спектр сигнал при частотной модуляции устроен более сложным образом. Он существенным образом зависит от индекса модуляции и может содержать спектральные компоненты, которых не было в исходном сообщении.

Скрипт, позволяющий увидеть спектр сигнала при однотональной частотной модуляции, представлен ниже. Зависимость спектра от индекса модулции для однотонального случая будет более подробно раскрыта в LiveScripts по узкополосной и широкополосной частотной модуляции.

```
clc; clear; close all;

FrameSize = 1011;                % размер обрабатываемой за один раз пачки данных
Fs = 100e3;                      % частота дискретизации (Hz)
Fc = 20e3;                       % частота несущей (Hz)
Am = 1;                          % амплитуда тонального сообщения
Fm = 2e3;                        % частота тонального сообщения (Hz)
ModIndex = 1.5;   % индекс модуляции

% вычисление чувствительности модулятора
Kf = ModIndex * Fm / Am;

% генератор однотонального сигнала
Message = dsp.SineWave(...
    'SampleRate',Fs,...
    'SamplesPerFrame',FrameSize,...
    'Frequency', Fm,...
    'Amplitude', Am ...
    );

% объект для вычисления спектра
SpecEstimator = dsp.SpectrumEstimator(...
    'PowerUnits','dBm', ...
    'FrequencyRange', 'centered', ...
    'SampleRate', Fs ...
    );

% объект для отрисовки графиков
Plotter = dsp.ArrayPlot(...
    'PlotType','Line', ...
    'XOffset', -Fs/2, ...
    'YLimits', [-30, 30], ...
    'XLabel', 'Frequency (Hz)', ...
    'YLabel', 'Amplitude (dBm)', ...
    'ChannelNames', {'Message', 'FM Signal'}, ...
    'SampleIncrement', Fs/FrameSize ...
    );

% начальная фаза интегратора
InitPhase = 0;

% запуск симуляции
for i = 1:100
    % формирование тонального сигнала
    MessageData = Message();
```

```
    % вычисление мгновенной частоты (Hz)
    freq = Fc + Kf * MessageData;

    % вычисление фазы в радианах с помощью интегрирования
    phase = InitPhase + 2*pi/Fs * cumsum(freq);

    % обновление начальной фазы интегратора
    InitPhase = phase(end);

    % формирование модулированного сигнала
    FmSignal = cos(phase);

    % вычисление спектров
    Spectrums = SpecEstimator([MessageData, FmSignal]);

    % вывод результатов на график
    Plotter(Spectrums)

    % задержка в 0.1 секунды для лучшей визуализации
    pause(0.1)
end
```



На графике спектр информационного тонального сообщения представлен желтым цевтом и состоит из двух дельта-функций на частотах f_m и $-f_m$. Спектр модулированного сигнала обозначен синим цветом. При малом значении индекса модуляции β спектр содержит три тона: несущую и две боковых компоненты. Это очень напоминает спектр АМ-сигнал при однотональной модуляции. Однако, при увиличении индекса модуляции β в спектре появляется множетсво гармоник и его ширина существенно возрастает. Таким образом, как уже упоминалось ранее, спектр ЧМ-сигнала имеет более сложную структуру, по сравнению с АМ-сигналом.

3. Модуляция несколькими тонами

Теперь рассмотрим более сложный случай, когда информационный сигнал представляет из себя сумму гармонических сигналов:

$$m(t) = \sum_{i=1}^{N} A_i \cos(2\pi f_i t) \qquad (4),$$

где A_i - амплитада i-ой гармоники, f_i - частота i-ой гармоники, N - число гармоник.

Максимальное значение, которое может принимать информационное сообщение равно сумме амплитуд отдельных гармоник, то есть

$$\max|m(t)| = \sum_{i=1}^{N} A_i.$$

Отсюда получаем, что девиация Δf равна произведению $K_f \cdot \sum_{i=1}^{N} A_i.$ Далее, максмальная частота в спектре сообщения равна частоте самой высокой гармоники. Если считать, что в (4) в сумме гармоники располагаются по возрастанию частоты, то максимальная частота в спектре сообщения равна $f_m = f_N \cdot$ Значит, при модуляции несколькми тонами индекс модуляции можно записать в виде:

$$\beta = \frac{\Delta f}{f_m} = \frac{K_f \cdot \sum_{i=1}^{N} A_i}{f_N},$$

где *N*- число гармоник.

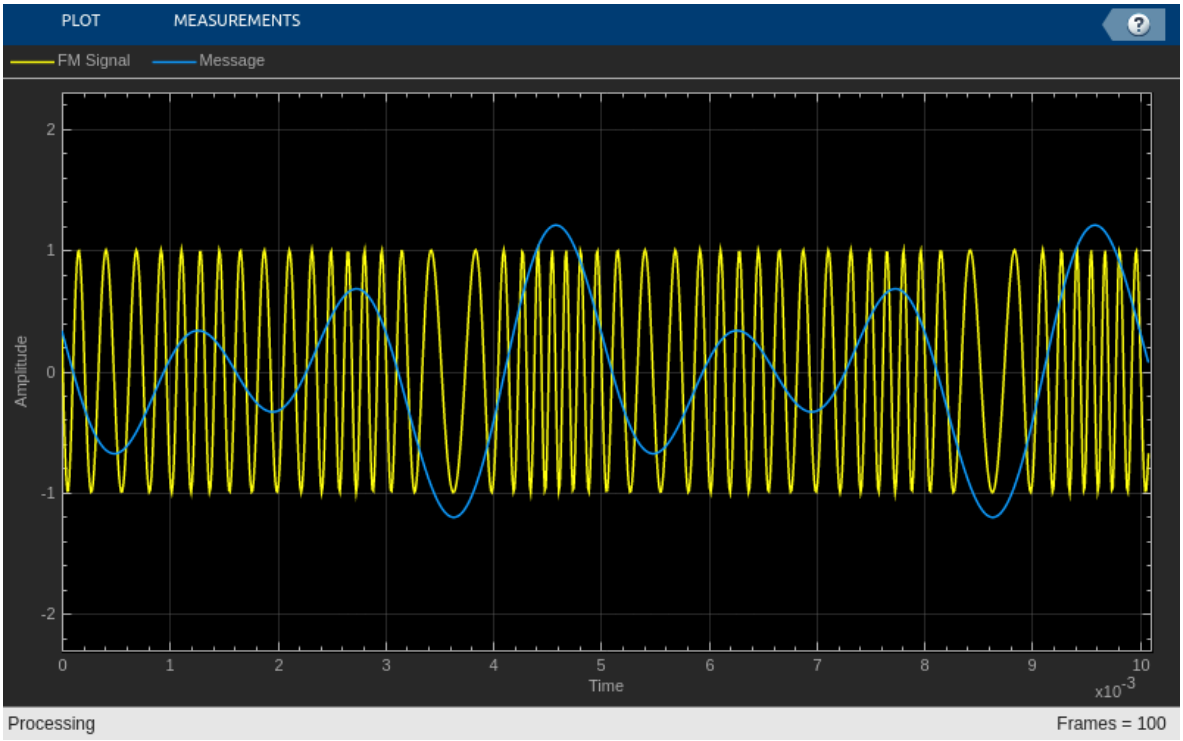
Тогда чувствительность модулятора можно представить в виде:

$$K_f = \frac{\beta \cdot f_N}{\sum_{i=1}^{N} A_i}.$$

Если информационное сообщение состоит из трех гармоник, временная диаграмма сигнала будет иметь вид:

```
clc; clear; close all;

FrameSize = 1011;                % размер обрабатываемой за один раз пачки данных
Fs = 100e3;                      % частота дискретизации (Hz)
Fc = 5e3;                        % частота несущей (Hz)
Am = [0.1 0.5 0.7];              % амплитуды тональнов сообщения
Fm = [200 400 600];              % частоты тональнов сообщения (Hz)
ModIndex = 5;   % индекс модуляции

% вычисление чувствительности модулятора
Kf = ModIndex * Fm(end) / sum(Am);

% генератор сообщения из трех тонов
Message = dsp.SineWave(...
```

```
    'SampleRate', Fs, ...
    'SamplesPerFrame', FrameSize, ...
    'Frequency', Fm, ...
    'Amplitude', Am ...
    );

% объект для отрисовки графиков
Plotter = dsp.ArrayPlot(...
    'PlotType', 'Line', ...
    'YLimits', [-2.3, 2.3], ...
    'SampleIncrement', 1/Fs, ...
    'XLabel', 'Time', ...
    'YLabel', 'Amplitude', ...
    'ChannelNames', {'FM Signal', 'Message'} ...
    );

% начальная фаза интегратора
InitPhase = 0;

% запуск симуляции
for i = 1:100
    % формирование сообщения из нескольких тонов
    MessageData = Message();
    MessageData = MessageData(:,1) + MessageData(:,2) + MessageData(:,3);

    % вычисление мгновенной частоты (Hz)
    freq = Fc + Kf * MessageData;

    % вычисление фазы в радианах с помощью интегрирования
    phase = InitPhase + 2*pi/Fs * cumsum(freq);

    % обновление начальной фазы интегратора
    InitPhase = phase(end);

    % формирование модулированного сигнала
    FmSignal = cos(phase);

    % вывод результатов на график
    Plotter([FmSignal, MessageData])

    % задержка в 0.1 секунды для лучшей визуализации
    pause(0.1)
end
```

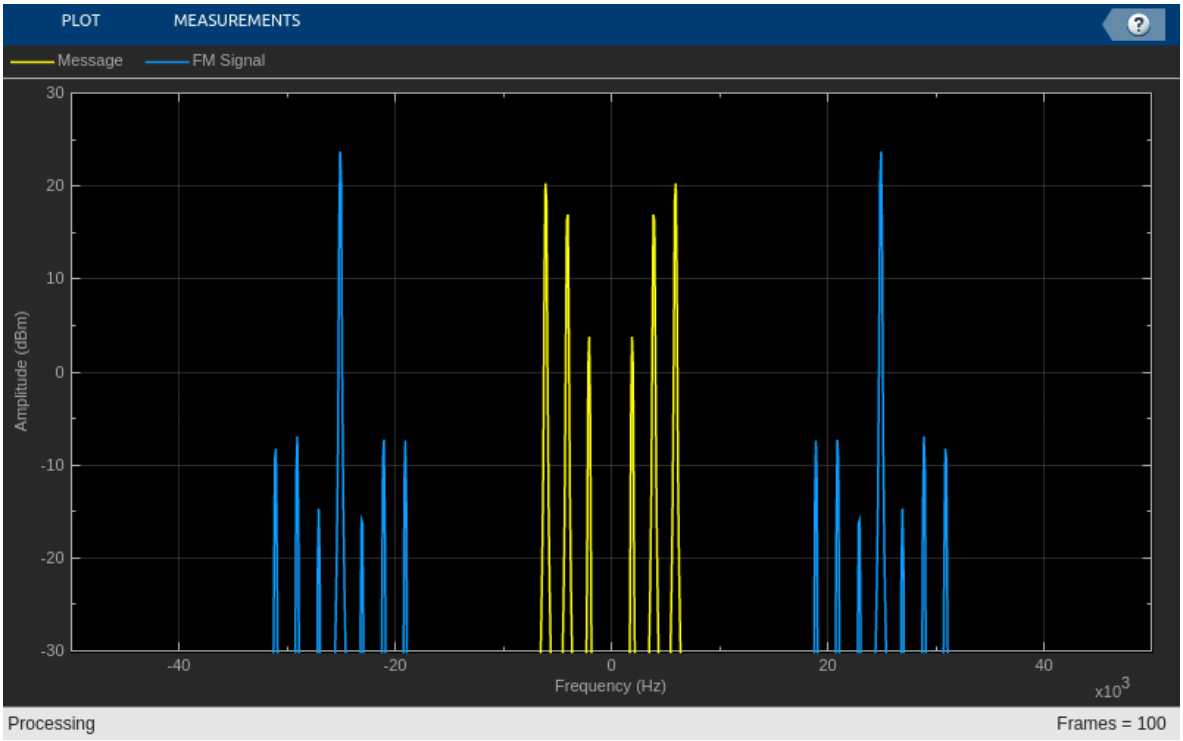


Как и для однотональной модуляции, можно увидеть, что частота несущей пропорциональна амплитуде информационного сообщения. Причем, при увеличении индекса модуляции эта зависимость выражается более ярко.

Рассмотрим спектр ЧМ-сигнала при модуляции несколькими тонами. Скрипт, позволяющий увидеть спектр сигнала, представлен ниже:

```
clc; clear; close all;

FrameSize = 1011;                % размер обрабатываемой за один раз пачки данных
Fs = 100e3;                      % частота дискретизации (Hz)
Fc = 25e3;                       % частота несущей (Hz)
Am = [0.1 0.5 0.7];              % амплитуды тональнов сообщения
Fm = [2e3 4e3 6e3];              % частоты тональнов сообщения (Hz)
ModIndex = 0.1;   % индекс модуляции

% вычисление чувствительности модулятора
Kf = ModIndex * Fm(end) / sum(Am);

% генератор сообщения из трех тонов
Message = dsp.SineWave(...
    'SampleRate', Fs, ...
    'SamplesPerFrame', FrameSize, ...
    'Frequency', Fm, ...
    'Amplitude', Am ...
    );

% объект для вычисления спектра
SpecEstimator = dsp.SpectrumEstimator(...
```

```
    'PowerUnits','dBm', ...
    'FrequencyRange', 'centered', ...
    'SampleRate', Fs ...
    );

% объект для отрисовки графиков
Plotter = dsp.ArrayPlot(...
    'PlotType','Line', ...
    'XOffset', -Fs/2, ...
    'YLimits', [-30, 30], ...
    'XLabel', 'Frequency (Hz)', ...
    'YLabel', 'Amplitude (dBm)', ...
    'ChannelNames', {'Message', 'FM Signal'}, ...
    'SampleIncrement', Fs/FrameSize ...
    );

% начальная фаза интегратора
InitPhase = 0;

% запуск симуляции
for i = 1:100
    % формирование сообщения из нескольких тонов
    MessageData = Message();
    MessageData = MessageData(:,1) + MessageData(:,2) + MessageData(:,3);

    % вычисление мгновенной частоты (Hz)
    freq = Fc + Kf * MessageData;

    % вычисление фазы в радианах с помощью интегрирования
    phase = InitPhase + 2*pi/Fs * cumsum(freq);

    % обновление начальной фазы интегратора
    InitPhase = phase(end);

    % формирование модулированного сигнала
    FmSignal = cos(phase);

    % вычисление спектров
    Spectrums = SpecEstimator([MessageData, FmSignal]);

    % вывод результатов на график
    Plotter(Spectrums)

    % задержка в 0.1 секунды для лучшей визуализации
    pause(0.1)
end
```



Как и в предыдущем примере, при малом значении индекса модуляции β спектр похож на спектр АМ-сигнала. Он состоит из тона на несущей частоте и двух боковых полос. При увиличении индекса модуляции β в спектре появляется множетсво гармоник и его ширина существенно возрастает.

4. Модуляция аудиосигналом

Рассмотрим, как будет выглядеть спектр сигнала после модуляции, если информационное сообщение является аудиосигналом. В файле Audio_Source.wav записано звуковое сообщение с частотой дискретизации $f_s = 44.1$ kHz. Частота несущей будет равна 100 kHz. Чтобы избежать наложения спектров, выберем итоговую частоту дискретизации модулированного сигнала в 10 раз больше частоты аудиосигнала, то есть 10 * 44.1 kHz. Это потребует провести интерполяцию для увеличения частоты дискретизации информационного сообщения.

Максимальное значение отсчетов данного конкретного аудиосигнала равно единице, то есть $\max|m(t)| = 1.$ Также будем считать, что максимальная частота в спектре сообшения равна половине частоты дискретизации:

$$f_m = \frac{f_s}{2} = 22.05\ \text{kHz}.$$

Отсуда, индекс модуляции можно записать в виде:

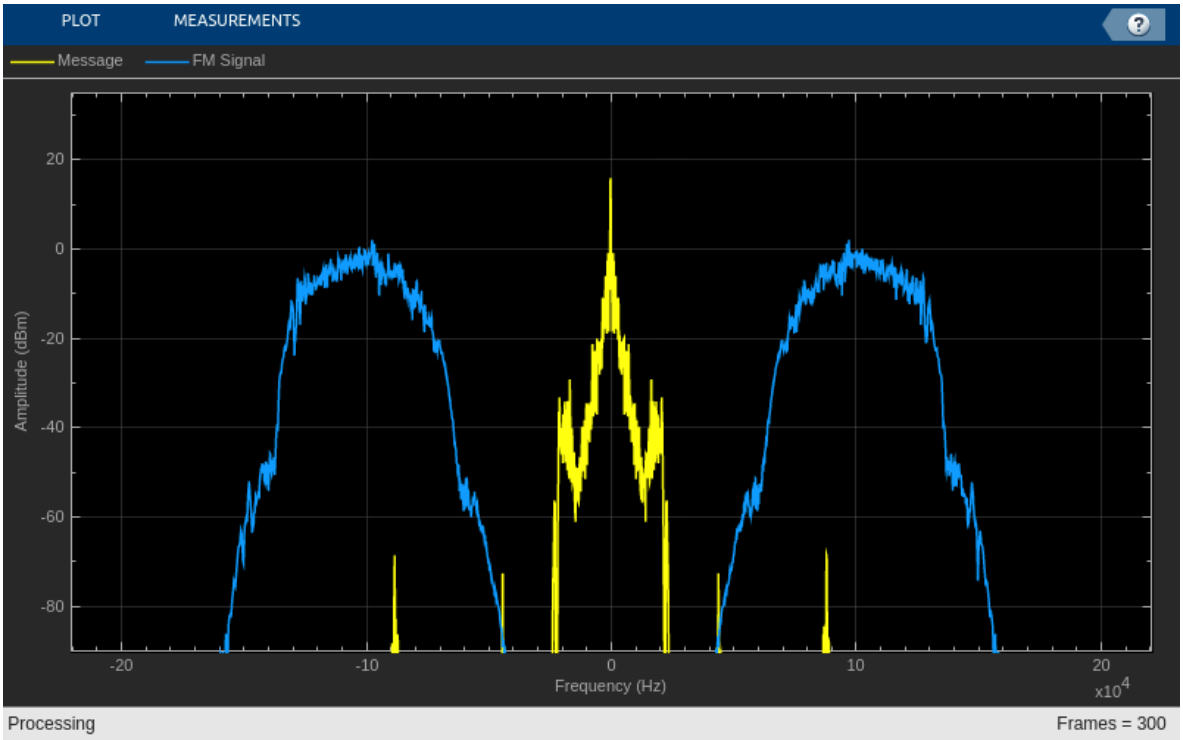
$$\beta = \frac{K_f \cdot \max|m(t)|}{f_m} = 2 \cdot \frac{K_f}{f_s}.$$

Тогда чувствиельность модулятора будет равна:

$$K_f = \frac{\beta \cdot f_s}{2}.$$

```
clc; clear; close all;

AudioFrameSize = 1000;          % количество отсчетов аудиофайла, получаемых за один раз
FramesNumber = 300;             % число обрабатываемых пачек данных
RateRatio = 10;                 % коэффициент увеличения частоты дискретизации
Fc = 100e3;                     % частота несущей (Hz)
ModIndex = 1.5;   % индекс модуляции

% объект для считываения отсчетов аудиофайла
AudioReader = dsp.AudioFileReader(...
    'wav/Audio_Source.wav', ...
    'SamplesPerFrame',AudioFrameSize...
    );

% дополнительные расчеты
AudioFs = AudioReader.SampleRate;               % получаем частоту дискретизации аудиосообщения
SignalFs = AudioFs * RateRatio;                 % частота дискретизации модулированного сигнала
SignalFrameSize = AudioFrameSize * RateRatio;   % количество отсчетов Ам-сигнала, получаемых за один раз

% вычисление чувствительности модулятора
Kf = ModIndex * AudioFs / 2;

% интерполятор
Upsampler = dsp.SampleRateConverter(...
    'Bandwidth', 40e3, ...
    'InputSampleRate',AudioFs, ...
    'OutputSampleRate', SignalFs ...
    );

% объект для вычисления спектра
SpecEstimator = dsp.SpectrumEstimator(...
    'PowerUnits','dBm',...
    'FrequencyRange','centered',...
    'SampleRate',SignalFs);

% объект для отрисовки графиков
Plotter = dsp.ArrayPlot(...
    'PlotType','Line', ...
    'XOffset', -SignalFs/2, ...
    'YLimits', [-90, 35], ...
    'XLabel', 'Frequency (Hz)', ...
    'YLabel', 'Amplitude (dBm)', ...
    'ChannelNames', {'Message', 'FM Signal'}, ...
    'SampleIncrement', SignalFs/SignalFrameSize ...
    );

% начальная фаза интегратора
InitPhase = 0;

% запуск симуляции
for i = 1:FramesNumber
    % считывание отсчетов аудиосообщения и выделение одного канала из
    % стерео сигнала
```

```
    AudioData = AudioReader();
    AudioData = AudioData(:,1);

    % увеличение частоты дискретизации аудиосообщения
    MessageData = Upsampler(AudioData);

    % вычисление мгновенной частоты (Hz)
    freq = Fc + Kf * MessageData;

    % вычисление фазы в радианах с помощью интегрирования
    phase = InitPhase + 2*pi/SignalFs * cumsum(freq);

    % обновление начальной фазы интегратора
    InitPhase = phase(end);

    % формирование модулированного сигнала
    FmSignal = cos(phase);

    % вычисление спектров
    SpectrumData = SpecEstimator([MessageData, FmSignal]);

    % вывод результатов на график
    Plotter(SpectrumData)

    % задержка в 0.1 секунды для лучшей визуализации
    pause(0.05)
end
```



Можно увидеть, что при увеличении индекса модуляции ширина спектра увеличивается, и он становится все менее похожим на спектр информационного сообщения.

5. Готовые решения Matlab

В среде Matlab есть готовая функция для выполнения частотной модуляции: fmmod(). Проверим ее работу. Функция принимает на вход информационный сигнал, частоту несущей, частоту дискретизации, девиацию и начальную фазу несущей.

С учетом рассуждений из предыдущего параграфа девиацию можно выразить следующим образом:

$$\Delta f = K_f \cdot \max|m(t)| = K_f \cdot 1 = K_f.$$

Основная часть скрипта совпадает со скриптом из предыдущего раздела.

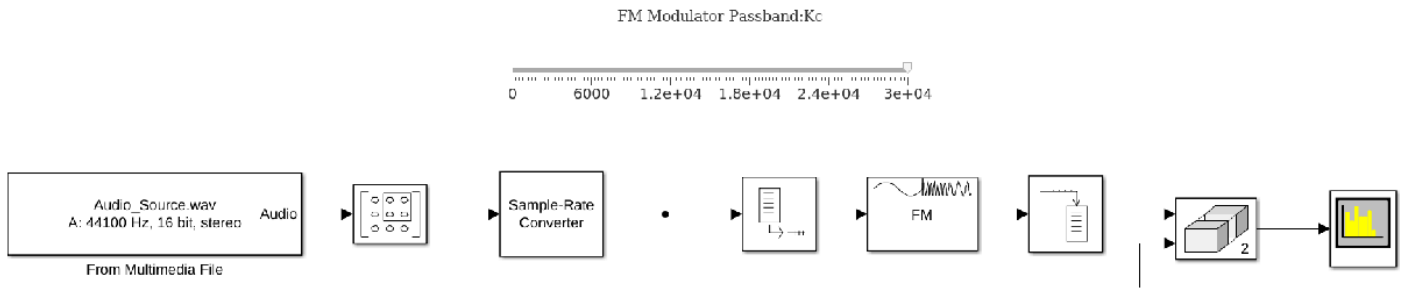
```
clc; clear; close all;

AudioFrameSize = 1000;          % количество отсчетов аудиофайла, получаемых за один раз
FramesNumber = 300;             % число обрабатываемых пачек данных
RateRatio = 10;                 % коэффициент увеличения частоты дискретизации
Fc = 100e3;                     % частота несущей (Hz)
ModIndex = 1.5;  % индекс модуляции

% объект для считываения отсчетов аудиофайла
AudioReader = dsp.AudioFileReader(...
    'wav/Audio_Source.wav', ...
    'SamplesPerFrame',AudioFrameSize...
    );

% дополнительные расчеты
AudioFs = AudioReader.SampleRate;               % получаем частоту дискретизации аудиосообщения
SignalFs = AudioFs * RateRatio;                 % частота дискретизации модулированного сигнала
SignalFrameSize = AudioFrameSize * RateRatio;   % количество отсчетов Ам-сигнала, получаемых за один раз

% вычисление чувствительности модулятора и девиации
Kf = ModIndex * AudioFs / 2;
FreqDeviation = Kf;

% интерполятор
Upsampler = dsp.SampleRateConverter(...
    'Bandwidth', 40e3, ...
    'InputSampleRate',AudioFs, ...
    'OutputSampleRate', SignalFs ...
    );

% объект для вычисления спектра
SpecEstimator = dsp.SpectrumEstimator(...
    'PowerUnits','dBm',...
    'FrequencyRange','centered',...
    'SampleRate',SignalFs);

% объект для отрисовки графиков
Plotter = dsp.ArrayPlot(...
    'PlotType','Line', ...
    'XOffset', -SignalFs/2, ...
    'YLimits', [-90, 35], ...
    'XLabel', 'Frequency (Hz)', ...
```

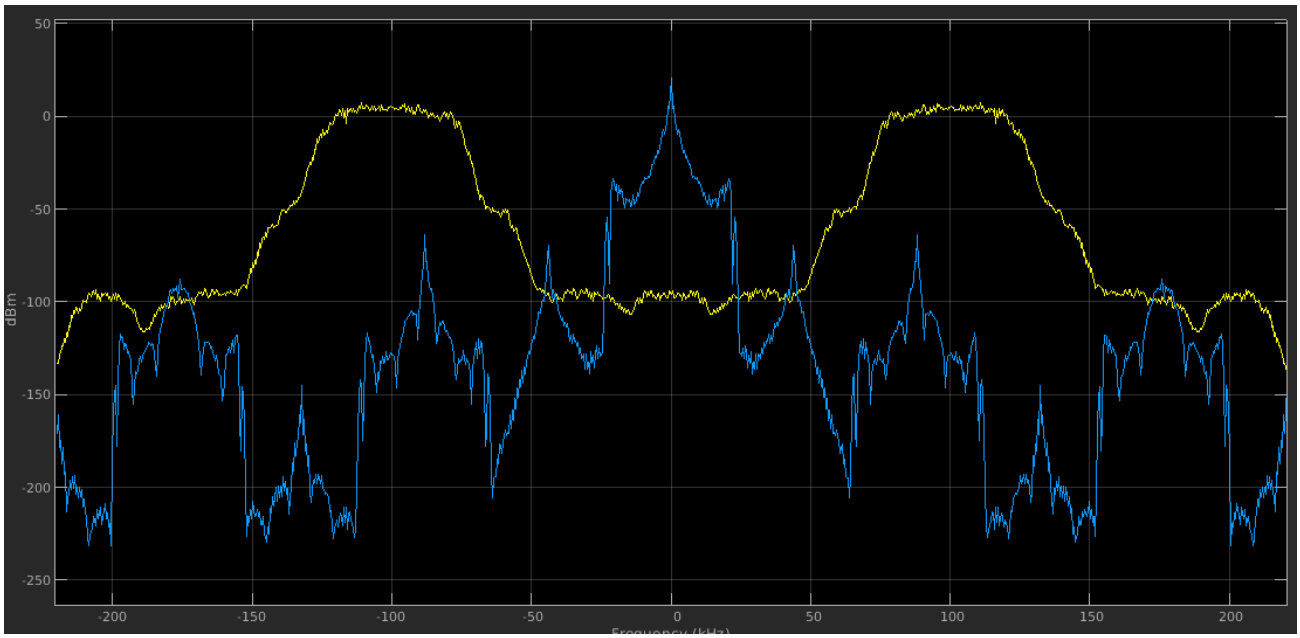
```
    'YLabel', 'Amplitude (dBm)', ...
    'ChannelNames', {'Message', 'FM Signal'}, ...
    'SampleIncrement', SignalFs/SignalFrameSize ...
    );

% начальная фаза интегратора
InitPhase = 0;

% запуск симуляции
for i = 1:FramesNumber
    % считывание отсчетов аудиосообщения и выделение одного канала из
    % стерео сигнала
    AudioData = AudioReader();
    AudioData = AudioData(:,1);

    % увеличение частоты дискретизации аудиосообщения
    MessageData = Upsampler(AudioData);

    % частотная модуляция аудиосообщения
    FmSignal = fmmod(MessageData, Fc, SignalFs, FreqDeviation, InitPhase);

    % обновление начальной фазы в радианах несущей после модуляции
    phase = InitPhase + 2*pi/SignalFs * cumsum(Fc + Kf * MessageData);
    InitPhase = phase(end);

    % вычисление спектров
    SpectrumData = SpecEstimator([MessageData, FmSignal]);

    % вывод результатов на график
    Plotter(SpectrumData)

    % задержка в 0.1 секунды для лучшей визуализации
    pause(0.05)
end
```

Спектр модулированного сигнала совпадает со спектром из предыдущего пункта.

6. Simulink модель

В файле FM_Transmitter_Basics_1.slx представлена Simulink-модель модуляции аудиосообщением. Как и в предыдущих скриптах в файле Audio_Source.wav записано звуковое сообщение с частотой дискретизации 44.1 kHz. Частота несущей равна 100 kHz. Перед модуляцией частота дискретизации увеличивается в 10 раз. Блок FM принимает и выдает данные только по одному отсчету. Поэтому до этого блока данные преобразуются из вектора в отсчеты (Unbuffer), а после него опять собираются в вектор (Buffer). С помощью слайдера можно изменить значение девиации и увидеть ее влияние на спектр сигнала. Схема модели и спектры сигналов представлены ниже:





7. Прямой метод частотной модуляции

Рассмотренный выше метод частотной модуляции с помощью формирования мгновенной частоты сигнала и ее интегрирования для получения фазы называют прямым методом. На практике для реализыции этого метода удобно использовать генератор, управляемый напряжением (Voltage Control Oscillator - VCO). Если сигнал формируется цифровыми методами, например с помощью SDR, такой генератор также называют Numerical Control Oscillator (NCO).

Принцип работы VCO следующий. Мгновенная частота генератора линейно зависит от напряжения входного сигнала:

$$f_{\text{vco}}(t) = f_0 + K_{\text{vco}} \cdot m(t),$$

где K_{vco} - коэффициент усиления генератора (Hz/Volt), f_0 - частота генератора (Hz) при нулевом входном воздействии $m(t)$ (free-running frequency).

Фаза генератора связана с мгновенной частотой с помощью интеграла:

$$\theta_{\text{vco}}(t) = 2\pi \cdot \int_{-\infty}^{t} f_{\text{vco}}(\tau)d\tau = 2\pi \cdot \int_{-\infty}^{t} (f_0 + K_{\text{vco}} \cdot m(\tau))d\tau.$$

Выходной сигнала VCO - это гармоническое колебание, фаза которого равна $\theta_{\text{vco}}(t)$:

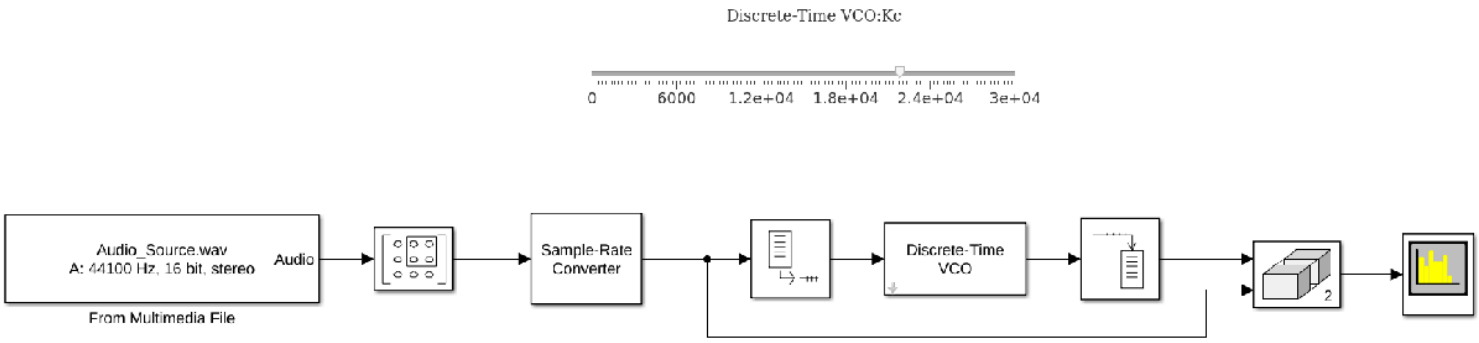
$$s_{\text{vco}}(t) = \cos(\theta_{\text{vco}}(t)) = \cos\left(2\pi \cdot \int_{-\infty}^{t} (f_0 + K_{\text{vco}} \cdot m(\tau))d\tau.\right).$$

Выполняя интегрирование, получим окончательный результат:

$$s_{\text{vco}}(t) = \cos\left(2\pi f_0 t + 2\pi K_{\text{vco}} \cdot \int_{-\infty}^{t} m(\tau)d\tau.\right).$$

Можно увидеть, что данное выражение с точностью до обозначений совпадает с уравнением (3). Таким образом, с помощью VCO можно реализовать прямой метод частотной модуляции. При этом его коэффициент усиления K_{vco} будет задавать чувствительность модулятора K_f, а его free-running frequency f_0 будет соответствовать частоте несущей f_c.

В файле FM_Transmitter_Basics_2.slx представлена Simulink-модель, реализующая прямой метод модуляции аудиосообщением с помощью VCO. Как и в предыдущих скриптах в файле Audio_Source.wav записано звуковое сообщение с частотой дискретизации 44.1 kHz. Частота несущей равна 100 kHz. Перед модуляцией частота дискретизации увеличивается в 10 раз.



Литература:

1. B. P. Lathi Modern Digital and Analog Communication Systems
2. R. Stewart, K. Barlee, D. Atkinson, L. Crockett Software Defined Radio using MATLAB® & Simulink and the RTL-SDR