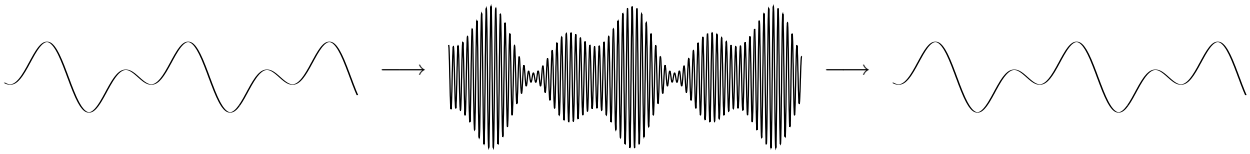
Experiment # 3

Amplitude Modulation and Demodulation



1 Purpose

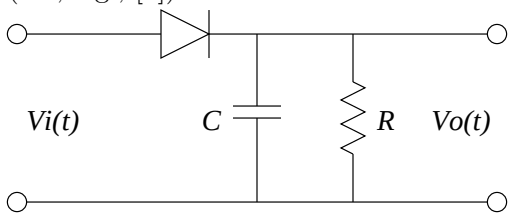
Amplitude modulation (AM), still widely used in commercial radio today, is one of the simplest ways that a message signal can modulate a sinusoidal carrier wave. The purpose of this lab is for you to gain familiarity with the concepts of amplitude modulation and demodulation. This will be done in three main steps:

- First, an amplitude modulation system will be created and simulated using `Simulink`.
- Second, this system will be implemented on a TMS320C6711 DSP platform. Similarly to an AM radio station, the modulating signal will be an audio signal.
- Third, the AM signal will be demodulated by an envelope detector and low-pass filter combination, which you will implement with discrete analog components on a prototype-board.

At each of these three steps, you will be required to change various system parameters and observe the consequences of the changes.

2 Background Reading and Preparation

Amplitude modulation is one type of continuous-wave modulation, covered in [1]. *Before* coming to the lab, you are encouraged to read [1], and also the principles of operation of the diode envelope detector circuit shown below (see, e.g., [2]).



A diode envelope detector

Before coming to the lab, you should be able to:

1. write an equation for an AM-modulated signal, and understand the significance of all constants in the formula;
2. understand the concept of overmodulation;
3. understand how to choose the values of R and C in the envelope detector to achieve an effective AM demodulator;
4. understand how the AM-modulated signal is described in the frequency domain.

In order to perform this experiment effectively, a good understanding of `Simulink` and `Matlab` is required.

3 Equipment

Hardware:

1. One Signal Generator
2. One Two-Channel Oscilloscope
3. One Spectrum Analyzer
4. One TMS320C6711DSK board attached to a workstation
5. Coaxial cables, BNC-to-BNC

Software:

1. `Matlab` Release 13
2. `Simulink` with ECE416 Toolbox
3. `Code Composer Studio`, v.2.1

4 Experiment

The experiment is divided in three parts, one of which being optional: the design and simulation of an AM modulator in `Simulink`; he download and test of the designed system on a DSP platfom; and, time allowing, the demodulation of the signal with analog, discrete, components.

All results that will be marked are to be reported in the proper spaces provided in this outline.

4.1 Designing and Simulating an AM Modulator

Based on the block diagram that you have reviewed for an AM modulator, build your model using the blocks found on the ECE416 blockset. For the initial simulation, use a `DSP Sine` generator as input, and for output use one time scope and one FFT-based scope. Do not forget to use an input buffer compatible with the length of the FFT you choose. Use an input signal of 1KHz, 0.5 peak Voltage (1Vpp).

As your block diagram should indicate, the input signal must be added to a DC component and this combined signal will be multiplied by the carrier frequency. Use a DC component of 1. The carrier frequency is generated by another discrete-time sine wave generator, this time generating a much higher frequency (use 12KHz – this number is compatible with the sampling frequency used by the DSP board).

Run your simulation, and observe the result on the `Simulink` scopes. Adjust the setting of your scopes appropriately. You may be required to resolve some issues regarding signal shape and compatibility between `Simulink` blocks. You should observe an AM signal similar to the one you drafted on your lab preparation sheet.

- *In the space below, sketch the output signal observed. The T.A. might ask you if the values make sense, and why*

- *Now modify your system to obtain a modulating index greater than 1 (i.e., greater than 100%), and sketch the resulting signal in the space below. Explain how you have obtained that result and what are the consequences of such modulation, based on the theory.*

- *Vary the input signal to a square wave($1V_{pp}$), and report the results below, in time and frequency domain. Report and explain the voltage values in the time domain and the frequency components in the frequency domain.*

4.2 Building and Running Your Model

As you have done in previous experiments, insert your modulator on one of the channels out of the model provided in the `template` directory. This is to say that one of the outputs of the `Channel Separator` block should be modulated and one whould connect straight trough to the `Channel Combiner`. Your signal generator will provide the same signal (a sine wave, initially) to the two inputs on the board. Connect both outputs to the oscilloscope.

Set the parameters on the blocks of your modulator (DC, Carrier sine wave generator) to the original values of the previous section. If you recall from the first experiment, each of the outputs of the `Channel Separator` will handle a type double data, which resulting from casting an integer word of 32 bits. In order to add your DC level to this signal, the value output by the `DC Constant` block must be given a significant gain. That number will be provide to you. As for the signal from the generator, use as a standard input signal a 1Vpp, 1KHz sine wave.

Follow these steps in order to get your system working:

- Build your model from the `Simulink` build. You can achieve this by using the button or pressing Ctrl-B. The result of this action will be a project exported to `Code Composer Studio`.
- In `Code Composer Studio`, build your project. This means to compile the programs generated and creating an executable program to run on the board. You can do this by clicking on the "build all" button or going under project/build all. You will see the files being compiled, linked and assembled in the dialogue window.
- Now that an executable has been created, you must load it onto the target hardware. Go under file/load program. Select the program you just built as the one to be loaded. You will see a progress bar indicating that your file is being loaded into memory.
- Run your program. You can do this by pressing on the "running-man" button on the left-hand side of your screen, by pressing F5 or by going under project/run. If your board is connected and powered and if your signal generator is set to provide you with an output

(these are common mistakes), you should see your AM modulated signal on the scope. It can be the case that you may need to load the file once again for it to work. If everything looks fine but it does not work, try loading it again and running it. If the problem persists, call the TA or the Lab Engineer.

Answer the questions below, pertaining to the system you are running.

1. **Time Domain Results**

   - *Sketch the time domain results. Yes, these should be similar to the first ones you got in your simulation. As you sketch your output, report the values that you believe are relevant for someone who reads it to understand what you have done.*

   - *If you change your DC value to zero, what is the consequence? (you can try changing it directly in the code, and then recompiling your project in CCS). Draw the signal in time domain and explain.*

2. **Frequency Domain Results**

   In the simulation-only part of this experiment, you included an FFT-based scope provided by `Simulink` to see the frequency domain representation of the output signal. Since now you have a system generating actual signals, you have two options to visualize the output signal in the frequency domain: one is to attach a spectrum analyzer to the output of the target board; the second is to use `Matlab` to retrieve the data from the memory into the workspace. Feel free to use either method to respond to the questions below. It is likely that `Matlab` will give you a better view of it. The code you need is found below; you may also find it in the `template` directory in `Matlab6p5/work`.

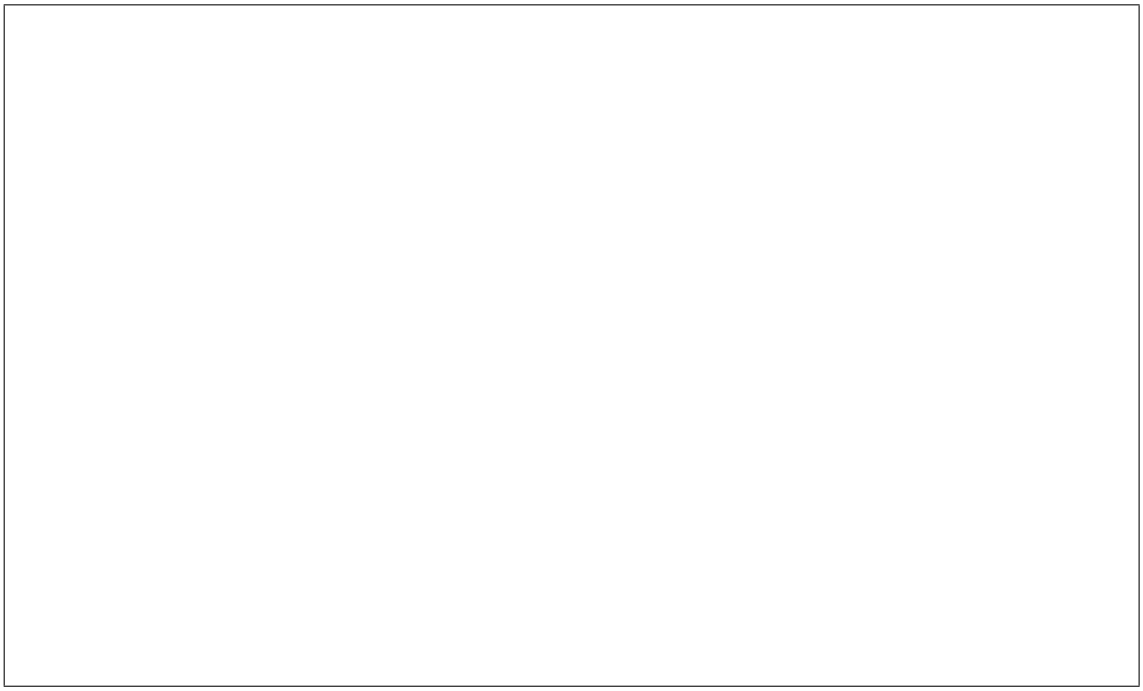
- Create a *".m"* file with the following contents:

```
cc=CCS_Obj;
x=read(cc,address(cc,'channel_a'),'double',1024);  % channel A
xx=read(cc,address(cc,'channel_b'),'double',1024);  % channel B
z=fft(x,1024);
zz=fft(xx,1024);
figure
subplot(2,1,1), plot(abs(z)), axis([0 512 0 300])  % using 1Vpp on signal generator
subplot(2,1,2), plot(abs(zz)),axis([0 512 0 300])
figure
subplot(2,1,1), plot(x), axis([0 150 -3.5 3.5])
subplot(2,1,2), plot(xx), axis([0 150 -3.5 3.5])
```

- Run the file
- Modify the file above and run again as you wish, to visualize the data in the frequency domain.

Answer the questions below pertaining to the frequency domain results.

- *Choose either to run the program presented above or to connect your output to the spectrum analyzer. In the space below, sketch the frequency domain representation of the output signal (AM signal). Clearly indicate the relevant values.*



- *Change the input signal level until you achieve overmodulation, or simply reduce the DC value in your program, recompile and run it. Sketch below the frequency domain*

representation of the resulting signal and explain what are the consequences of this over-modulation.

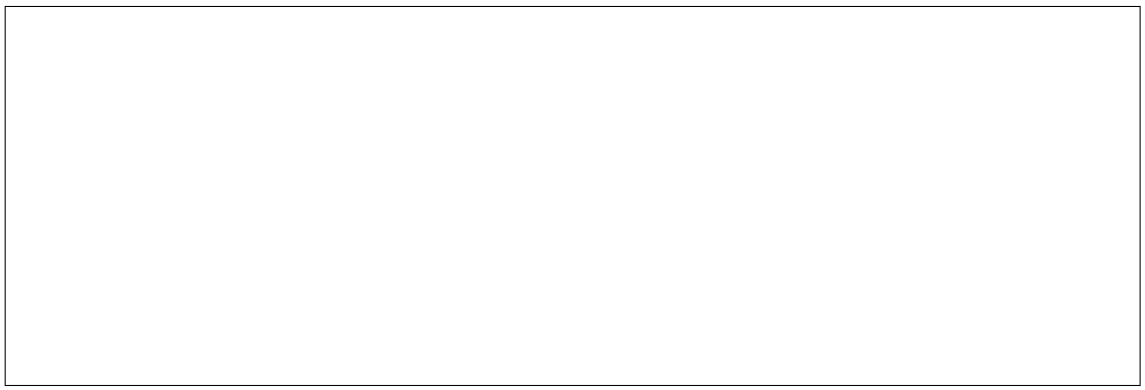
- *Change the input signal to a $1V_{pp}$, 1KHz square wave. Draw and explain what do you see as your output signal.*

4.3 If Time Allows: Demodulating the AM Signal

Your modulator runs fine, but your intention is to transmit the desired signal and receive it after it goes through modulator, amplifiers, antenna, channel and receiving antenna. You must demodulate the received signal to extract the **modulating** signal, which is the signal containing the information (in your case, this information is a 1KHz, 1Vpp sine wave, or tone).

You are now to build and test a simple AM demodulator which is made of the envelope detector you have designed in your preparation sheet. The calculation of the components should have been done prior to the experiment. Your task is to transmit the output signal of the DSP target onto the input of the envelope detector you have assembled on a prototype board, and the output of your envelope detector to the oscilloscope.

- *Sketch below the resulting waveform at the output of the envelope detector. Did it recover the modulating signal? Provide measurements.*



5 Going The Extra Mile

Another extra adventure you may want to try is to modify the C code of the generated project and make your own project. This will help you to understand many details of writing code for real-time applications. At this point you know that you do not need to go all the way back to the model every time you need to modify a parameter. You may consider thinking how to modulate a signal in amplitude using a C program. Give some thought to generating a tone (your carrier), and modulating that tone with samples of the incoming signal. How do you add DC to your incoming samples? What are the common methods to generate a tone on the DSP? If you want to put that to practice, come by the lab and try out your own AM modulator.

6 Conclusion

In this sample experiment, you were presented with the key issues involved in designing, simulating and implementing an AM modulator and demodulator. This was done by using a simulation model, a DSP platform in which the model was implemented and an envelope detector for the demodulation of the signal. The experiment intended to guide you through the steps necessary to achieve a practical understanding of the concepts studied not only in the theory, but also in other courses of the curriculum.

References

[1] S. Haykin *Communication Systems*, 4th Edition. Toronto: John Wiley & Sons, Inc., 2001.

[2] B. P. Lathi, *Modern Digital and Analog Communication Systems*, 3rd Edition. New York: Oxford University Press, 1998.

[3] The Mathworks, Inc., *Using Simulink*.

[4] The Mathworks, Inc., *Real-Time Workshop*.

- *Name:*
- *Student No.:*
- *Name:*
- *Student No.:*
- *Session:*