Chapter 6

Matlab ADSL Signal Models and Transform Verification

The work described in this chapter arose for three main reasons:

- Matlab ADSL signal models could be used to examine the effect of different sets of frequency multiplication and addition vectors when developing the simulator's software drivers.
- To enable basic verification of the transform and sampling scheme for DMT ADSL signals.
- To satisfy the author's 'bit pushing' inclinations and intrigue at the physical nature of DMT QAM based modulation.

Seven Matlab functions have been written and the associated m-files included on the CD ROM, which enable multilevel QAM signals to be visualised, sampled and then filtered to recover pseudo analogue waveforms. Online help is included with all functions and can be accessed with the usual Matlab `help` keyword.

6.1 QAM Signal Generation

As described more fully in chapter 3, QAM fundamentally consists of the sum of sinusoid and cosinusoidal carriers of the same frequency whose amplitudes are discrete functions of the data words

to be transmitted. Data is encoded onto each carrier pair by mapping an m-bit binary word onto a unique pair of carrier amplitudes, forming a symbol. To enable a set of different words to be transmitted, the amplitudes of the two carriers are changed from the old to new values during the time between sampling epochs. The nature of the transitions determine the transmission bandwidth required. An instantaneous change from one set of amplitudes to another would require an infinite bandwidth, whereas if the transition is in the form of a raised cosine or Nyquist signalling function, the bandwidth required is equal to the data symbol rate. For example, with ANSI DMT ADSL, the symbol rate is 4.3125 kHz and each carrier group occupies a 4.3125 kHz bandwidth. Conceptually, DMT is simply the sum of a set of different orthogonal carrier pairs, each modulated according to its own data input.

6.1.1 Baseband QAM

The Matlab function `genbbnew` generates a time vector describing the baseband signalling envelope of a user defined series of data symbols (i.e. a time vector describing the amplitudes and transitions between of a set of carrier amplitudes at the sampling epochs). To illustrate this, consider the following two vectors which contain the cosine and sine carrier amplitudes at a contiguous group of sampling epochs:

```
cossym = [+1 +3 +3 –1 –3  +1 +1 –1]
sinsym = [+1 –1 –3 –1 –1  +3 –1 –3]
```

These two vectors contain the amplitudes of the two carriers only at the time sampling epochs and are illustrated in figure 6.1

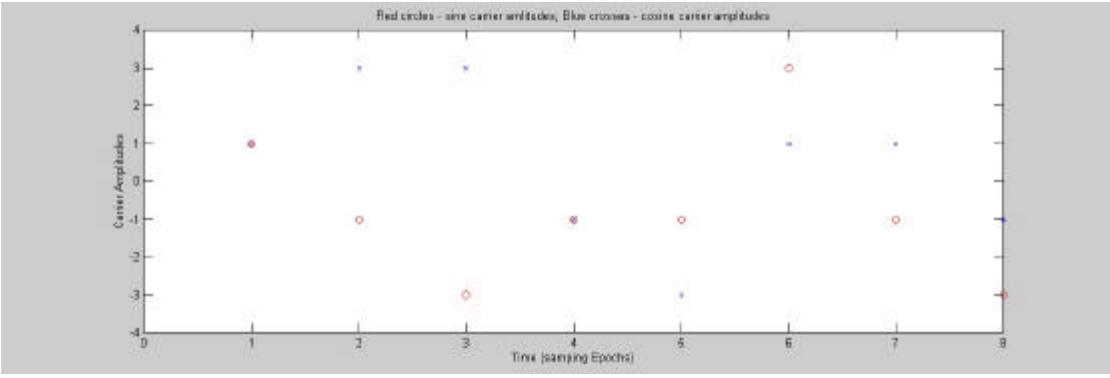

Figure 6.1 Illustrative cosine and sine carrier amplitudes for QAM

For ANSI DMT ADSL, the transitions between the points shown are of the form a raised cosine. The Matlab function

```
[tout, ycosbb, ysinbb] = genbbnew(cossym, sinsym, M, ts, tr);
```

returns two pseudo continuous time vectors `ycosbb` and `ysinbb` with points spaced `tr` seconds apart according to the two sets of carrier amplitudes `cossym` and `sinsym` using the raised cosine function to give the intermediate transition points. The top plot of figure 6.2 shows the baseband envelopes produced when `ts=1`, `tr=0.01`, `M=4` and the previous symbol vectors.

6.1.2 Passband QAM

The two carrier envelopes modulate the cosine and sine carriers by multiplication to produce the two orthogonal passband signals. The final QAM waveform is given by the addition of the two modulated carriers. The function

```
[tout, ybp, ycosbp, ycosbb, ysinbb, cossymb, sinsymb,]
            = qamdef(cossym, sinsym, M, ts, carrierfreq, tr);
```

generates the baseband envelopes using `genbbnew`, then multiplies with the two carriers of a user defined frequency, to produce the two orthogonal signals `ycosbp` and `ysinbp` and finally the QAM signal `ybp`.

The vectors returned using the same parameters as before and a carrier frequency at four times the symbol rate were used to plot figure 6.2 below.

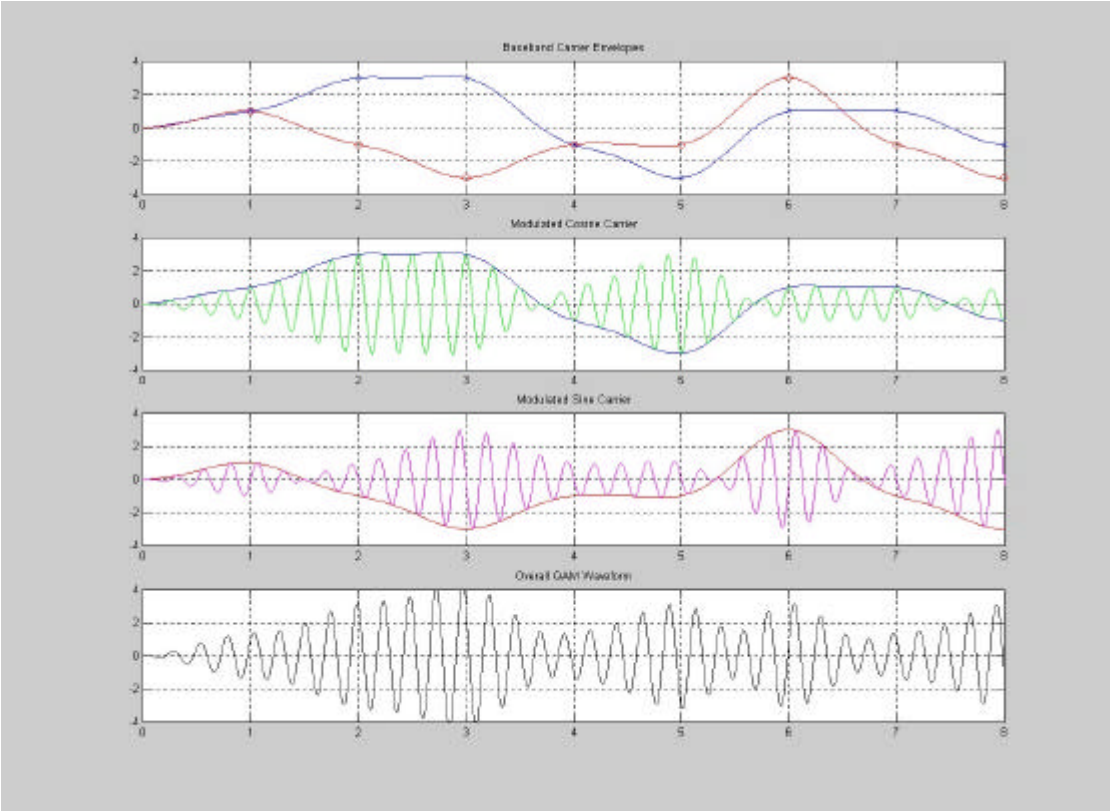

Figure 6.2 QAM signals produced by the Matlab function `qamdef`

It should be noted that although the piecewise approach used to give baseband envelopes, carriers and then passband signals by multiplication is perfectly valid, in practical DMT transmitters the multiple QAM signals are actually produced digitally by means of an inverse DFT, described more fully in section 3.1.2 and associated references.

6.1.3 Extension to DMT

Functions to produce a composite DMT signal, which is simply the sum of multiple individual QAM waveforms with different carrier frequencies, can easily be written through repeated execution of the `qamdef` function and a final vector summation.

6.2 Sampling

The functions above produce pseudo continuous descriptions of QAM signals. The term pseudo continuous is used to indicate that although no digital representation of a signal can be truly continuous (i.e. have an infinite number of points), the number of actual points is far greater than the number of points which would produced by the simulator's sampling of the real ADSL signal. For example, if the simulator sampled at four times the highest carrier frequency, the simulator's internal time data vector representing that pair of carriers would have just four points per carrier cycle. The functions `sampstep` and `sampfreq` both replicate the action of the simulator's ADC time sampling. The first allows the user to define the sampling step in seconds, the second as a sampling rate. Figure 6.3 shows plots of the vectors produced by sampling the final QAM signal of figure 6.2 with a sampling frequency of four times the carrier frequency (illustrated only over the first 3 symbols for clarity). It should be borne in mind when viewing the plot that the QAM signal is a composite addition of modulated cosine and sine carriers, therefore the number and position of samples within subsequent complete oscillations seems to vary. This is deceptive, as it is infact the QAM waveform which is varying compared to a simple sinusoid due to its very nature of modulation.

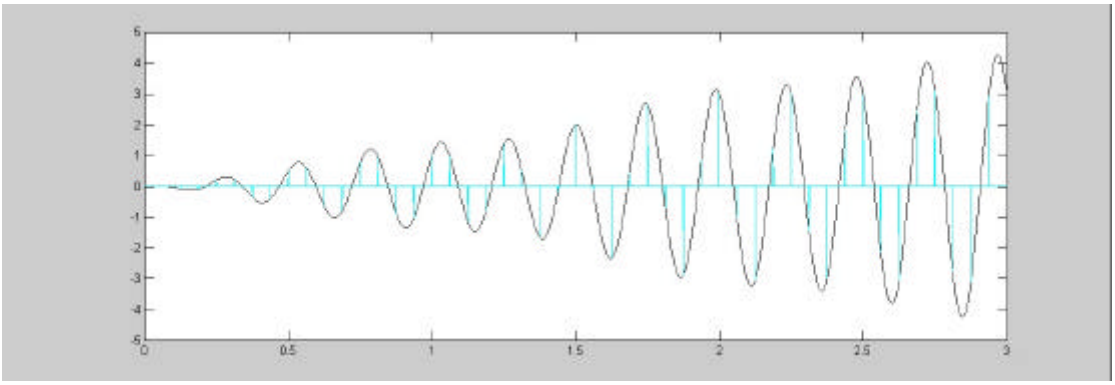

Figure 6.3 Sampled QAM signal produced by the Matlab function `sampfreq`

6.3 Filtering

The samples from the IDFT are converted to an analogue signal in the simulator by the action of the DAC and output low pass filter. Using Matlab, filtering of the sampled discrete time vector of the form shown in figure 6.3 can be achieved via convolution with a pseudo continuous filter impulse response. The function `brckfilt` performs the function of a perfect brickwall filter by convolved with the sampled time vector resulting from the IDFT. The function allows the user to enter the bandwidth of the filter.

6.4 Basic Transform Verification

Using the custom and standard signal processing Matlab functions, it is possible to generate signals for each of the 256 ADSL DMT subtones, sample, perform a DFT then IDFT and finally filter and compare with the original pseudo analogue signal. Physically, this is the same as A/D conversion, discrete Fourier transformation, multiplication by unity coefficients and addition with zero components in the frequency manipulation block, inverse discrete Fourier transformation and finally lowpass filtering on the hardware simulator board.

Various different sets of symbols, symbol periods, carrier frequencies and sampling rates were tested as described above. Comparison of all the resulting waveforms showed that the original psuedo continuous signal was perfectly recovered after sampling, transformation and filtering for sampling rates greater or equal to twice the carrier frequency. In addition when sampling was carried out below the Nyquist limit, the recovered and original waveforms were distinctly different due to aliasing. Mathematically this is not difficult, although somewhat laborious, to prove as follows:

1. Form a continuous equation in the time domain describing the QAM waveform using raised cosine signalling.
2. Sample this waveform by multiplication with a repetitive time impulse function with non-zero impulses at the desired sampling rate over a user defined time sampling window.
3. Perform a DFT
4. Perform an IDFT, which will give the same data vector as the original time samples because the DFT and IDFT are inverse operations with both being one to one functions.
5. Filter the IDFT results through repeated convolution in the time domain of all IDFT output samples with the filter's continuous impulse response.

This proof is by no means new, but the procedure of verification with Matlab is comforting in that the processing employed doesn't fundamentally alter the input signal and therefore in the perfect case introduces no changes to the signal during processing (zero quantisation noise, constrained periodic input signal, brickwall filter response).

6.5 Extension to ‘Simulator’ Simulation

One of the main motivations in writing the custom m-files was to produce a set of functions which would allow the entire signal processing operation from input to output to be modelled in Matlab to aid in the development of the future software driver. Such a model can be used from the command line or a GUI such as Simulink employed using modified functions as block operations. Such as model would enable the effect of different frequency and time manipulation vectors, sampling rates, quantisation precision and filtering schemes to be studied before a hardware board is finally constructed.