ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
School of Computer and Communication Sciences

Handout 1 Modern Digital Communications
Assignment 1 September 19, 2018 (due Sept 25, 2018)

A MATLAB Refresher and AM Signals

In this first assignment, you have to write and upload on moodle several MATLAB files. Take advantage of the reference card that you find on the course webpage[1]. Also, take into account the style guidelines discussed in class.

Unless otherwise instructed, in this assignment you are allowed to use only standard MATLAB functions (as opposed to toolboxes).

EXERCISE 1. Let $s(t)$ be a baseband signal with support in the interval between 0 and `d` [seconds]. Let `s` consist of its samples taken every `1/fs` seconds, with the first sample taken at $t = 0$.

1. Generate the vector `t` such that `plot(t, s)` produces a plot with the correct time scale on the x-axis.

2. Let `s_f` be the DFT of `s`. Generate the vector `f` such that `plot(f, fftshift(abs(s_f)))` produces a plot with the correct frequency scale on the x-axis.

3. Making use of the vectors which you generated above, write a function `tfplot` characterized by the header given below and save it as `tfplot.m`. Keep this file somewhere safe, as it will be used again in future assignments.

```
function tfplot(s, fs, name, plottitle)
% TFPLOT Time and frequency plot
%    TFPLOT(S, FS, NAME, TITLE) displays a figure window with two
%    subplots.  Above, the signal S is plotted in time domain; below,
%    the signal is plotted in frequency domain. NAME is the "name" of the
%    signal, e.g., if NAME is 's', then the labels on the y-axes will be
%    's(t)' and '|s_F(f)|', respectively.  TITLE is the title that will
%    appear above the two plots.
```

Figure 1 shows the output of the function when called like this:

```
tfplot(m, fs, 'm', 'A sinc signal')
```

(where `m` is the sinc signal shown in the figure).

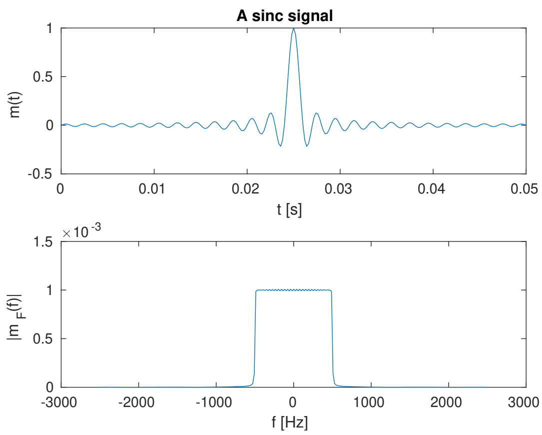

Figure 1: Example output of the function `tfplot`

For testing purposes, on the course webpage we provide a function `sincplot` which produces Figure 1. `sincplot` calls the compiled version of `tfplot`, named `sol_tfplot`. In order to test your implementation, you can just replace `sol_tfplot` with your own version of `tfplot`.

[1] `http://moodle.epfl.ch`

EXERCISE 2.

1. Write a function `s = my_ammod(m, K, A, fc, fs)` that modulates the signal `m` using AM, where `fc` is the carrier frequency f_c, `K` and `A` are constants as defined in class, and `fs` is the sampling frequency. Save the function as `my_ammod.m`.

2. In a new function `test_am()`, create the message signal

$$m(t) = \frac{1}{2}\cos(2\pi f_{\text{info}} t),$$

where $f_{\text{info}} = 10\,\text{Hz}$, $f_c = 300\,\text{Hz}$, $A = K = 1$, and $f_s = 4000\,\text{Hz}$. Let the duration of the signal be $d = 1\,\text{s}$. Use your function `tfplot` to display the message signal as well as the modulated signal. Do the plots correspond to your expectations?

3. Write a function `m = my_amdemod(s, fc, fs)` that demodulates the AM signal `s` and returns the message signal `m`, normalized to have values between -1 and 1. As before, `fc` is the carrier frequency and `fs` is the sampling frequency. Test your function by calling it from within `test_am()`. Plot the result using `tfplot` and verify that the message signal has been correctly reconstructed.

   Hint. To filter the signal, use a second order Butterworth filter[2]. The filtered signal has a transient at the beginning that one can remove. The length of the transient is the length of the filter's impulse response[3].

4. Now modify the parameters of the message signal $m(t)$ to produce a more audible signal, and use `sound` to play both the original and the reconstructed message signal to verify that they are the same. Note that we do not necessarily use the default sample rate of the command `sound` (you should check the help page).

5. Once you feel confident that your code works and that you understand what is going on, download the AM data file from the course webpage. This file contains an AM signal of carrier frequency $f_c = 20\,\text{KHz}$, sampled at $f_s = 100\,\text{KHz}$.

   Write a function `play_am()` that loads the signal from the file, demodulates it using `my_amdemod`, downsamples[4] the result by a factor 10 and plays it using `sound` (again, for this command you should pay attention to the sampling rate).

As for the previous exercise, in order to help you with the implementation, we provide on the course website the compiled versions of the functions you have to write. We provide you as well with a script named `function_mapper`, that you can use to choose between your implementation and the compiled solutions of `my_ammod.m` and `my_amdemod.m`. The script `function_mapper` is called from within `test_am.p` and `play_am.p`. If you wish to use it within your own scripts or functions, you need to include the command `function_mapper;` in your code before calling `hw1_ammod.m` or `hw1_amdemod.m`.

WHAT TO HAND IN:

An archive containing the files `tfplot`, `my_ammod`, `my_amdemod`, `test_am`, and `play_am`.

[2] see the MATLAB help on the functions `butter` and `filter` to learn how to filter the signal.

[3] In order to check your filter's impulse response, you can just pass the sequence $[1\,0\,0\,0\ldots0]$ through your filter. Furthermore, you can use `tfplot` to visualize it in time and frequency domain.

[4] cf. the MATLAB function `downsample`