Lab: Upconversion and downconversion

In this lab, we will demonstrate how to:

- Generate a random baseband signal from symbols
- Simulate upconversion and downconversion
- Simulate a simple passband filter
- Measure and plot the PSD of signals

Complete all the sections labeled `TODO`. Publish the file, print to PDF and submit the PDF. Do not submit the MATLAB files.

Create a sequence of symbolss

To illustrate the process of up- and down-conversion, we will create a simple sequence of complex symbols. This type of sequence is used commonly digital transmissions where each symbol encodes bits to be transmitted. We will discuss this later.

Right now, generate a vector `xsym` of `nsym=1024` random symbols with values $(\pm 1 \pm 1)/\sqrt{2}$ complex symbol values (i.e. QPSK). You may use the command `randsample`.

```
nsym = 1024;

% TODO
%    sym = ...
```

Create the baseband signal

Let $x(t)$ be the tranmitted baseband signal $x(t)$ generated from the sequence of complex symbols `xsym`. Assume the symbol rate is `fysm=20` MHz. Create a vector x from $x(t)$ sampled at `nov=16` times the symbol rate. Also, create a corresponding time vector `t`.

```
fsym = 20;  % Frequency in MHz
nov = 16;   % Over-sampling rate
fsamp = nov*fsym;  % Sampling rate for x

% TODO
%     x = ...
%     t = ...

% TODO:  Plot x(t) vs. t for the first 8 symbols.  Use separate plots
% for the real and imaginary components.  Label the x-axis with the
% correct units.
```

Plot the PSD of the TX signal

Properly measuring the PSD is somewhat tricky. Fortunately, Matlab has an excellent routines for computing the PSD:

```
[Px, fx] = pwelch(x,hamming(512),[],[],fsamp*1e6,'centered');
```

This routine uses the Welch algorithm. Use this routine to compute and plot the PSD. Label your axes correctly. Plot the PSD in dBm/Hz. The units here assume that x is normalized such that `|x|^2` has units of mW.

```
% TODO
```

Upconvert

Create a real passband signal `xp` by upconverting `x` with a carrier frequency of `fc=80` MHz. Plot the PSD of `x` and `xp` on the same plot, so you can compare the two.

```
fc = 80;

% TODO
%     xp = ...
%     [Pp, fp] = ...
```

Passband channel

We now simulate a passband channel. Suppose the passband channel is described by the linear system:

```
dyp/dt = f0*(xp-yp)
```

for `f0=25` MHz. We can approximate this in discrete-time by:

```
yp(t+1) = yp(t) + f0*tsamp*(xp(t)-yp(t)).
```

Simulate the discrete-time filter with the `filter` command to create a sampled version of the output `yp`.

```
% TODO
%     yp = filter(...)


% TODO:  Plot the PSD of yp. Also plot the expected PSD of yp using
%
%    S_{y_p}(f) = S_{x_p}(f)|G_p(f)|^2,
%
% where G_p(f) is the passband frequency response.  Note that you can
% compute the frequency response using the freqs command.  You may notice
% a small discrepancy between the measured and expected frequency response.
% This is due to the digital implementation of the filter.

% TODO:
%     Gp = ...
```

Design a digital downconversion filter

We will design a simple digital filter to filter the downconverted signal. Use the [cheby1](https://www.mathworks.com/help/signal/ref/cheby1.html) function to create a digital fourth order filter with `f<=` 12.5 MHz. Use a passband ripple of 0.5 dB.

```
% TODO
%     [blpf,alpf] = cheby1(...)
```

Plot the filter frequency response

Use the freqz to plot the frequency response of the digital filter. Label your axes

```
% TODO
%    Hcheb = freqz(...)
%    plot(...)
```

Downconvert the signal digital domain

Downconvert the signal yp by mixing and filtering it with the filter. Plot the PSD of the received signal.

```
% TODO
%     y = ...
```