ELEC1207 Electronic Systems
Communications Coursework

Rob Maunder
rm@ecs.soton.ac.uk

1 Introduction

In this coursework, your task is to design a transmitter that can convert an analogue message signal $x(t)$ into a modulated signal $y(t)$, having an amplitude spectrum $Y(f)$ that adheres to a particular set of constraints. You will also need to design a receiver that can generate a reconstructed message signal $\hat{x}(t)$ that matches $x(t)$ as closely as possible, even when the received modulated signal $\hat{y}(t) = y(t) + n(t)$ is corrupted by the Additive White Gaussian Noise (AWGN) $n(t)$, as shown in Figure 1.

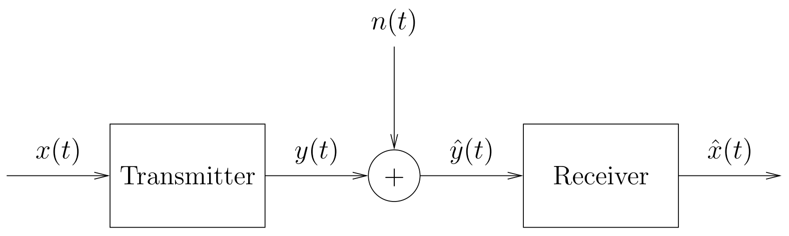

Figure 1: Schematic showing the interaction between the transmitter and receiver.

2 Matlab

You are required to implement your transmitter and receiver as functions in the Matlab files `transmitter.m` and `receiver.m`, respectively. Your design can be tested using the following Matlab files, which can be downloaded from
`https://secure.ecs.soton.ac.uk/notes/elec1207/comms/coursework.zip`

`main.m` - This script can be run to generate $x(t)$, provide it to `transmitter.m`, check that the constraints are met by the amplitude spectrum $Y(f)$ of the resultant $y(t)$, generate $n(t)$ and add it to $y(t)$, provide the resultant $\hat{y}(t)$ to `receiver.m` and finally measure the Signal to Noise Ratio (SNR) of the resultant $\hat{x}(t)$. This script will also plot $x(t)$, $y(t)$, $\hat{y}(t)$ and $\hat{x}(t)$, as well as the corresponding amplitude spectra $X(f)$, $Y(f)$, $\hat{Y}(f)$ and $\hat{X}(f)$.

`transmitter.m` - This file is listed in Figure 2. It includes a function that implements an example transmitter, which employs undermodulated Amplitude Modulation (AM). You need to replace this function with your own `transmitter` function, which must use the line `function [y, y_title] = transmitter(x, t)` to define its inputs and outputs. Here, the vector `y` should have the same dimensions as `x` and `t`, while `y_title` should return a string that describes the operation of the transmitter. You should try to pick something unique for `y_title`, because you will need to use it to identify your work when the anonymised marks are returned.

`receiver.m` - This file is listed in Figure 3. It includes a function that implements an example receiver, which employs a non-coherent demodulator. You should replace this function with your own `receiver` function, which must use the line `function [x_hat, x_hat_title] = receiver(y_hat, t)` to define its inputs and outputs. Here, the vector `x_hat` should have the same dimensions as `y_hat` and `t`, while `x_hat_title` should return a string that describes the operation of the receiver. You should try to pick something unique for `x_hat_title`, because you will need to use it to identify your work when the anonymised marks are returned.

`sample.m`, `desample.m`, `quantise.m`, `dequantise.m`, `pcm_encode.m`, `pcm_decode.m`, `raised_cosine_filter.m`, `root_raised_cosine_filter.m`, `low_pass_filter.m` and `high_pass_filter.m` - These files include Matlab functions that may like to call from within your `transmitter` and `receiver` functions.

`transmitter_BPSK.m` - This file provides an example `transmitter` function, which employs Binary Phase Shift Keying (BPSK) modulation. You may like to copy and paste the contents of this file into `transmitter.m`, so that you can use it as the basis of a digital transmitter.

`receiver_BPSK.m` - This file provides an example `receiver` function, which employs BPSK demodulation. You may like to copy and paste the contents of this file into `receiver.m`, so that you can use it as the basis of a digital receiver.

```matlab
% A transmitter, which converts a message signal into a modulated signal
% x -  a vector of L real values, representing the message signal
% t - a vector of L regularly-spaced time instants
% y - a vector of L real values, representing the modulated signal
% y_title - a string which describes the operation of the transmitter
function [y, y_title] = transmitter(x, t)

% Describe the operation of the transmitter
y_title = 'Undermodulated AM using f_c = 10, V_c = 2.5 and k_{am} = 1';

% Carrier frequency
f_c = 10;

% DC offset
V_c = 2.5;

% Modulation sensitivity
k_am = 1;

% Full AM modulation
y = (V_c+k_am*x).*cos(2*pi*f_c*t);
```

Figure 2: Listing of example `transmitter.m`.

```
% A receiver, which converts a modulated signal into a demodulated signal
% y_hat - a vector of L real values, representing the received modulated signal
% t - a vector of L regularly-spaced time instants
% x_hat -  a vector of L real values, representing the demodulated signal
% x_hat_title - a string which describes the operation of the receiver
function [x_hat, x_hat_title] = receiver(y_hat, t)

% Describe the operation of the receiver
x_hat_title = 'Non-coherent demodulator using half-wave rectifiction and a LPF cutoff frequency of 1.5 Hz';

% Half-wave rectify the received signal
x_hat = max(y_hat,0);

% Choose a cutoff frequency
f_cutoff = 1.5;

% Low pass filter the signal
x_hat = low_pass_filter(x_hat,t,f_cutoff);

% Remove the DC offset
x_hat = x_hat-mean(x_hat);

% Normalise the signal
x_hat = x_hat/sqrt(mean(x_hat.^2));
```

Figure 3: Listing of example `receiver.m`.

3 Example

The example `transmitter` and `receiver` functions of Figures 2 and 3 implement the transmitter and receiver schematics shown in Figure 4. When using these `transmitter` and `receiver` functions, the `main.m` script produces the plots shown in Figures 5 – 8.

A 10 s portion of the analogue message signal $x(t)$ is shown in Figure 5, together with the amplitude spectrum $X(f)$ of the message signal. Note that this signal has a maximum frequency of 1 Hz. Although this signal has been generated randomly, the script `main.m` seeds Matlab's random number generator so that the *same* random signal is generated every time.

Figure 6 shows a 10 s portion of the modulated signal $y(t)$ that is produced by the example `transmitter` function of Figure 2, as well as the corresponding amplitude spectrum $Y(f)$. As described in Section 1, this amplitude spectrum is required to adhere to a particular set of constraints. The first part of these constraints is represented by the red line that is shown in Figure 6, which imposes an upper limit on the amplitude spectrum $Y(f)$. More specifically, the amplitude spectrum $Y(f)$ must not exceed an amplitude of 5 within the band 5 to 15 Hz and it must not exceed an amplitude of 0.3 outside of this band. The second part of the constraints is that all carrier frequencies must be a multiple of 0.1 Hz. Note that the example `transmitter` function of Figure 2 satisfies these constraints, but it does not make very good use of the available bandwidth.

A 10 s portion of the received modulated signal $\hat{y}(t)$ and the corresponding amplitude spectrum $\hat{Y}(f)$ are shown in Figure 7. As described in Section 1, this has been corrupted by the AWGN $n(t)$. Note that $n(t)$ is rather severe and so it has obliterated the modulated signal. Although this AWGN has been generated randomly, the script `main.m` seeds Matlab's random number generator so that the *same* random AWGN is generated every time.

Finally, Figure 8 shows a 10 s portion of the reconstructed message signal $\hat{x}(t)$ that is produced by the example `receiver` function of Figure 3, as well as the corresponding amplitude spectrum $\hat{X}(f)$. Figure 8 also quantifies the SNR of the reconstructed signal $\hat{x}(t)$, which is calculated as

$$\text{SNR [dB]} = 10\log_{10}\frac{E\{[x(t)]^2\}}{E\{[x(t)-\hat{x}(t)]^2\}},$$

where the operator $E\{\cdot\}$ quantifies the average value of the operand. Note that a very low value results for the SNR of the reconstructed message signal $\hat{x}(t)$ that is produced by the example `receiver` function

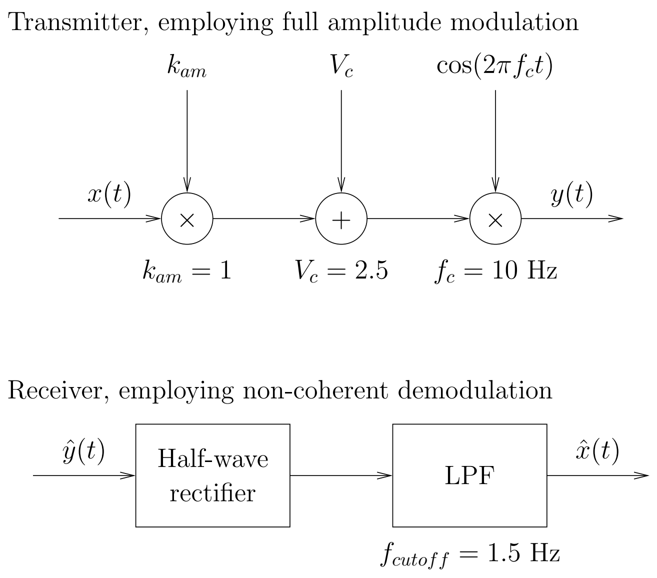

Figure 4: Schematic showing the operation of the transmitter and receiver.

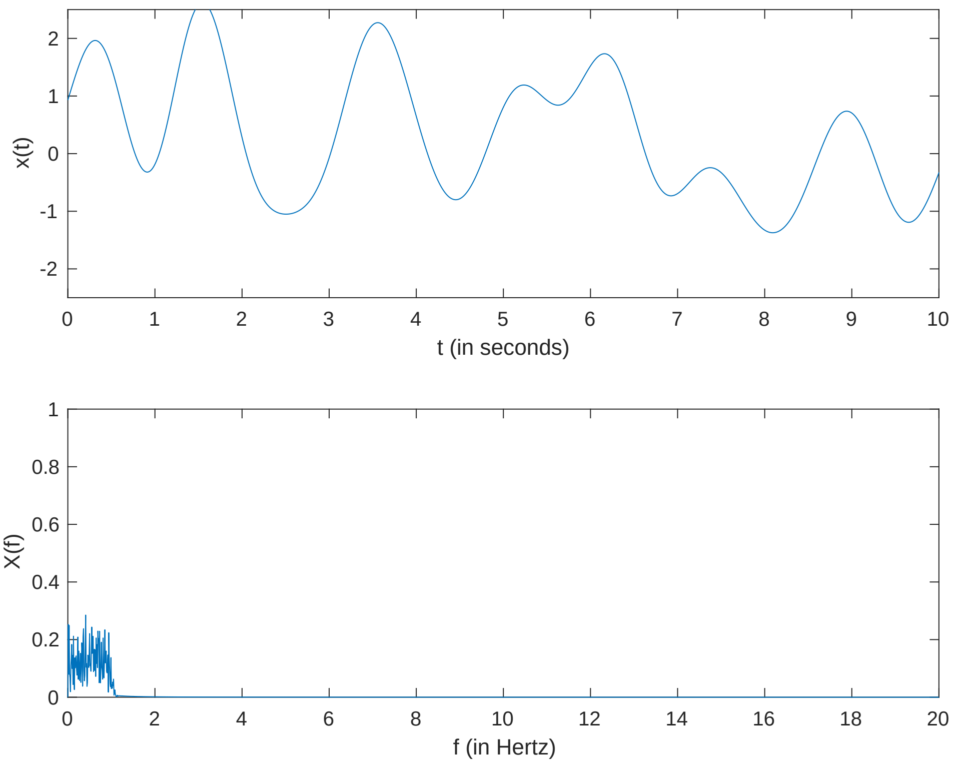

Figure 5: Plot of a 10 s portion of the analogue message signal $x(t)$, as well as the corresponding amplitude spectrum $X(f)$.

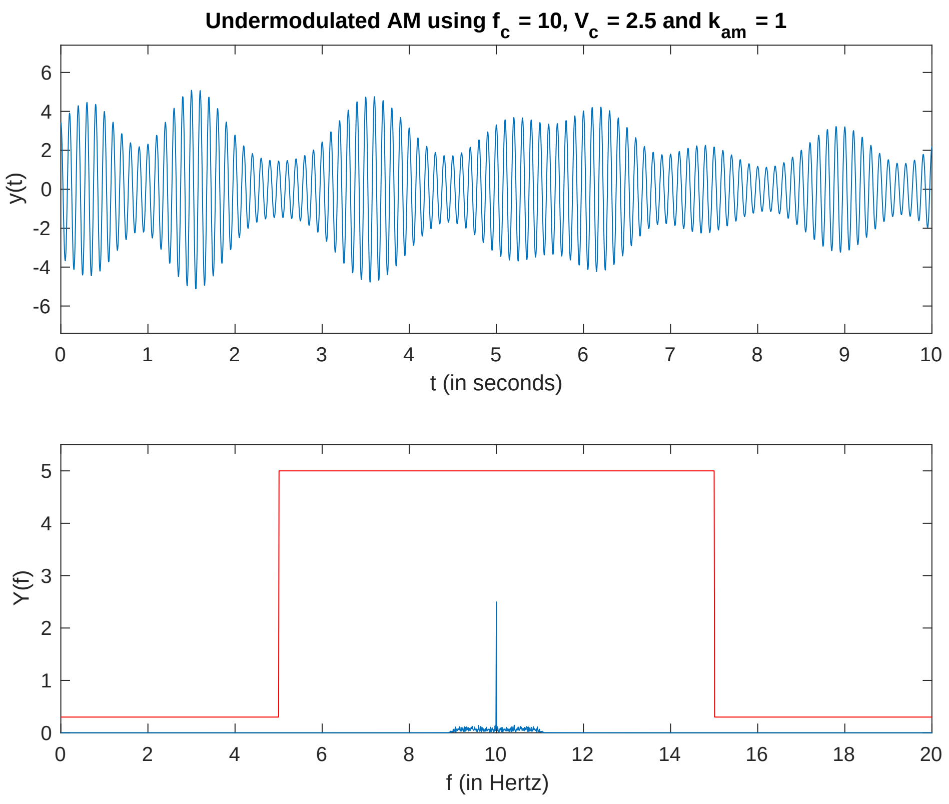

Figure 6: Plot of a 10 s portion of the modulated signal $y(t)$, as well as the corresponding amplitude spectrum $Y(f)$.

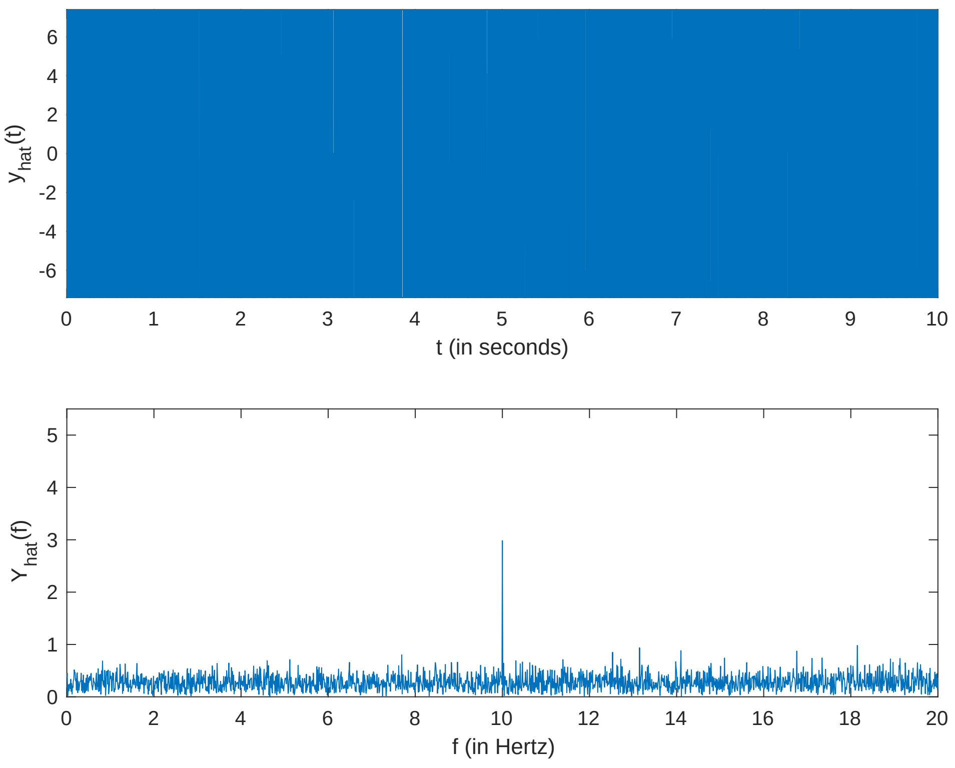

Figure 7: Plot of a 10 s portion of the received modulated signal $\hat{y}(t)$, as well as the corresponding amplitude spectrum $\hat{Y}(f)$.

of Figure 3. This is because the example `receiver` function of Figure 3 is unable to mitigate the severe AWGN that is present in $\hat{y}(t)$ and so $\hat{x}(t)$ has very little resemblance to $x(t)$. Your task is to write `transmitter` and `receiver` functions that produce a much higher SNR than that shown in Figure 8.

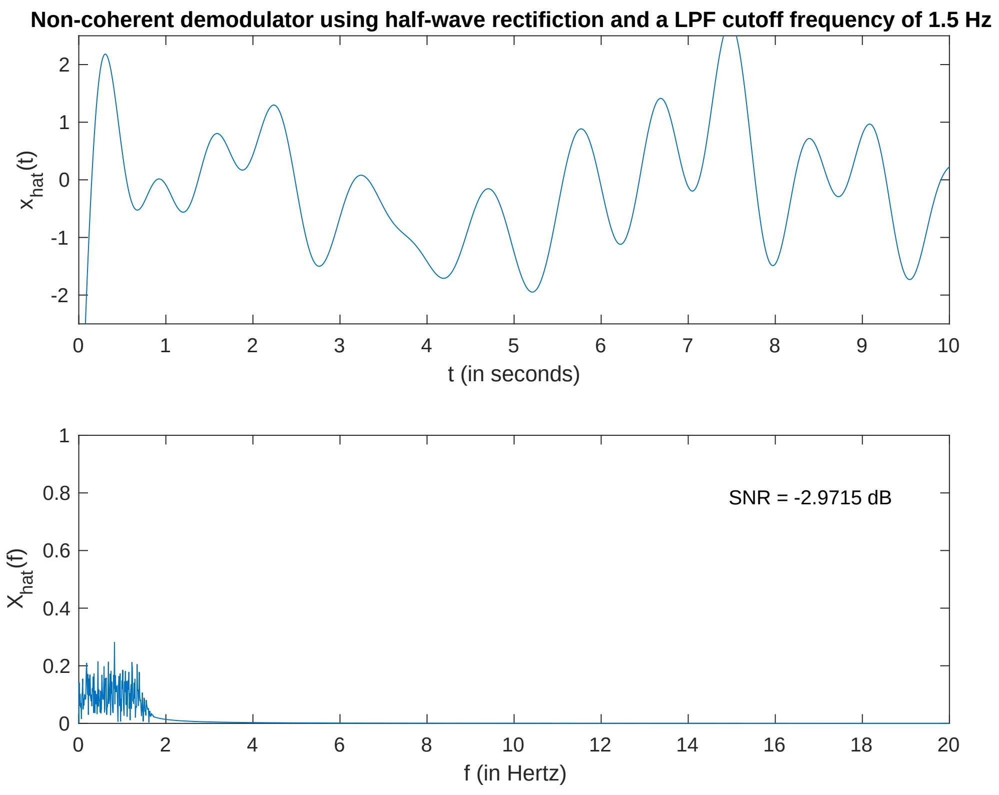

Figure 8: Plot of a 10 s portion of the reconstructed message signal $\hat{x}(t)$, as well as the corresponding amplitude spectrum $\hat{X}(f)$.

4 Submission

The process for submitting your work is as follows:

1. Southampton students should go to `https://handin.ecs.soton.ac.uk/handin/2021/ELEC1207/1/`, while USMC students should go to `https://handin.ecs.soton.ac.uk/usmc-handin/2021/ELEC1207/1/`.
2. There, you should submit the following five things before 4pm on Friday 7th May 2021:
   - Your `transmitter.m` file, containing your `transmitter` function. This must be a .m file, not a .pdf, .doc, .zip or .anythingelse file. I will run your .m file with a fresh copy of `main.m` extracted from `coursework.zip`, so you should make sure that this will work for me.
   - Your `receiver.m` file, containing your `receiver` function. This must be a .m file, not a .pdf, .doc, .zip or .anythingelse file. I will run your .m file with a fresh copy of `main.m` extracted from `coursework.zip`, so you should make sure that this will work for me.
   - A PDF file containing the plot of $y(t)$ and $Y(f)$ that is produced when the script `main.m` calls your `transmitter` function. This plot should include the red line, as in Figure 6.

- A PDF file containing the plot of $\hat{x}(t)$ and $\hat{X}(f)$ that is produced when the script `main.m` calls your `receiver` function. This plot should include the SNR, as in Figure 8.
- A PDF file containing a schematic of your transmitter and receiver design. This should contain a similar level of detail as is provided in Figure 4, so that somebody else could independently reproduce your `transmitter` and `receiver` functions.

5 Tips

Here are some tips that will help you get a high mark in this coursework:

- This coursework contributes 2.5% of your mark for the 15 credit module ELEC1207 Electronic Systems. You should therefore aim to spend about 3.75 hours on it. Please do not spend any more time than this unless you are enjoying yourself or you feel that it is helping your revision for the course.
- I would like to encourage you to *verbally* discuss your ideas with your classmates, since I believe that this can enhance your learning. However, you should not show each other your Matlab code, schematics or plots, since this counts as a breach of academic integrity.
- The mark that you receive for this coursework will be directly related to the SNR of your reconstructed message signal.
- You may like to think of a simple way that an analogue modulation scheme can make good use of the available bandwidth.
- It is possible to implement a digital `transmitter` function by successively calling the `sample`, `quantise`, `pcm_encode`, `desample` and `raised_cosine_filter` functions, as exemplified in `transmitter_BPSK.m`. Likewise, a corresponding digital `receiver` function can be implemented by successively calling the `low_pass_filter`, `sample`, `pcm_decode`, `dequantise`, `desample` and `low_pass_filter` functions, as exemplified in `receiver_BPSK.m`.
- I have tried to design this coursework so that the highest SNRs will be achieved by the students that invest the most effort. In particular, the students that achieve the highest SNRs are likely to be those that implement a variety of different solutions (both analogue and digital) and then submit the best one. My model solution achieves an SNR of 30.67 dB. There was once a student who achieved 54.57 dB! Even higher SNRs may be possible...
- I will run `main.m` on my own computer and get it to call your `transmitter.m` and `receiver.m` files. So, you should make sure that you submit Matlab files that will work first time for me! You must submit exactly your `transmitter.m` and `receiver.m` files. Do not submit .zip, .doc, .pdf or .anythingelse files (or rename them to .m files), since these will not open in Matlab. You are in danger of receiving a mark of 0 if I cannot get your .m files to run on my computer.
- I will manually check that your carrier frequencies are multiples of 0.1 Hz.
- At the end of the semester, I will show the submitted solutions having the highest SNRs to the rest of the class.