

Chapter Exercises

● [Home](#)

ROBERT W. STEWART

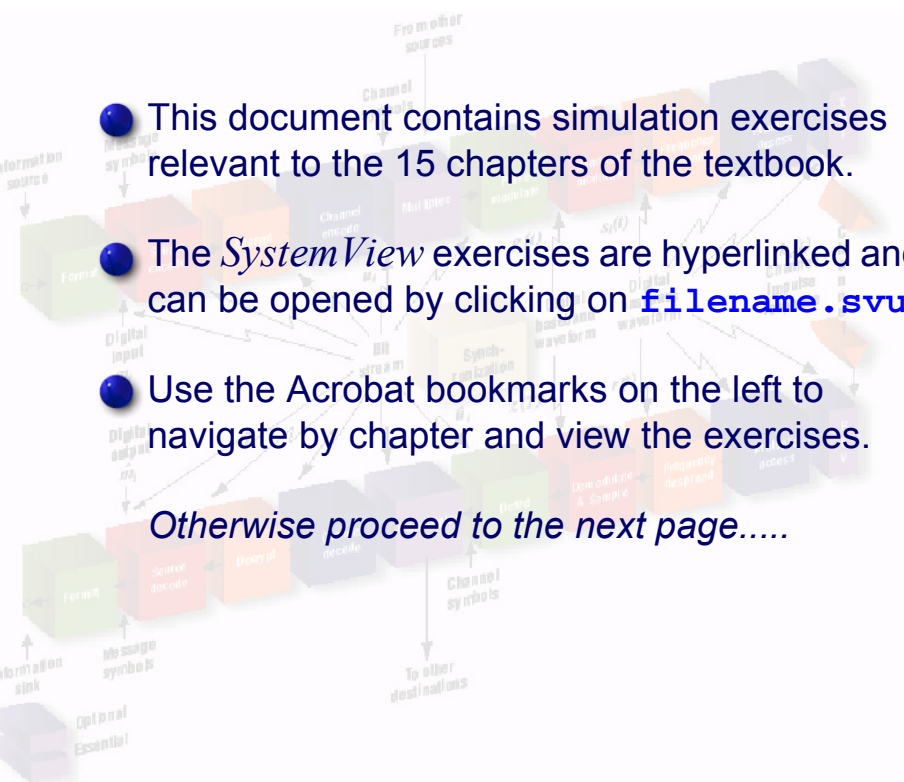
MAURICE L. SCHIFF

Digital Communications

BERNARD SKLAR

2nd Edition

the *Companion* **CD**

- 
- This document contains simulation exercises relevant to the 15 chapters of the textbook.
 - The *SystemView* exercises are hyperlinked and can be opened by clicking on [filename.svu](#)
 - Use the Acrobat bookmarks on the left to navigate by chapter and view the exercises.

Otherwise proceed to the next page.....

● [Getting Started with *SystemView*](#)

● [Exercises: Book Chapters 1-15](#)

● [Concise DSP Tutorial](#)

● [Readme and Help](#)

● [Acknowledgments](#)



*Printable Version
of this document*

● [Simulations powered by](#)



● [Tutorial by](#)

DSPedia[®]
www.dspedia.com




1 Signals and Spectra


Exercise 1.1 Means and Variances of Random Signals




Open the system:

Digital Comm\ch_01\mean_value.svu

This system generates three different signals: (i) Gaussian (or normally) distributed noise; (ii) Uniformly distributed noise; and (iii) Random binary sequence.

(a) Run the system. Go to the  window and note we generate a SQUARED version of all signals.


View all six windows simultaneously by tiling using the toolbar button .

Using the  window “STATISTICS” toolbar button,  we can extract information to complete the table below. (In the “Design” columns enter the theoretical values calculable from the signal source parameters that were set in the design space, and in the “Analysis” columns enter the time average values you calculated in  windows.)

Signal Type	Gaussian		Uniform		Binary	
	Design	Analysis	Design	Analysis	Design	Analysis
Mean Value						
Mean Squared Value						
Standard Deviation						

For these zero mean signals the variance and mean squared values should be the

same (Recall variance is the square of the standard deviation).


- (b) Return to the design space,  and change the mean of (i) the Gaussian noise to 2, (ii) the range of the uniform noise from $2 \rightarrow 4$, and (iii) the offset of the random binary signal to 2. Rerun and complete the table below:

Signal Type	Gaussian		Uniform		Binary	
	Design	Analysis	Design	Analysis	Design	Analysis
Mean Value						
Mean Squared Value						
Standard Deviation						

Exercise 1.2 Generating Probability Density Functions

Open the system:

`Digital Comm\ch_01\pdfs.svu`

- (a) Note that the number samples in the simulation is 1000, and that a uniform and Gaussian noise sources are used. Run the system and in the  **SystemView Analysis** window observe the time form of the waveforms which should appear similar to [Figure 1.1](#):

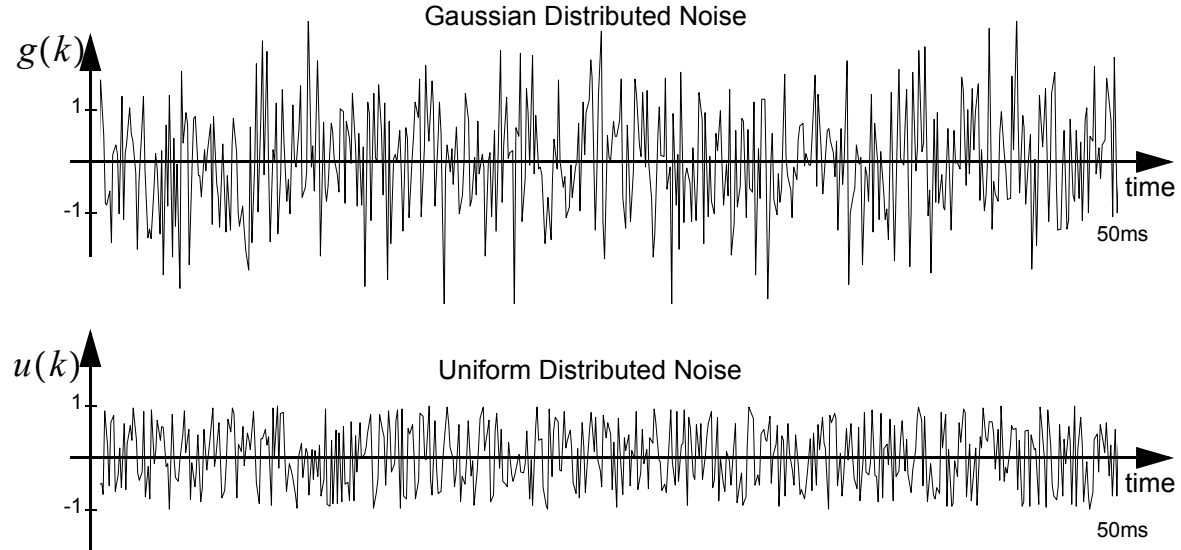


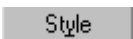



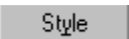





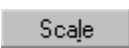



Figure 1.1: *Gaussian and uniform noise signals.*


- (b) In the  window, 40 “bin” histograms have been generated (i.e. pdf approximations) for the uniform and the Gaussian noise tokens. View these histograms and confirm the respective Gaussian and uniform “shapes”. Note that if you add all of the bin values of the histograms the sum will be 1000, i.e. the total number of samples in the simulation.
(The histogram function is located in the CALCULATOR   button then  button.)
- (c) Increase the no. of samples from 1000 to 10000 and run the system. Note when you view the results in the  window the “shape” of the histogram is more defined given the longer averaging being performed.
- (d) In the  window, generate a new histogram with a reduced bin width by increasing the total number of bins from the current value of 40 to, say, 100. (Go to  button in the CALCULATOR  to find the  button) Note the “shape” of the histogram which should be “smoother”.
- (e) The histograms can be scaled in the  window in order to produce an approximation to the pdf. Given there were 10000 samples in the previous simulation, then the y-axis of the histograms can simply be scaled by:

$$\frac{1}{\text{No. of Samples} \times \text{Bin Width}}$$

The number of samples was 10000, and you can calculate the bin width from the histogram plot using the  facility (probably around 0.05 for the uniform noise).

You can then scale by using   then  and entering the

appropriate value to scale the y-axis. Note that SystemView will calculate $1/AB$ inline if you enter $1/\text{"Number"}$.



- (f) Increase the number of samples to 100000 and produce a pdf approximation by scaling the histogram appropriately.
- (g) Confirm the area under the graph is 1 by using the integrate function from ,

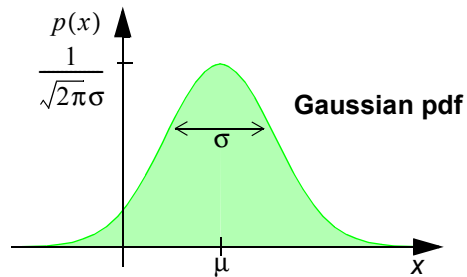
 ,  .

Exercise 1.3 Varying the Mean and Variance of Gaussian Noise Sources

Open the system:

Digital Comm\ch_01\pdfs.svu

Run the simulation and then rerun after changing the mean and standard deviation values. Note the effect this has on the pdf (see also Figure 1.2). Note that using the  window statistics button, , summary this will calculate mean and standard deviation information from time waveforms.

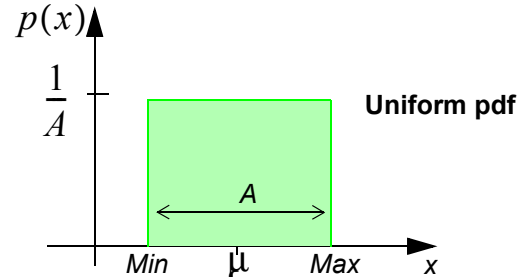


$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\text{Mean, } E[x] = \mu$$

$$\text{Variance, } E[(x-\mu)^2] = \sigma^2$$

$$\text{Standard Deviation} = \sigma$$



$$p(x) = \begin{cases} 0, & x < (2\mu - A)/2 \\ 1/A, & |x - \mu| \leq A/2 \\ 0, & x > (2\mu + A)/2 \end{cases}$$

$$\text{Mean, } E[x] = \mu = (\text{Max} + \text{Min})/2$$

$$\text{Variance, } E[(x-\mu)^2] = A^2/12$$

$$\text{Standard Deviation} = A/(2\sqrt{3})$$

Figure 1.2: General Gaussian and uniform distributed probability density functions.

Exercise 1.4 Gaussian Random Variables

Open the system:

`Digital Comm\ch_01\gaussian_variables.svu`

The probability density function (pdf) $p(z)$ of a Gaussian random variable (GRV) z shown in Eq. 1.1, is the most commonly used pdf in communication theory.

$$p(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{z-a}{\sigma}\right)^2} \quad (1.1)$$

Show that for a Gaussian pdf the mean and variance is given by

$$\text{mean: } E(z) = \int_{-\infty}^{\infty} zp(z)dz = a \quad (1.2)$$

$$\text{variance: } E[(z - \bar{z})^2] = \int_{-\infty}^{\infty} (x - \bar{z})^2 p(x)dx = \sigma^2 \quad (1.3)$$

We can show that if x and y are statistically independent GRVs, then $z = \alpha x + \beta y$ is a GRV and

$$E(z) = \alpha E(x) + \beta E(y) \quad (1.4)$$

$$E[(z - \bar{z})^2] = \alpha^2 E[(x - \bar{x})^2] + \beta^2 E[(y - \bar{y})^2] \quad (1.5)$$

- (a) Run the file and verify the above. Try several examples with different α and β gains and the different statistics of the input GRV's x , and y .
- (b) As mentioned in the textbook, the '*law of large numbers*' states that if you add up a series of n identically distributed random variables (regardless of their distribution), then the sum becomes a GRV in the limit of large n . The simplest case of this idea is when the input is a random ± 1 . From statistics, the pdf of the sum n of these numbers is the binomial distribution.

Look at the overlay of the binomial pdf (histogram) with the equivalent GRV. The mathematical connection is Sterlings's formula. Look up "Sterling's formula" and substitute it into the above expression.

Exercise 1.5 Deterministic pdfs

Open the system:

`Digital Comm\ch_01\deterministic_pdf.svu`






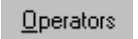

Note that we have deterministic sawtooth, a uniformly distributed noise source, and a sawtooth with a “little” random noise added. In this exercise we generate a pdf for a (periodic) sawtooth wave and we will note that it is a uniform distribution - just because we plot the pdf does not mean that the signal is random!

Run the system and note the histograms (pdfs) that are generated. The pdf from the uniform noise and the noisy sawtooth are essentially the same!

Exercise 1.6 The Cumulative Density Function

Open the system:

Digital Comm\ch_01\intro\cdfs.svu

- (a) Note the signal sources, their parameters and that there are 10000 samples in a simulation. Run the system and view the histograms produced in the  **SystemView Analysis**    windows.
- (b) To generate the cdf we require to integrate the pdf or histogram. Using  choose  then . First the Gaussian histogram, then the uniform histogram.


The resulting two plots show the form of the cdf, although the y-axis scaling is not quite correct... yet. This can be set up as follows.

To correctly scale first note the actual function that was performed by the above integration. For an input plot, $h(k)$ the integrated output, $z(k)$ is:

$$z(k) = \left(\sum_{n=-\infty}^k h(n) \right) \Delta k$$

where Δk is the histogram bin width. The multiplication by Δk is in order that the result is truly the area under the (discrete) function, and of course the continuous time equivalent is:

$$z(\tau) = \int_{-\infty}^{\tau} h(t) dt$$

- (c) The system ran with 10000 samples. For the Gaussian histogram (w2), calculate the bin width, Δk by zooming in to the plot and measuring the spacing between two samples (about 0.180 is about typical).
- (d) Using  choose SCALE, then SCALE AXES, enter the value of $1/\Delta k$ for the y-axis parameter and scale the Gaussian cdf (i.e. the integral of the histogram calculated above). Remember that SystemView will calculate $1/\Delta k$ in-line just by entering $1/\text{"Number"}$ to save doing the division elsewhere.
- (e) The resulting plot is almost the cdf. If the scaling has been done correctly the peak plot value will be 10000 as there were 10000 samples in the simulation. In order to make the peak value 1, given that $F(\infty) = P(x < \infty) = 1$ then it is clear that we should scale the axis again by $1/10000$.
- (f) For the uniform noise histogram, calculate Δk and then scale once by $1/\Delta k \times 10000$ to produce an approximation to the cdf.

Exercise 1.7 Probability Density Function of a Sum of IID Noise Sources

Open the system:

`Digital Comm\ch_01\sum_pdfs.svu`


This system sums two Gaussian random processes both of mean zero, and standard deviation 1. Our random process therefore has a probability density function given by:

$$p(x(k)) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2(k)}{2}}$$

It is well known and relatively straightforward to show that the pdf for the sum of N independent identically distributed (IID) Gaussian noise processes of standard deviation σ will have a standard deviation of:

$$\text{Standard Deviation} = \sqrt{N}\sigma \text{ i.e., } \text{Variance} = N\sigma^2$$

Run the system and in the  **SystemView Analysis**    window view the sum of the five IID Gaussian noise sources, and also the plot of the single noise source.

Use the  button to calculate the mean and standard deviation of the signals. For the sum of the 5 IID Gaussian sources, we might expect the standard deviation to be $\sqrt{N}\sigma$ and noting that $N = 5$ and $\sigma = 1$ then, $\sqrt{5} = 2.24\dots$

The joint pdf of the sum of the five sources will also be Gaussian as shown in the histograms presented in the  **SystemView Analysis**    window.

Exercise 1.8 Resistor Capacitor (RC) Filter

Open the system:

[Digital Comm\ch_01\res-cap-filter.svu](#)

This system simulates a resistor-capacitor (RC) circuit as described in Sections 1.6.3.2 and 1.6.4 of the textbook, and illustrated in Figure 1.3:

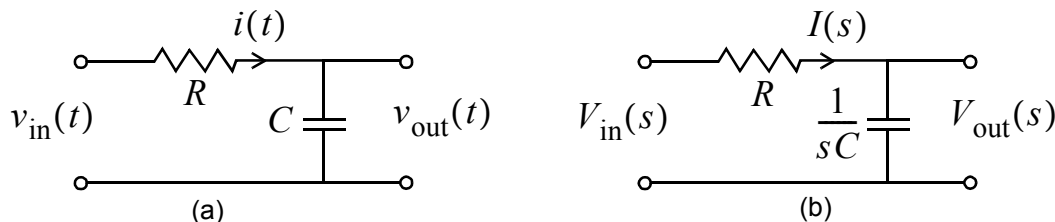


Figure 1.3: (a) Resistor capacitor circuit, (b) Laplace transform representation.

The RC circuit implements a low pass filter. Its transfer function can be expressed as:

$$H(f) = \frac{1}{1 + j2\pi fRC} = \frac{1}{1 + j\omega RC} \quad (1.6)$$



where $\omega = 2\pi f$. From Appendix E, Equation (E.14), we can present this equation as a Laplace transfer function:

$$H(s) = \frac{1}{1 + sRC} \quad (1.7)$$





where $s = j\omega$.

It can be shown that the simple RC filter shown in Figure 1.13 of the textbook has a transfer function, Equation (1.63) equivalent to a 1 pole Butterworth filter whose 3 dB cutoff frequency is:

$$f_c = \frac{1}{2\pi RC} \quad (1.8)$$

In SystemView we can design a first order Butterworth filter by selecting a linear filter system and then  the analog (IIR filter) design tool .

The three filters set up in this exercise are identical and represent a first order lowpass Butterworth filter with a cutoff frequency of 1 Hz. The response of the first order Butterworth filters to an impulse, noise and a 1 sec duration square pulse is simulated.



- Run the simulation and in the in the analysis window  verify that the cutoff frequency is 1 Hz. Verify this by observing the  of the RC circuit *impulse* response. Observe the response of the filter to the 1 sec pulse width.
- In the analysis window use the  to calculate the  of the filtered NOISE. Compare this result with that of the FFT of the impulse response. Can you explain what you see?
- Rerun the simulation for different pulse widths and observe the results.

2 Formatting and Baseband Modulation

Exercise 2.1 Aliasing Frequency Calculation

Open the system:

`Digital Comm\ch_02\aliasing.svu`

In this file a 25 Hz sinewave is sampled at $f_s = 512$ Hz as defined in . Note that a resampling token  is then used and the signal is also sampled at 32 Hz.

Since the output of the resampler is at $f_s = 32$ Hz which is less than twice the signal frequency (50 Hz) aliasing is expected. The 32 Hz sine wave, $s(k)$ can be written in the form:

$$s(k) = s\left(t = \frac{k}{32}\right) = \sin\left(2\pi\frac{25k}{32}\right) = \sin\left(2\pi\frac{(32-7)k}{32}\right) \quad (2.1)$$

- Expand the above equation to determine what the resulting aliased frequency will be. Run the simulation and verify your result.
- Try changing the rate of the sampler and observe that change in the sampled sinewave frequency is consistent with the theory.

Exercise 2.2 Reconstruction of Sampled Signals

Open the system:

[Digital Comm\ch_02\sinc-reconstruction1.svu](#)

This exercise illustrates a random noise source that has been band limited to 1 Hz (one sided). This signal has been sampled at a 5 Hz rate and is reconstructed using a sinc ($\sin x/x$) function. In Equation (2.8) of the textbook, it was shown that the spectrum, $X_s(f)$ of a sampled signal is given by:

$$X_s(f) = \frac{1}{T} \sum_{n=-\infty}^{\infty} X(f - nf_s), \text{ where } f_s = \frac{1}{T_s} \quad (2.2)$$


and $X(f)$ is the spectra of the original unsampled signal.

It was further shown in Appendix E of the textbook, that the perfect reconstruction of the original signal $x(t)$ from its samples $x_s(t)$ is given by:

$$x(t) = \sum_{k=-\infty}^{\infty} x_k \text{sinc}(t - kT_s) \quad (2.3)$$

Taking into account Eq. 2.3 for the number of samples K in the simulation, we have

$$x(t) = \sum_{k=0}^{K-1} x_k \text{sinc}(t - kT_s) \quad (2.4)$$

- Run the simulation to verify the basic operations described. In the analysis window  observe the spectrum of the sampled signal and note the repetition of the spectrum which is represented in Eq. 2.2.
- Observe the input signal (band limited noise), the sampled signal and the reconstructed waveform. Are the input and the reconstructed signals equal?
- Note that in order to perfectly reconstruct the sampled signal as shown in Eq. 2.3 a perfect sinc function is required, i.e with an infinite number of taps. Observe the impulse response of the sinc function used by opening the parameter dialog box of the sinc token (token 6) and note, by the finite number of taps, that it is a truncated sinc waveform as shown in [Figure 2.1](#) since the impulse response does not completely decay to zero.

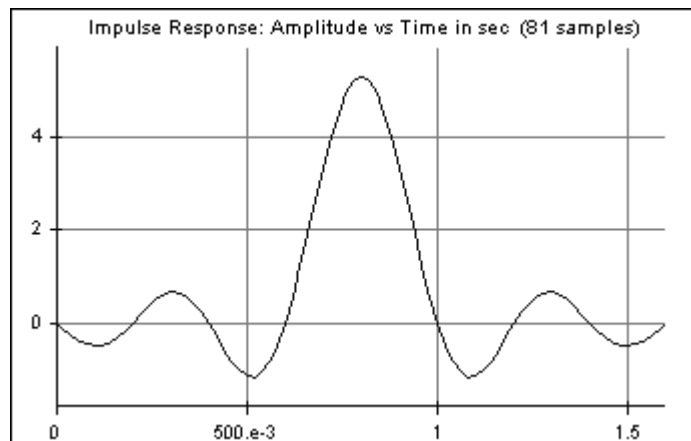
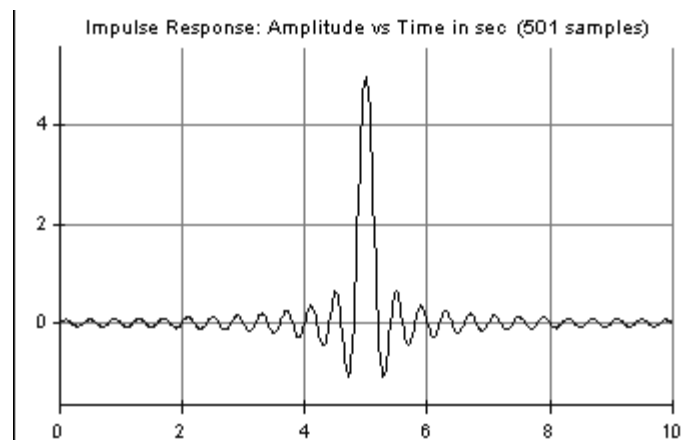


Figure 2.1: Truncated sinc waveform (81 taps)

- (d) In order to obtain a more accurate representation of a sinc signal the number of taps of the sinc filter have to be increased. :

Digital Comm\ch_02\sinc-reconstruction2.svu

The sinc used in this exercise is represented by a 501 tap filter with the impulse response shown in Figure 2.2. Compare with Figure 2.1.





- (e) Run the simulation and observe the input and reconstructed waveform. Is this reconstruction *nearly* perfect? Of course an infinite number of taps are required in the sinc filter to attain perfect reconstruction.
- (f) Reduce the sampling rate in the  and observe the new spectra.

Figure 2.2: Truncated sinc waveform (501 taps)

Note when you change the sample rate the sinc reconstruction filter  must be adjusted accordingly. To do this double click on the token and when the Operator library form appears select **Edit Parameters...**. When the linear system design form appears select **Comm...** at the right top of the form. Select **Sin(t)/t** as the filter, and finally in the filter design form change the **Symbol Rate (Hz)** parameter to the new sample rate selected. Click OK on the forms all of the way back to the design screen.

How low a sample rate can the signal tolerate before aliasing becomes a problem?

Exercise 2.3 Oversampling

Open the system:

`Digital Comm\ch_02\8x-oversampled-sys.svu`

Section 2.4.3 of the textbook lists a variety of reasons why it is desirable to sample a band limited system at a rate higher than the minimum required Nyquist rate (oversample). This exercise adds to the list via a very practical application of oversampling that is commonly used in audio CD systems.

In Section 2.5.3 of the textbook, it is shown that the quantization noise power after an ADC operation is given by $q^2/12$, where q is the ADC quantization step size. Note that there is no mention of the sampling rate in this equation.

- (a) Assuming the frequency spectra of the quantization noise is uniform (i.e. white), show that the power spectral density of the noise, N_q , is given by

$$N_q = q^2/f_s \quad (2.5)$$

where f_s is the system sampling rate.

- (b) For a fixed q , what does Eq. 2.5 say about the quantization noise floor as a function of the sampling rate?

The exercise illustrates the improvement in quantization noise by use of oversampling. A Nyquist system and an oversampled system (8x) are running side by side. The input signal is Gaussian noise band limited to 4 kHz.

- (c) What is the Nyquist sampling rate for this signal?

- (d) Run the exercise and verify that the quantization noise power (variance) is the same for both systems.
- (e) In terms of the quantization spacing q , calculate the value for N_q for both systems.
- (f) Finally, the 8x system has quantization noise but no signal beyond 4 kHz. Therefore the 8x signal can be filtered and resampled back to the Nyquist frequency of 8 kHz. Now compare the noise floor for both systems. Explain your result.

Exercise 2.4 The Impulse Sequence as a Sum of Cosines

Open the system:

[Digital Comm\ch_02\sum_of_cosines.svu](#)

Equation 2.6 in the textbook states that the Fourier Transform (FT) of the pulsetrain is another pulse train. Specifically, if

$$x(t) = \sum_{k=-\infty}^{\infty} \delta[t - kT_s] \quad (2.6)$$

is the sampling function, then the FT of $x(t)$ is given by


$$X(f) = \frac{1}{T_s} \sum_{k=-\infty}^{\infty} \delta[f - k/T_s] \quad (2.7)$$

(a) Use FT theory to show that,

$$\text{FT} \left\{ \sum_{n=-\infty}^{\infty} \frac{\epsilon_n}{T_s} \cos(2\pi n t / T_s) \right\} = \frac{1}{T_s} \sum_{n=-\infty}^{\infty} \delta[f - n/T_s] \quad (2.8)$$

$$\text{with } \varepsilon_n = \begin{cases} 1, & n = 0 \\ 2, & \text{otherwise} \end{cases}$$

Thus a very useful way of looking at the sampling function is to consider it as the sum of an infinite number of oscillators with frequency spacing $1/T_s$.

- (b) The sums together cosines at 1 Hz, 2 Hz, 3 Hz.... 9 Hz of amplitude 1 volt, and cosines at 0 Hz (DC!) and 10Hz both of amplitude 0.5 volts. The sampling rate in  is set to 20 Hz.


Run the system and note that the result is indeed a train of impulses.

Exercise 2.5 Quantization Noise

Open the system:

`Digital Comm\ch_02\quantization-noise.svu`

This system is a simple quantizer using uniform noise as an input. The data is quantized into 4 bits or 16 levels.

- (a) Run the simulation.
- (b) Using Equation (2.18b) in the textbook, calculate the expected quantization noise power variance. Compare with the results displayed. Why aren't they exactly the same?
- (c) Repeat the above with various levels of quantization.
- (d) Observe at the quantized noise waveform in the analysis window . Notice the asymmetry with respect to ± 1 . Compare this picture with Figure 2.15 in the textbook. Explain the difference.
- (e) As noted in the textbook, every time a bit is added to the quantization level, the spacing is cut in half. Since the power is the square of the amplitude, what happens to the quantization noise power? How many dB is the quantization noise reduced per bit of quantization? How many bits of quantization are required for the average quantization noise power to be 96 dB down below the peak signal power?

Exercise 2.6 Interpolation of a Signal

Open the system:

Digital Comm\ch_02\interpolation.svu

This system is interpolating from 10kHz sampling rate to 40kHz as in Figure 2.3

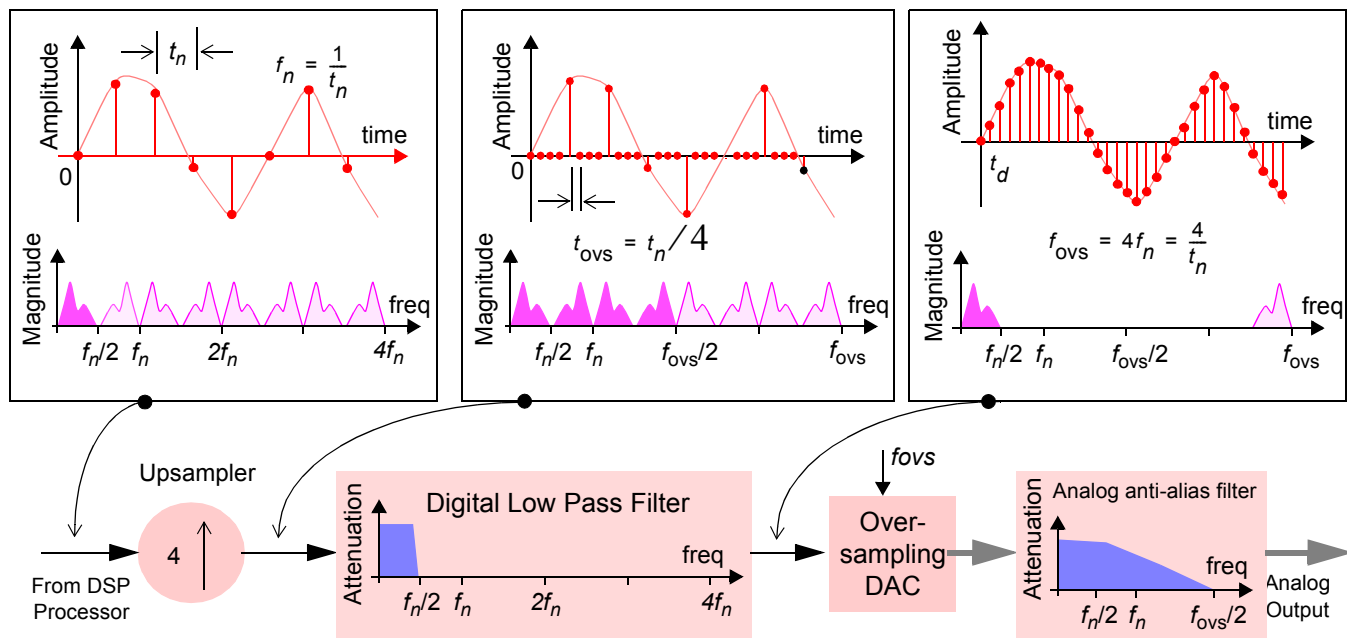


Figure 2.3: 4x's Interpolation of a signal by upsampling and low pass filtering.

Run the system and in the analysis window observe the time domain and frequency domain spectra as illustrated in Figure 2.3 at the various stages of the simulation.

Exercise 2.7 Sampling Rate Conversion

From a system sampling at 48000 Hz, design a system that will convert to an 8kHz sampling rate using a suitable decimation system with a downsampling factor of 6 as shown in Figure 2.4.

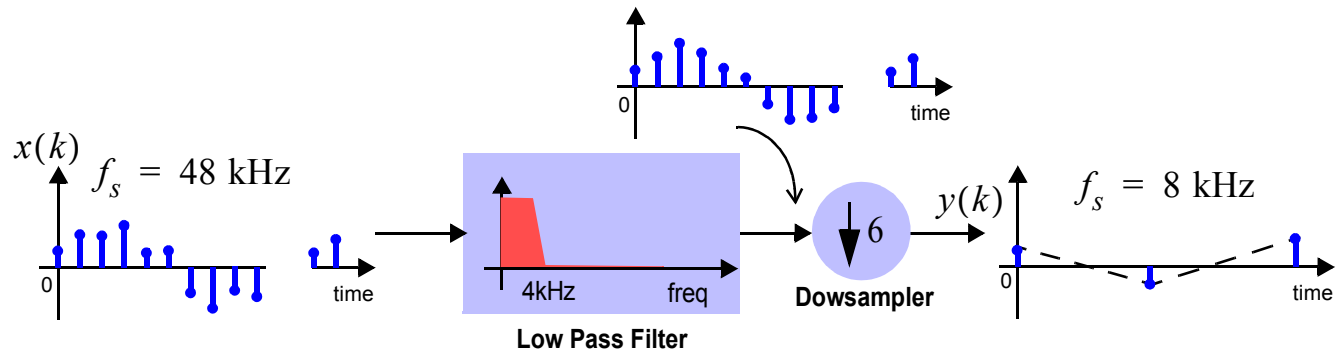


Figure 2.4: Decimation from 48kHz to 8kHz

You should use a suitable low pass filter, and a decimation token (see OPERATOR-SAMPLE/HOLD-DECIMATOR) set to a suitable downsampling factor. What would be the implication of not using a linear phase FIR filter?

Exercise 2.8 4x's Oversampling Rate and the 6 dB Advantage

Open the system

`Digital Comm\ch_02\oversamp\oversmp.svu`

This system is demonstrating the 6 dB resolution advantage gained by oversampling and decimating by a factor of 4. In the example we take a broadband signal with components of interest from 0 to 5000 Hz, and sample at Nyquist by using a low pass anti-alias filter and a 10,000 Hz sample. We also take the broadband signal, oversample at 160,000 Hz to “spread” the quantization noise, then decimate to 10,000Hz. Using the oversampling strategy we effectively get 6 bits of resolution from the 4 bit ADC.

Exercise 2.9 Duobinary Exercise

Open the system:

`Digital Comm\ch_02\duobinary.svu`

This exercise implements the precoded duobinary system as described in Section 2.9.3 of the textbook.

- (a) Run the exercise file.
- (b) Work through the decoder tokens to verify the decision making criteria.

Verify the data as it goes from x_k to the input of the Nyquist filter.

3 Baseband Demodulation Detection

Exercise 3.1 Transmitting Impulses

If we are to send binary data (i.e. 0's and 1's), a suitable mechanism might be to send a +1 volt for a binary "1" and a -1 volts for a binary "0":

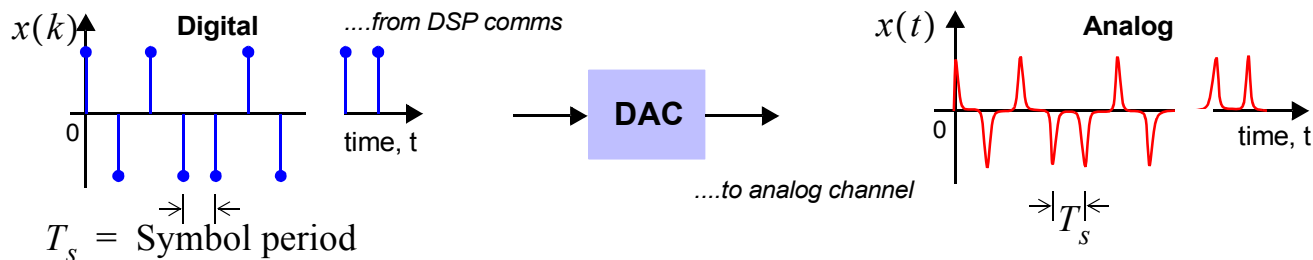


Figure 3.1: *Transmitting binary data.*

The above figure shows a DSP communication system transmitting data at a rate of $1/T_s$ symbols per second. However the bandwidth required to transmit this data is somewhat higher given that each bit is represented by a pulse shape approaching that of an impulse. (Recall that an impulse contains "all" frequencies.)

Open the system:

`Digital Comm\ch_03\impulse_data.svu`

which is representative of the above Figure 3.1, except that we do not actually perform the ADC step in our digital simulation.

In the examples which follow the data rate is always set to 2400 bits/sec for consistency

$$T_s = \frac{1}{2400} = 0.00041666 \text{ seconds}$$

Note the simulation sampling frequency is set at the high value of 96,000 Hz (40×2400) to simulate the final output being “analog”.

Run the system and in the  window note that the impulsive data requires the entire available bandwidth to transmit the data; this can be viewed from the magnitude FFT of the data sequence.

Exercise 3.2 Square Pulse Shaping

To reduce our bandwidth requirements we clearly need to reduce the high frequency content of the data pulses. Hence, in this example we will use a PULSE SHAPING FILTER, whereby prior to transmission the data (impulses) will be shaped to be square pulses:

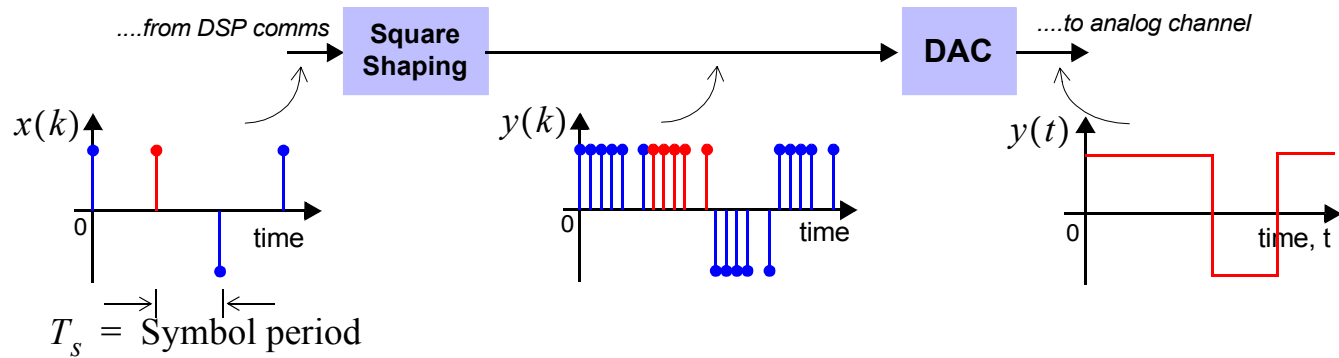


Figure 3.2: Square pulse shaping.

This pulse shaping can be easily accomplished using a digital filter with a square impulse response of duration 1/2400 second.

Open the system:


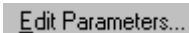
Digital Comm\ch_03\squ_pulse.svu


In this system we have introduced a custom picture for the square pulse shaping filter as shown in Figure 3.3.



Custom picture for a LinearSys token

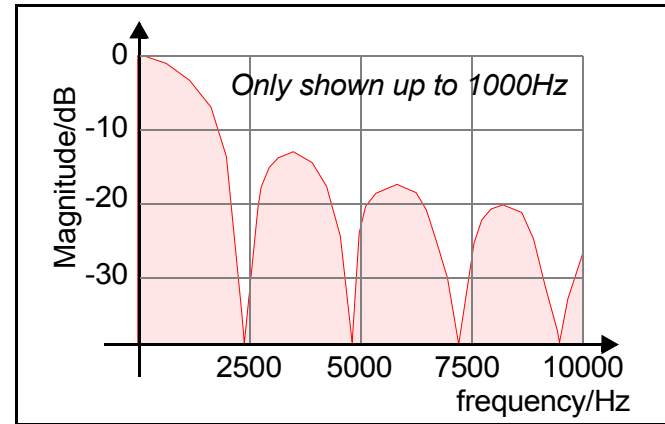
Figure 3.3: Custom token pictures.





This is to “simplify” the diagram and (hopefully) make more readable. If you   on this “Square Shaping” token you will see that it is a normal SystemView linear system token.

(SystemView allows you to change the icon/picture of any token to one of your choosing. To see this facility  on any token.)

- (a) View the impulse response and filter weights (all set to $1/40 = 0.025$) of the square pulse shaping filter and confirm it is a 40 weight filter.

Recalling the sampling rate was 96,000 Hz the duration of the impulse response of this filter is $64/96000 = 1/2400$ -th second. Confirm the magnitude frequency response of the filter is as shown on the right (we have only plotted up to 10,000 Hz), and that the filter has linear phase.



- (b) Run the system in the  **SystemView Analysis**  window view the transmitted pulses (which will be output from a DAC) and also view the bandwidth of the square pulse train by viewing the magnitude frequency spectra of the square pulse waveform.
- (c) Increase the number of samples to 8192, run the system again, and in the  **SystemView Analysis**  window note the magnitude frequency spectra of the square pulse waveform which should now be smoother (more averaged) than the previous run.

Exercise 3.3 Return to Zero Square Pulse Shaping

In this example we can modify the PULSE SHAPING FILTER, to have a duration of 1/4800 the second rather; hence we might be tempted to call this a return-to-zero (RZ) square pulse:

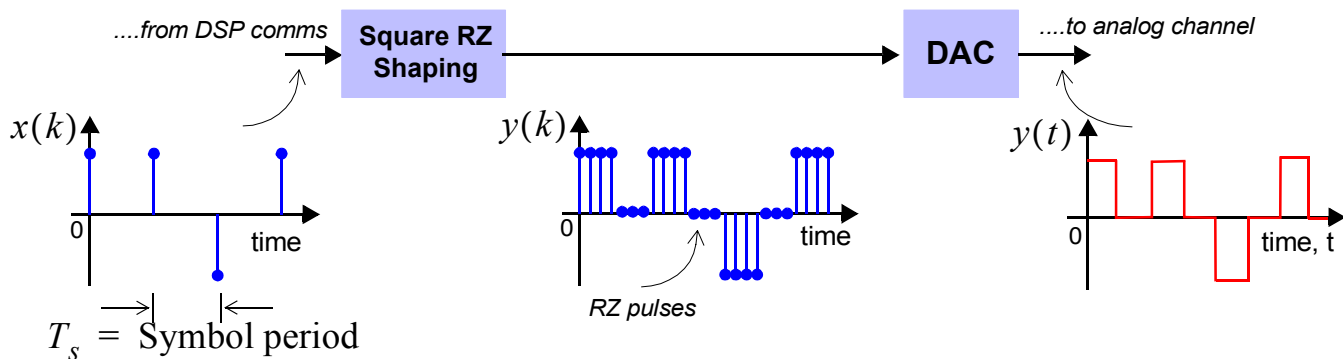


Figure 3.4: Square Pulse Shaping.

This pulse shaping can be easily accomplished using a digital filter with a square impulse response of duration 1/4800 second.

Open the system:

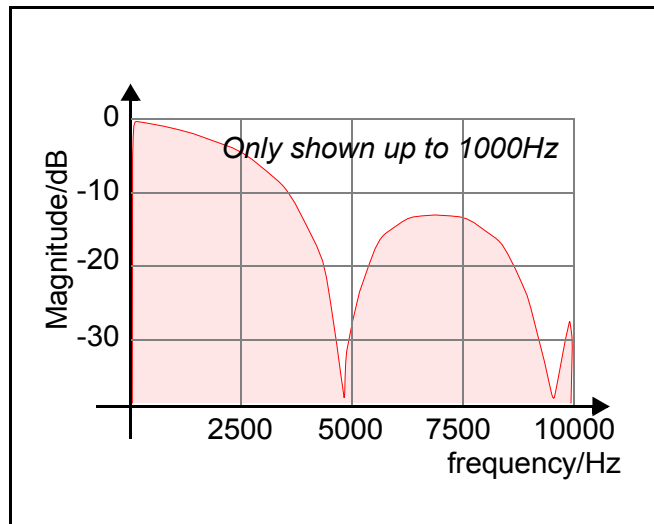
`Digital Comm\ch_03\squ_pulse_rz.svu`


Note that again just to simplify the screen view (reducing the token clutter), we have created a simple meta system for the 2400 bps generator and used a custom picture

(purple token on the left and labelled data stream).

- (a) View the impulse response and filter weights (all set to $1/20 = 0.05$) of the square pulse shaping filter and confirm it is a 20 weight filter.

Recalling the sampling rate was 96,000 Hz the duration of the impulse response of this filter is $20/96,000 = 1/4800$ -th second. Confirm the magnitude frequency response of the filter is as shown on the right (again we have only plotted up to 10000Hz), and that the filter has linear phase.



- (b) Run the system and the  window view the transmitted pulses and also view the bandwidth of the square pulse train by viewing the magnitude frequency spectra of the square pulse waveform. As necessary Increase the number of samples to produce a smoother (more averaged) frequency plot.

Exercise 3.4 Recovering Data by Sampling

Open the system:

`Digital Comm\ch_03\squ_pulse_recover.svu`

where the pulse shaping is the same as in Exercise 3.2. In order to recover the data it is a simple matter of sampling the received square pulse at approximately the middle of the pulse. This is done with a simple sampler which is timed (or synchronised) to sample at the middle of a pulse.

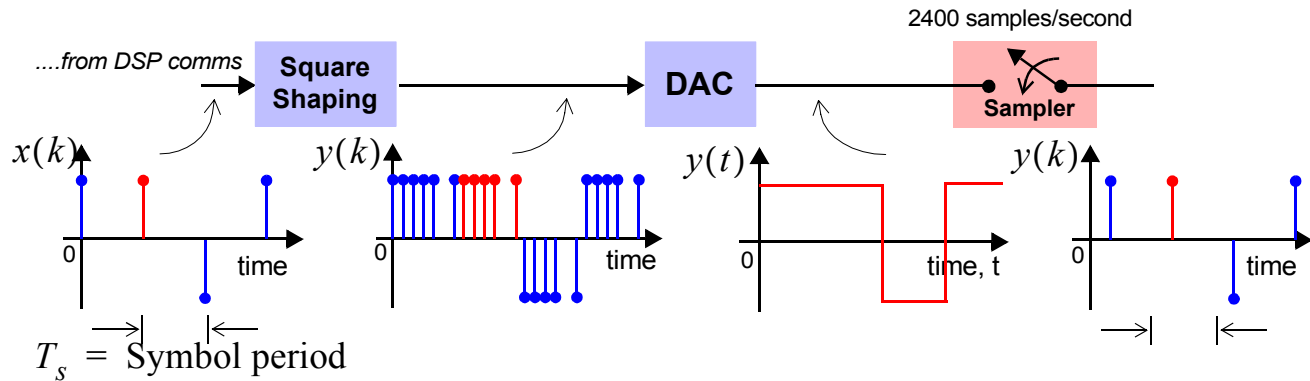


Figure 3.5: *Recovering data by synchronized sampling*

Run the system and confirm in the  **SystemView Analysis** window that we have in fact recovered the original 2400 bps data.

What is the latency/delay associated with the recovery of each sample?

Exercise 3.5 Data in a Noisy Channel

Open the system:

Digital Comm\ch_03\squ_pulse_noisy.svu

In this example we will assume that the data is transmitted over a channel which introduces a low level of noise. The pulse shaping is again the same as in Exercise 3.2:

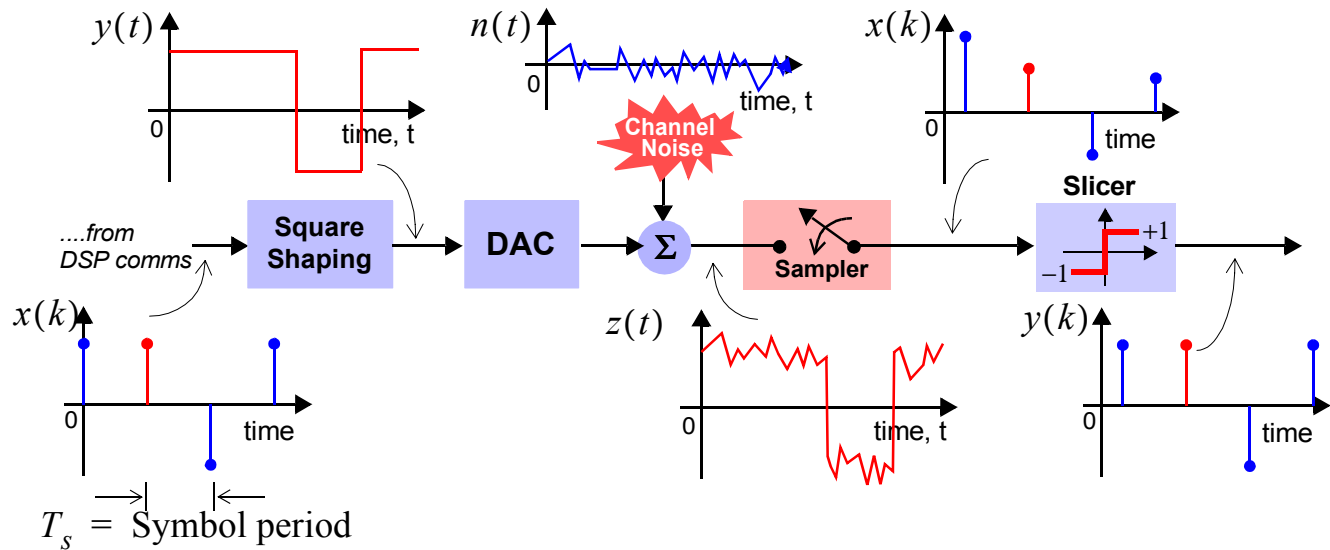


Figure 3.6: Recovering final data values by slicing.

Because of the channel noise, when the sampler produces an output every T seconds the result will actually be:

$$y(k) = z(kT) + n(kT)$$

Noting that a sample is given by a 1 or -1, if the noise at the specific sampling instant is $n(kT) > 1$ when the data is a -1, or when $n(kT) < -1$ when the data is 1, a bit error will result - i.e. a wrongly detected bit.

In this example a BER token (bit error rate) is used to count the bit errors. This token simply compares two input bits (i.e the transmitted and received bits) and produces outputs that reflect the number of detected errors so far. Before continuing inspect the parameters of, and outputs available from this token as it will be extensively for other data communications systems.

- (a) Run the system and confirm that for a noise with a standard deviation of $0.1/40 = 2.5 \times 10^{-3}$ (note that the square pulse amplitude is 1/40 volts after being shaped) there are no bit errors by inspecting the transmitted and received data bits; and also by checking the output of the BER token.
- (b) Now increase the channel Gaussian noise to a standard deviation of $1/40 = 2.5 \times 10^{-2}$. You should see some bit errors occurring now.
- (c) Increase the no. of samples to $40 \times 1000 = 40,000$, corresponding to 1000 data bits. How many bit errors do you get now?
- (d) Increase the channel noise even higher to a standard deviation of 2.5×10^{-1} . What is the BER now?

Exercise 3.6 Bit Error Rate in a Uniform Noise Channel

Open the system:

`Digital Comm\ch_03\uniform_noise_channel.svu`

The noise on the channel in this exercise is uniformly distributed with a probability density function as shown in Figure 3.7. Therefore the noise samples are likely to be between $\pm 1.2/40$. After the pulse shaping the data pulses have an amplitude of $\pm 1/40$. If the data input to the slicer was greater than 0 volts then 1 volt (binary 1) was output, and if less than 0 volts then a (binary 0) -1 output.

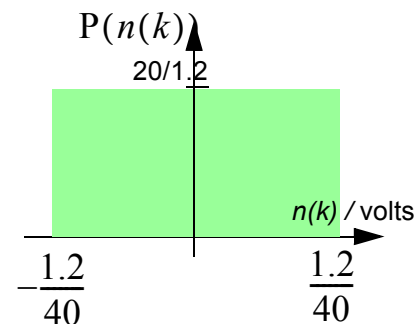


Figure 3.7: Uniform pdf.

Therefore there are two situations for an error to occur:

“1” sent (amplitude $1/40$ volts) and a “0” detected - $P(0/1)$;

“0” sent (amplitude $-1/40$ volts) and a “1” detected - $P(1/0)$.

The probability notation $P(A/B)$ is usually named a “conditional probability” as we are stating the probability of event A given the event B has already occurred.

$P(A/B)$ - Symbol B was transmitted but symbol A was detected.

When data is sent we assume that over a period of time it is equally likely that a “1” or a “0” will be sent. The probability of sending a “0” is therefore: $P(0) = 0.5$ and sending a 1 is $P(1) = 0.5$. This is of course the case in most digital communication systems; if it were not the case (e.g. there were many more 1’s than 0’s) we would try and force it

to be the case by additional forms of coding. Therefore errors happen when we send a “1” and detect a “0” and when we send a “0” but detect a “1”. The overall probability of an error occurring, $P(\text{Error})$ is therefore:

$$P(\text{Error}) = P(1)P(0/1) + P(0)P(1/0) \quad (3.1)$$

From the uniform pdf curve we can represent the region of the noise that could cause an error if a symbol “1” was sent (i.e. 1/40 volts). Recall for a pdf that the total area under the curve is 1. To find the probability a certain region simple integrate over the region of interest. For this simple uniform pdf it is straightforward calculate the area:

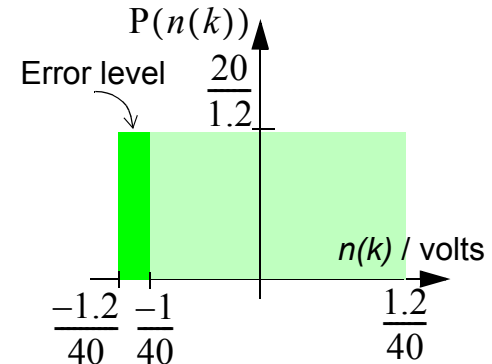


Figure 3.8: Uniform noise pdf.

$$P(0/1) = P(n(k) < -1/40) = \frac{0.2}{2.4} = 0.08333 \quad (3.2)$$

Hence the probability of error is:

$$\begin{aligned} P(\text{Error}) &= P(1)P(0/1) + P(0)P(1/0) \\ &= (0.5 \times 0.08333) + (0.5 \times 0.08333) \\ &= 0.08333 \end{aligned} \tag{3.3}$$

In this example the channel noise has a uniform probability distribution as shown above in [Figure 3.7](#). As before the data transmitted is square pulse shaped and detected by a slicer as shown in [Figure 3.6](#).

Noting that we calculated the theoretical probability of error to be 0.0833, we can now confirm this by simulation. The system is set up to run for 4000 samples, corresponding to 100 bits sent.

- (a) Run the system and note the total number of errors detected from the BER token which is currently set up to add a 1 to its output each time an error is detected. Note that we are displaying using a FINAL VALUE sink as well as a sink showing the running total of errors so far.
- (b) How many errors did you get? About 8 possibly? - if not run again! Around 8 would be right noting that $P(\text{Error}) \approx 0.08$, hence over 100 bits we note $100 \times 0.08 = 8$ errors.
- (c) Increase the number of samples to 40000 (i.e. 1000 bits sent). How many errors do you see? About 80 would concur with our prediction.
- (d) Increase the number of samples to 400000 (i.e. 10000 bits sent). How many errors do you see? About 800 would be the predicted value.

- (e) Increase the range of the noise to $\pm 1.4/40$ (remember to change both Maximum Value and Minimum Value parameters in the uniform noise token). For a run of 400000 samples, or 10000 bits how many errors do you get now?

Calculate the theoretical $P(\text{Error})$ for this new noise. Your result *should* agree with the simulation result.

Exercise 3.7 Bit Error Rate in a Gaussian Noise Channel

Open the system:

Digital Comm\ch_03\gaussian_noise_channel.svu

For this exercise the channel noise is AWGN with a standard deviation of $\sigma = 1/40$ as shown below.

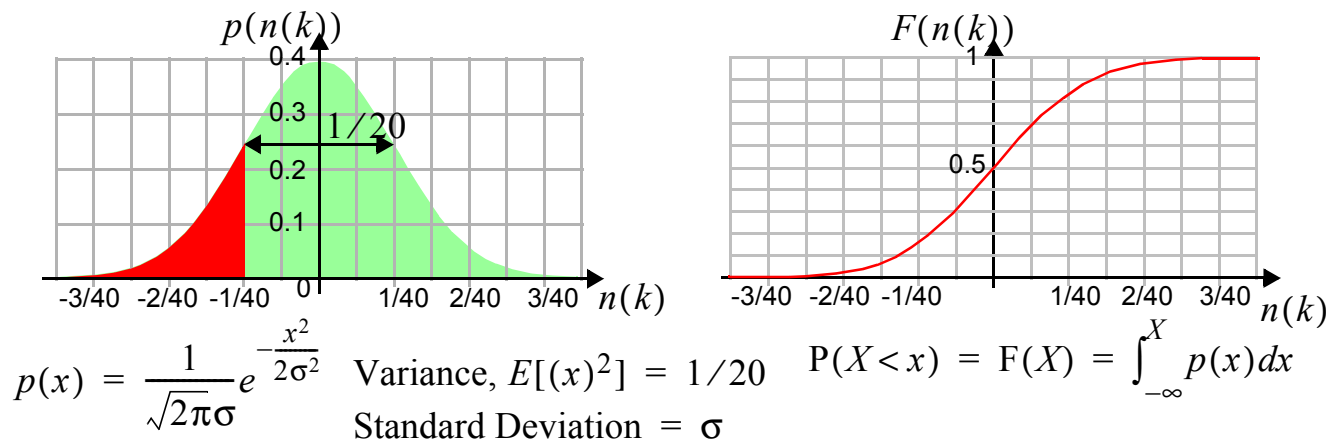


Figure 3.9: Gaussian pdf of mean = 0 and variance = 1/20.

Hence the probability of the noise having a value less than $-1/40$ is about $P(n(k) < -1/40) = 0.15$. Hence the probability of a bit error is:

$$\begin{aligned}P(\text{Error}) &= P(1)P(0/1) + P(0)P(1/0) \\&= (0.5 \times 0.15) + (0.5 \times 0.15) \\&= 0.15\end{aligned}\tag{3.4}$$

As before the data passes is square pulse shaped and detected by a slicer as shown in Figure 3.6.

Noting that we calculated the theoretical probability of error to be 0.15, we can now confirm this by simulation. The system is set up to run for 4000 samples, corresponding to 100 bits sent.

- (a) Run the system and note the total number of errors detected from the BER token.
- (b) How many errors did you get? About 15 possibly? - if not run again! Around 15 would be right noting that $P(\text{Error}) \approx 0.15$, hence over 100 bits we note $100 \times 0.15 = 15$ errors.
- (c) Increase the number of samples to 40000 (i.e. 1000 bits sent). How many errors do you see? About 150 would concur with our prediction.
- (d) Increase the standard deviation of the noise to 1/20. For a run of 40000 samples, (1000 bits) show many errors do you get now?

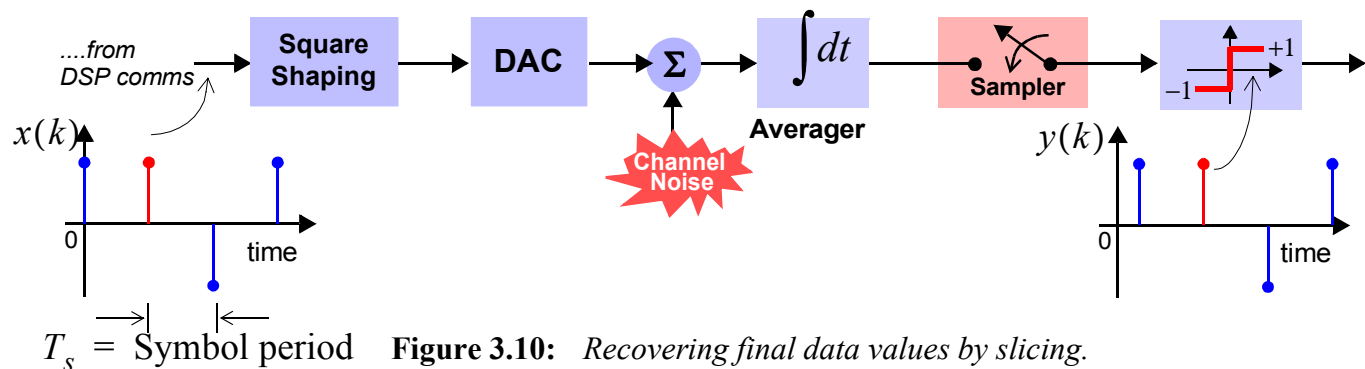
Calculate the theoretical $P(\text{Error})$ for this new noise by appropriately rescaling the pdf of Figure 3.9. Your result *should* agree with the simulation result.

Exercise 3.8 Improving the Detection Rate by Averaging

In this example we will assume that the data is transmitted over a channel which introduces a low level of noise in the same manner as [Exercise 3.2](#). In the previous example we decided on a bit being a 1 or -1 based on one single sample taken in the middle of the square pulse. If this just happened to coincide with a high level of noise at **exactly** the sampling instant then there was a bit error. A more useful process would be to derive the data bit by looking over the entire pulse duration ($1/2400$ th in our case). Hence prior to the sampler we can integrate over the duration of one data bit, then sample the result and finally make a decisions (+1 or -1), i.e. slice the data.

A basic correlator performs the integration of a received signal that has been multiplied by a candidate signal (see Section 3.2 in the textbook). Notice that the integration/averaging seen here represents the correlation of a received noisy pulse with an ideal unity amplitude pulse.

Hence our data communication system is now:



The integration \int symbol is often used for the averager. The token used in the SystemView simulation is a simple moving average which outputs the average from the last K seconds. In the simulation we have set this to be $40/92000 = 416.667 \times 10^{-6}$, corresponding to an average of the last 40 samples which of represents one square pulse.

Open the system:

[Digital Comm\ch_03\gaussian_noise_channel.svu](#)

- (a) Set the noise to a standard deviation of $1/20$. Therefore the probability of error can be calculated as previously to be:

$$\begin{aligned} P(\text{Error}) &= P(1)P(0/1) + P(0)P(1/0) \\ &= (0.5 \times 0.3) + (0.5 \times 0.3) \\ &= 0.3 \end{aligned}$$

i.e 300 bit errors for every 1000 bits (use the error functions of the textbook to calculate).

- (b) Now open the system:

[Digital Comm\ch_03\gaussian_noise_chan_averag.svu](#)

where the value of the square pulse is averaged over the 40 samples of its duration corresponding to a time average of $40/96000 = 0.00041667$ seconds.

Has this averaging improved over the 0.3 error rate above?

- (c) Increase the no. of samples to $40 \times 1000 = 40,000$, corresponding to 1000 data bits. How many bit errors do you get now?





Increase the channel noise even higher to a standard deviation of 2.5×10^{-1} . What is the BER now?

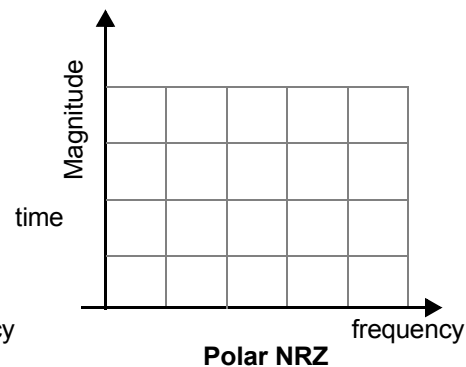
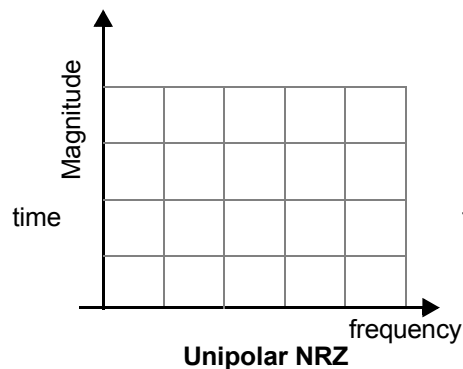
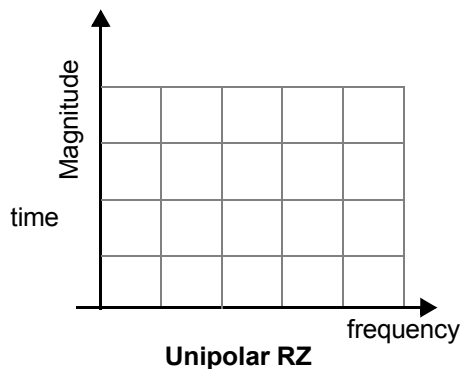
So clearly by averaging we have greatly reduced the number of bit errors. In the above exercise an error is only generated if the **sum of the previous 40 noise samples** is greater than $1/40$. Noting the channel noise had a zero mean, this is at best “unlikely”.

Exercise 3.9 Binary Signalling

Open the system:

`Digital Comm\ch_03\binary_signalling.svu`

- The signalling pulse duration is 0.1 seconds, hence the data rate is 10 bits/second. Sampling rate is 100 Hz, hence one bit is represented by 10 samples. The data source is a text file of 1's and 0's, which has the initial sequence 0101011100100101000000....
- Run the simulation and compare the different signalling trains in the  **SystemView Analysis**  window.
- Increase the number of samples to 10000 and run the system again. In the  **SystemView Analysis**  window view the magnitude frequency spectra that have been generated. Sketch the spectra below:



Exercise 3.10 Integrate and Dump Filter

Open the system:

`Digital Comm\ch_03\integrate-and-dump.svu`

The top section of this system is known as the integrate and dump matched filter. Determine the impulse response $h(t)$ of this circuit.



- Determine the frequency transfer function $H(f)$ of this circuit, and the squared frequency transfer function $|H(f)|^2$. For what values of f is $|H(f)|^2 = 0$?
- Evaluate $|H(f)|^2$ for frequencies $f = (k + 1/2)/T$, $k = 1, 2, 3, \dots$. Where are these points on the spectra? Normalize your answer with respect to the value at $f = 0$, and convert the result to dB. A good system number to remember is the result for $k = 1$, which is known as the first sidelobe of the spectra.
- Run the system with $T = 0.5$ sec. Verify your calculations to the above using the results shown.
- Change the two delay tokens from 0.5 to 2 sec., and change the 'start time' in the step function token to 2. Also compare the spectra obtained from the two delays. Where do the nulls of the spectra move to, and what happens to the value of the spectra (not in dB) at $f = 0$ Hz? Calculate the product of the first null frequency and the spectral value at $f = 0$ Hz in both cases. Comment on your result.
- Another way to generate this signal (which is actually done in some cases) is to take an integrator and zero out (dump) the output every two seconds. This technique is the basis of the name, '*integrate and dump*'.

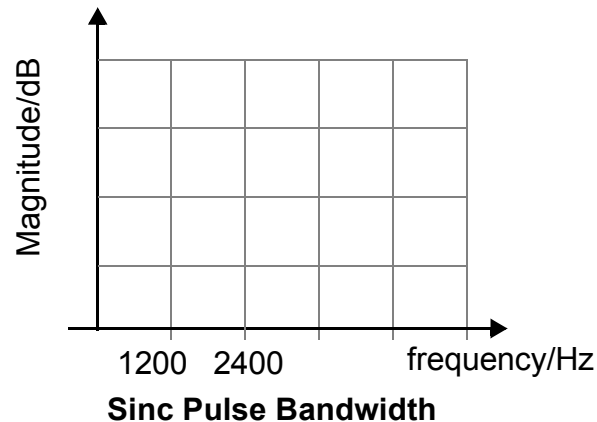
Exercise 3.11 Sinc Pulse Shaping

Open the system:

`Digital Comm\ch_03\sinc_pulse.svu`

The data impulses at the rate of 2400 bits/sec are shaped by a sinc pulse.


- Run the system and note in the  **SystemView Analysis** window that there is NO intersymbol interference (ISI) occurring due to the zero crossings of sinc pulses occurring at the data sampling intervals.
- Increase the number of samples to 16384, run the system again, and in the  **SystemView Analysis** window take the 20logFFT spectrum of the received sinc pulse shaped signal. Confirm that the required bandwidth is around 1200 Hz.
- Sketch the bandwidth required (compare with the values suggested in the textbook).

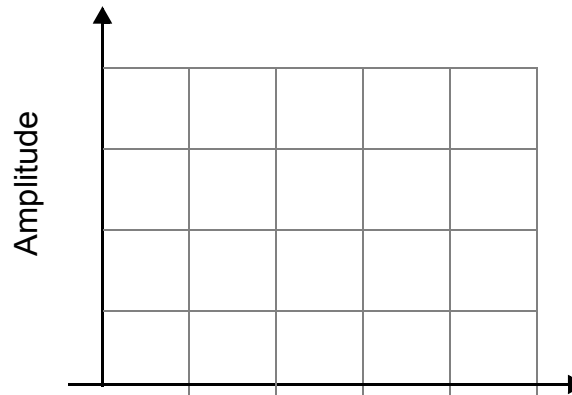


Exercise 3.12 Generating an Eye Diagram

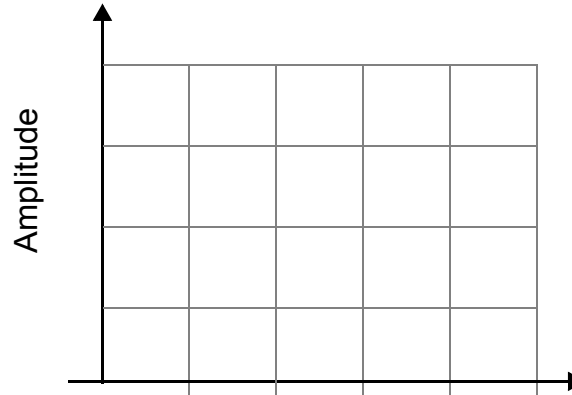
In this example we will generate the eye diagram for the signal in the previous example:

`Digital Comm\ch_03\sinc_pulse.svu`

- (a) After running the system, in the analysis window choose then from the . Set the start time to an "appropriate value, and also set the time slice to the symbol rate. You should see only two significant crossing points which represent the sampling instants (Note you will also see the zero crossing point). Sketch the eye diagram for this signal below:



- (b) Return to the design space and modify the data source to be a 4 level signal (you will require to open the “data stream” meta system) and then return to view the eye diagram in the analysis window. Sketch the eye diagram for this signal below:





- (c) Add a low level of Gaussian noise to the shaped pulses and note the effect on the eye diagram.
- (d) Remove the low level of noise, and pass the output through a simple phase distorting channel (perhaps design a simple IIR that essentially passes up to 4800 but distorts the phase), and again note the effect on the eye diagram.

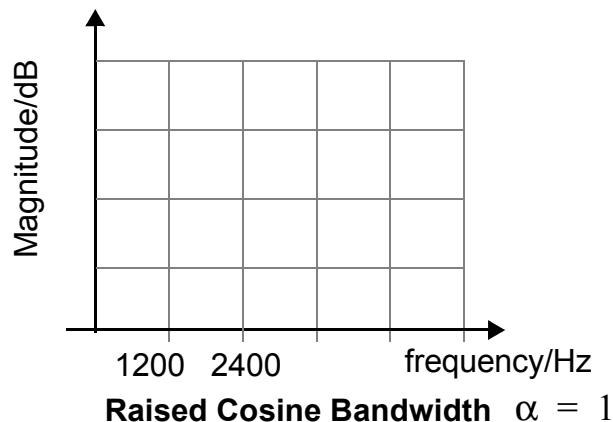
Exercise 3.13 Raised Cosine Pulse Shaping

Open the system:

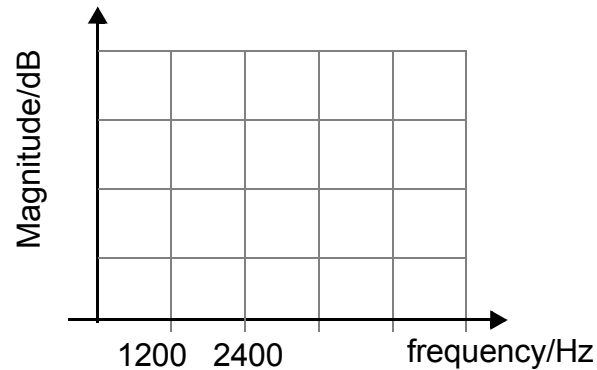
`Digital Comm\ch_03\raised_cosine_pulse.svu`

The data impulses at the rate of 2400 bits/sec are shaped by a raised cosine pulse. The raised cosine pulse has $\alpha = 1$ in this example.

- Run the system and note in the  **SystemView Analysis** window that there is NO intersymbol interference (ISI) occurring due to the zero crossings of raised cosine pulses occurring at the data sampling intervals.
- Increase the number of samples to 16384, run the system again, and in the  **SystemView Analysis** window take the 20logFFT spectrum of the received raised cosine pulse shaped signal. Confirm that the required bandwidth is now 2400 Hz. Hence sketch the bandwidth required below.



- (c) Change the roll-off parameter, α , to 0.22 in the raised cosine filter dialog box (see LINEARSYS/FILTER-COMM), rerun the system. Note there is still zero ISI, but the excess bandwidth is reduced. Sketch the bandwidth required.



Raised Cosine Pulse Bandwidth $\alpha = 0.22$

- (d) Generate the eye diagram for this system.
(e) Change the shaping filter to a root raised cosine. Note there is now some ISI.

Exercise 3.14 Root Raised Cosine Pulse Shaping and Matched Filtering

The ideal requirement for zero ISI is of course over the entire channel. Therefore the pulse shaping filters must be judiciously placed within the overall transmission link given this requirement, and the matched filtering requirement. Recall that there are two clear requirements of the pulse shaping: (1) to limit the bandwidth, and (2) to have zero ISI over the link.

In practice we often use root raised cosines (RRC) rather than raised cosine. This means that the combination of an RRC at the transmit and the receive (thus forming a matched filter pair) gives a desirable raised cosine filter from input to output.

(a) Modify the previous system:

`Digital Comm\ch_03\raised_cosine_pulse.svu`

to have an RRC at the transmit and an RRC at the receive. Confirm there is no ISI, and also that the response of two cascaded RRC filters is a raised cosine.

(b) Modify your system to include a modulation stage of the signal up to a frequency of say 48,000 Hz, and then a demodulation and low pass filtering stage using a “suitable” low pass filter design.

Exercise 3.15 Raised Cosine Pulse Analysis

Open the system:

`Digital Comm\ch_03\raised_cosine1.svu`

In Section 3.3.1 of the textbook it was mentioned that one disadvantage of the Nyquist sinc pulse is its sensitivity to timing errors. Another difficulty is the slow roll-off of the sinc function with respect to t .

- Show analytically that the raised cosine pulse falls off as $1/t^3$ for large t . Since we can only realize either of these filters over a limited time extent, the raised cosine filter takes less time to diminish to a specified level than the sinc function.
- In the SystemView file the impulse response of the sinc and raised cosine filter ($r = 0.5$), and $T = 1$ sec are generated for a 10 sec duration. View these waveforms in an overlay mode.
- View the power spectra of the corresponding time waveforms. What can be concluded about the practical implementation of the two waveforms over a finite time period?
- Repeat (a) and (b) by changing the time extent from 10 to 40 sec (e.g., increase the number of taps in the filter by a factor of 4). Observe the change in the spectra of the sinc function vs. any change in the raised cosine spectra. Comment on the results.

Exercise 3.16 Raised Cosine, Root-raised cosine, and Sinc Filters

Intersymbol interference (ISI) can seriously degrade the performance of a digital communication system. Care must be taken when choosing a pulse shaping filter to insure that ISI is not introduced. The common technique for observing ISI is the so called 'eye diagram'. The eye diagram is obtained by taking the data waveform and folding it back on itself modulo the data rate. [Figure 3.11](#) below shows such a diagram with no ISI. Note the sharp points in the centre where all of the traces converge. This is the sample time for best recovering the data. [Figure 3.12](#) is a similar plot which indicates

the presence of ISI.

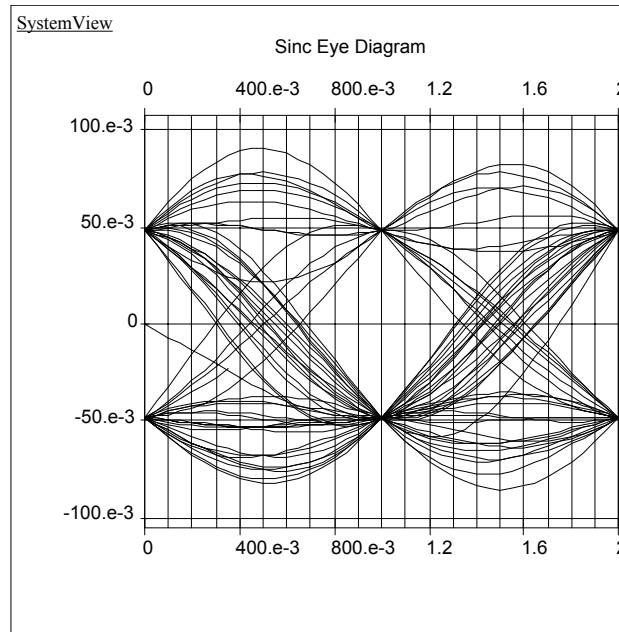


Figure 3.11: *No ISI eye diagram*

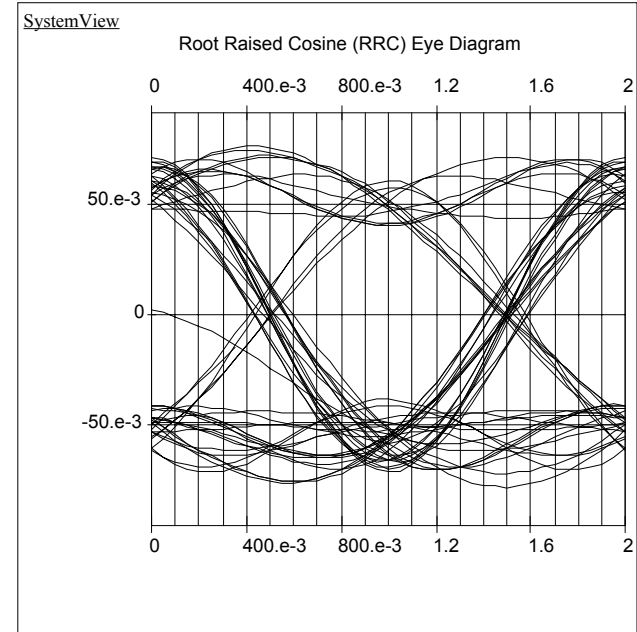


Figure 3.12: *ISI eye diagram*

So far the issue of pulse shaping has been limited to generating the signal. Nothing has been said regarding the optimum processing of this signal at the receiver end. We need a filter that is matched to the pulse shaping filter. In terms of noise only, the optimum receiver filter is identical to the filter used in the transmitter.

Now reconsider the ISI issue. The question is; 'What are the ISI properties of the

waveform after recovery of the receiver with a matched filter, regardless of the ISI properties at the transmit side?'

Open the system:

`Digital Comm\ch_03\raised-cosine2.svu`

The sinc and raised cosine filter are described in the textbook. It was stated that the raised cosine filter exhibits no ISI and is easier to work with than the theoretically optimum sinc filter.

- Run the exercise file. Go to the analysis window where several eye diagrams are plotted. Observe the one labeled RC*RC (the * symbol indicates convolution). This is the eye diagram of the output of the raised cosine matched filter. Does this signal exhibit ISI?
- A common approach used in many systems is to use a root raised cosine (RRC) filter. A RRC filter is essentially half of a raised cosine filter with one half placed in the transmitter and the other half in the receiver. The matched filter properties are still preserved. What do you expect the ISI at the output of the RRC matched filter to be? Verify your answer by observing the plot labelled RRC*RRC.
- Theoretically the plot sinc*sinc should have no ISI but it does. Explain this result.
- It is instructive to fill in the table below with a yes or no.

Filter Type	ISI at transmitter	ISI at receiver
sinc		
raised cosine		
root raised cosine		

Exercise 3.17 Downsampling a Channel

Open the system:

`Digital Comm\ch_03\simple_channel.svu`

This system has a simple channel. The simulation rate is set to 80,000 Hz. The symbol rate for this channel is 10,000 symbols/sec (10,000 baud).

(a) Run the system.

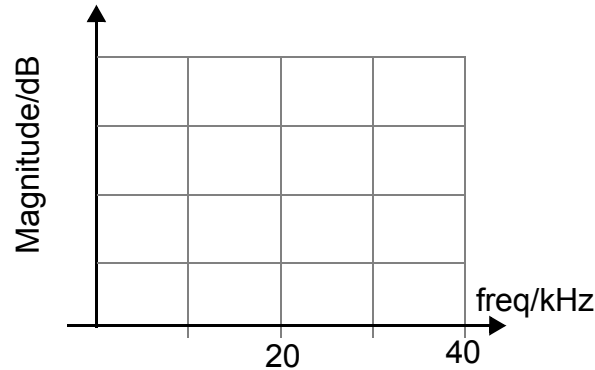
What is the time duration of the impulse response, ignoring the initial channel bulk delay?

(b) By observing the impulse response in the analysis window, set the delay to a value that synchronises the sampler to sample at the “correct” (peak value) time.

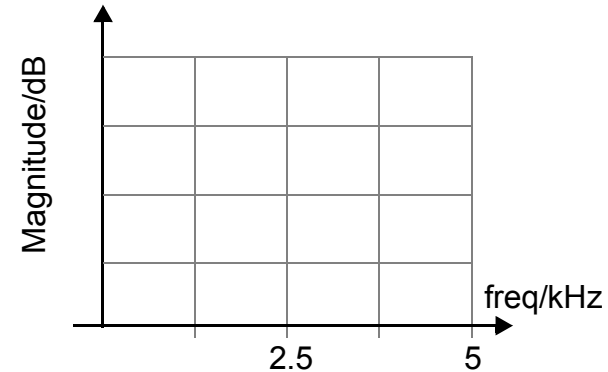
Synchronizing delay value:

Therefore, noting that the data rate is one symbol (a -1 or 1) every 1/10000th of a second, will there be intersymbol interference at the receiver?

- (c) In the analysis window sketch the magnitude frequency responses of the channels in the figures below. (Ensure you have rerun the simulation with the correct synchronising delay):



Channel (80kHz simulation rate)



Channel downsampled to 10kHz

One of the effects of the downsampling is to “alias” the channel frequency response or essentially fold the frequency response and thus produce a “flatter” response as illustrated in [Figure 3.13](#). Clearly the sampler at the receiver must sample at the correct instant to achieve this, and thus a synchronisation is required.

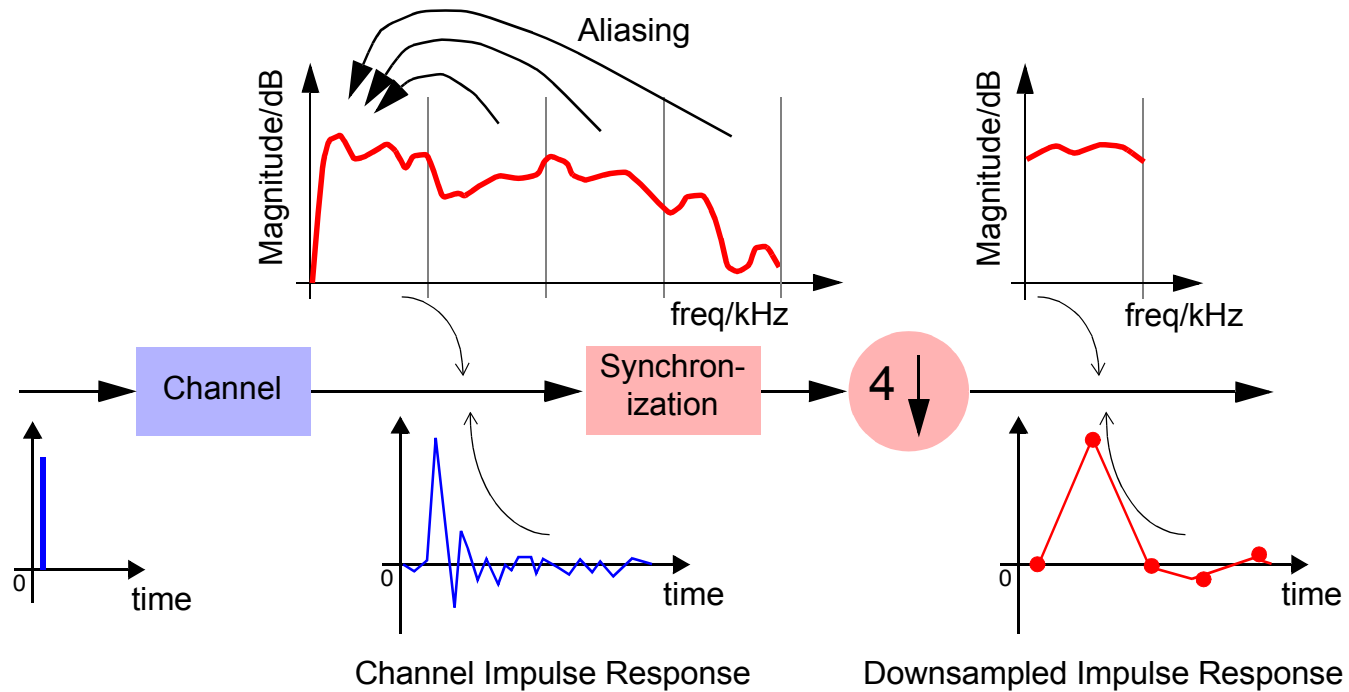


Figure 3.13: *Downsampling a channel impulse response.* Note the effect of aliasing and the “flatter” response of the downsampled channel. Note that the synchronising delay is important, and given the non-linear process of downsampling, there are a number of downsampled channels that can be produced.

Exercise 3.18 Simple Intersymbol Interference

Open the system:

`Digital Comm\ch_03\channel_isi.svu`

This system uses the same channel as Exercise 3.17. The correct synchronising delay (of value 44) has been included.

Run the system and note at the outputs sampled at the simulation rate and at the symbol rate of 10,000 symbols/sec there is ISI present.

Exercise 3.19 Signal Detection

Open the system:

`Digital Comm\ch_03\detection.svu`

Figure 12.2 in the textbook, shows a simple diagram of an energy detector or radiometer. The basic idea is to determine the size of the signal envelope R , and compare it with a threshold Th . If $R > Th$ a detection is declared. To make the radiometer more sensitive, Th should be made as small as possible. But this tactic ignores the case when there is no signal (just noise). As Th becomes smaller, the probability that noise alone will trip the detector increases. This is called a false alarm. Thus the problem becomes a trade-off between missed detection versus false alarm.

The analysis of the radiometer is based on standard I and Q demodulation techniques. There are several steps required to get to the final result. Let's take them one at a time. Assume the received signal is given by

$$r(t) = A \sin(2\pi f_c t + \theta) + n(t) \quad (3.5)$$

where $n(t)$ is a Gaussian random variable (GRV) with 0 mean and variance σ . The inphase I and quadrature Q components of the signal are given by

$$I = \int_0^T r(t) \cos(2\pi f_c t) dt \quad (3.6)$$

$$Q = \int_0^T r(t) \sin(2\pi f_c t) dt \quad (3.7)$$

where T is the symbol interval. Taking $\theta = 0$ without loss of generality

$$I = AT_i + N_i \quad (3.8)$$

$$Q = N_Q \quad (3.9)$$

where

$$N_I = \int_0^T n(t) \cos(2\pi f_c t) dt \quad (3.10)$$

$$N_Q = \int_0^T n(t) \sin(2\pi f_c t) dt \quad (3.11)$$

(a) Considering that $n(t)$ is Gaussian, show that

$$E(N_I) = E(N_Q) = E(N_I N_Q) = 0 \quad (3.12)$$

$$E(N_I^2) = E(N_Q^2) = \sigma^2 T/2 \quad (3.13)$$

We can now write the pdf of the two dimensional Gaussian distribution of I and Q

$$p(I, Q) = \frac{1}{2\pi\sigma^2} e^{-\frac{(I - AT_i)^2 + Q^2}{2\sigma^2}} \quad (3.14)$$

(b) The radiometer compares the envelope of the signal

$$r = \sqrt{I^2 + Q^2} \quad (3.15)$$

with a threshold Th . Show the pdf of the random variable r is given by

$$p(r) = \int_0^{2\pi} p(I = r \cos \theta, Q = r \sin \theta) r d\theta \quad (3.16)$$

$$= \frac{r}{\sigma^2} I_0 \left(r \frac{A}{\sigma^2} \right) e^{-\frac{(r^2 + A^2)}{2\sigma^2}} \quad (3.17)$$

where

$$I_0(z) = \frac{1}{2\pi} \int_0^{2\pi} e^{z \cos \theta} d\theta \quad (3.18)$$

is the modified Bessel function of zero order.

- (c) The distribution function above is called the Rician pdf. In the special case, $A = 0$ show that this pdf reduces to the simpler form,

$$p(r, A = 0) = \frac{r}{\sigma^2} e^{-\frac{r^2}{2\sigma^2}} \quad (3.19)$$

called the Rayleigh pdf.

- (d) The radiometer sets a threshold Th . Show that the probability of detection p_d and probability of a false alarm p_{fa} are given by

$$p_d = \int_{Th}^{\infty} p(r) dr \quad (3.20)$$

$$p_{fa} = \int_{Th}^{\infty} p(r, A = 0) dr \quad (3.21)$$

(e) Finally, run the associated SystemView example **detection.svu**. In this exercise, the signal of interest which we attempt to detect is represented by a random signal with Rician pdf and $A = 2$, while the noise which might trigger a false alarm is a random signal with Rayleigh pdf. Observe the overlay of the p_d and p_{fa} . From this graph, explain the nature of the p_d and p_{fa} curves as a function of Th . There are 3 parameters that you can control: 1) the threshold Th , 2) the amplitude A , and 3) the integration time T_0 (average time). Rerun the examples several times varying the A , and get a feel for the sensitivity of the detection and false alarms to these parameters.

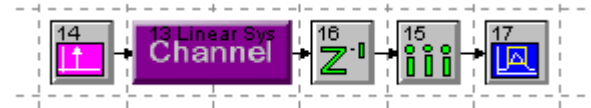
Find the region of threshold values that are optimum for large p_d and small p_{fa}

Exercise 3.20 Adaptive Channel Equalisation

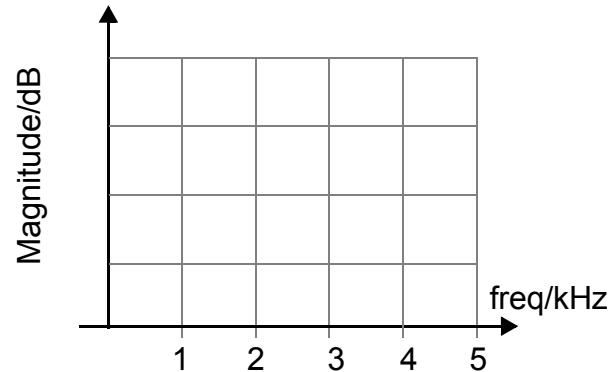
Open the system:

`Digital Comm\ch_03\lms_equalize.svu`

- (a) Run the system and in the analysis window note the impulse response and magnitude frequency response of the channel at the symbol rate as obtained from the tokens at the bottom of the design space (ignore the LMS token at this point.):



Sketch the downsampled channel magnitude frequency response below:



Channel Downsampled to 10kHz Magnitude Response

- (b) Increase the number of samples in the simulation to 16384 and run the system again.

Does the error output of the LMS converge to almost zero?

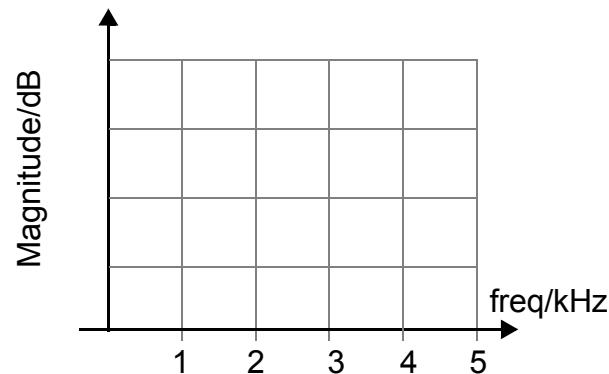
- (c) Via the LINEARSYS/FILTER token in the lower right corner of the design space, view the adapted weight values from file

`Digital Comm\External Files\General\lms_equalise.txt`

Also view the magnitude frequency response via the LINEARSYS/FILTER token



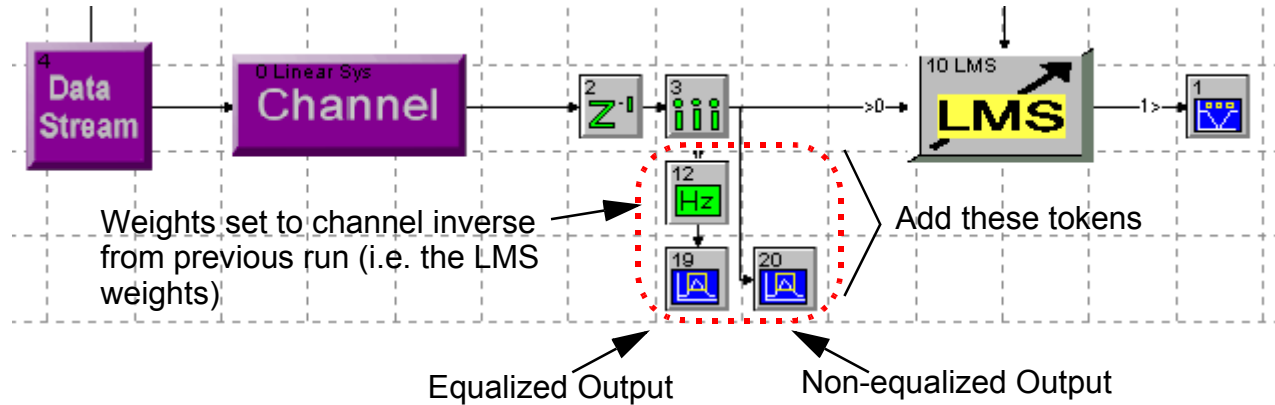
button, and sketch below:



Equalizer Magnitude Response

You should be able to confirm that the LMS equalising filter does indeed provide an inverse of the channel at the symbol rate.

- (d) Modify the system as indicated below such that you can view the non-equalized symbols output (with ISI) and the equalized symbols (with reduced ISI). As shown place the weights of the equalising filter calculated in the previous run of the simulation at the output of the symbol rate channel.



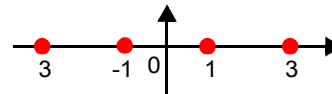
Reduce the number of samples to say, 1024 and run the system. View the equalized symbols in the analysis window. Can you again confirm that you have equalized the channel and therefore reduced the ISI?

Exercise 3.21 Adaptive Channel Equalisation with ASK data

Open the system:

`Digital Comm\ch_03\lms_equalize_4-ASK.svu`

This example uses 4 level data with the constellation:



Symbol Constellation

Run the system. Now, read in the adapted weights (you should note that the error has gone to almost zero) LINEARSYS/FILTER token and run again thus producing two outputs, the equalized and the non-equalized outputs.

Confirm that the equalisation has been successful and the 4-level output data is now available.

Exercise 3.22 Downsampled Channel

Open the system:

`Digital Comm\ch_03\simple_channel2.svu`

Run the system.

In the analysis window, normalize the downsampled and synchronized version of the channel using and , and thus confirm why we might call this channel a:

{1, 0.186, -0.053, -0.088, -0.025} channel @ symbol rate 10 KHz.

Exercise 3.23 Data Equalisation Training Mode

Open the system:

`Digital Comm\ch_03\data_equalize.svu`

This system is set up to operate in a training mode whereby both ends generate a broadband signal using a maximal length pseudo random binary sequence (PRBS). For this example we assume that there are only two symbols, $\{1, -1\}$. This type of training mode is in fact what a modem for telephone channel data communications will use in order to inverse identify the channel prior to sending data.

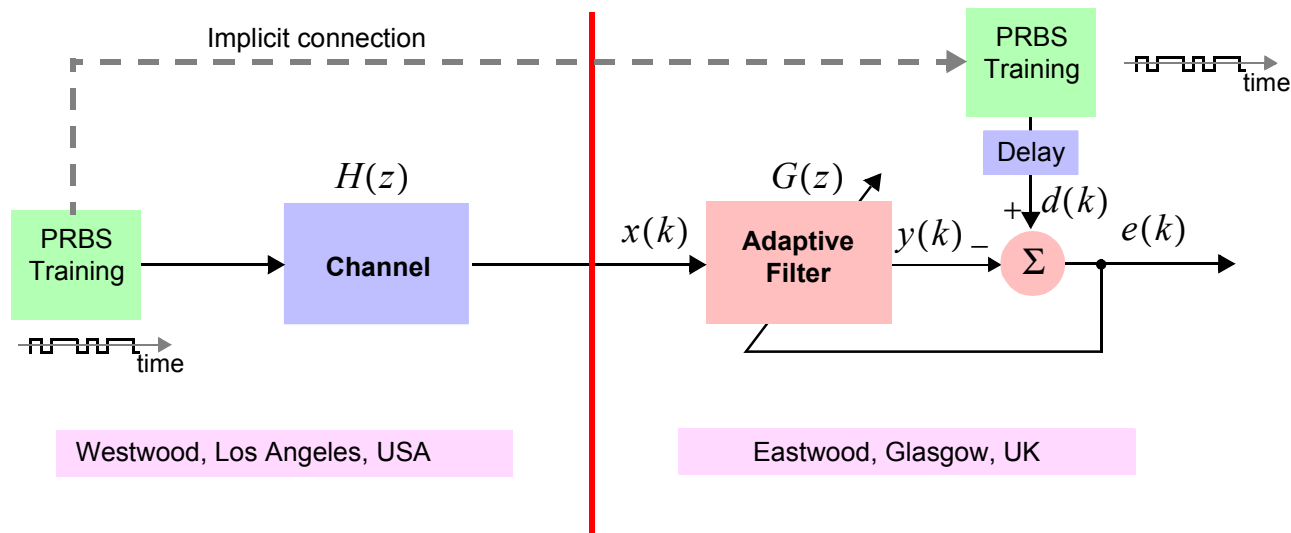


Figure 3.14: Training mode of adaptive equalizer.

Run the system, and note that the error does indeed adapt to zero. Note the

downsampled channel impulse response is non-minimum phase and has a delay equivalent to 4 samples and is a $\{0,0,0,1,0.5\}$ channel:

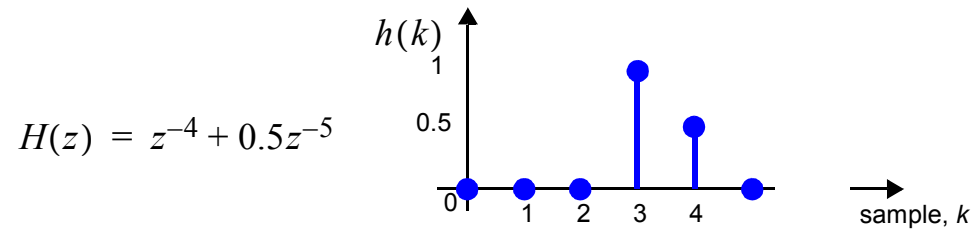


Figure 3.15: *Non-minimum phase impulse response.*

Hence, at the receiving end a delay has been inserted between the desired input and the training PRBS generator.

Exercise 3.24 Data Equalisation Decision Directed Mode

Open the system:

`Digital Comm\ch_03\data_equalize_ddm.svu`

This system is set up to operate in a decision directed mode (DDM). In this example we assume that data is being transmitted through the channel but the adaptive filter has **NOT** been trained beforehand. Hence we will attempt to perform blind equalisation.

First we make a couple of assumptions. Let us assume that the distortion introduced is not “too bad”! Second, we will use our knowledge that the data samples being transmitted are either +1 and -1. At the adaptive filter receiver we do not have a training sequence as in the previous example, however let us use the signal at the adaptive filter output, $y(k)$, as a training sequence by passing through a limiter, decision device or “ slicer”. The slicer outputs +1 or -1 which correspond to the transmitted symbols of 1 and -1:

$$\text{Slicer } [y(k)] = \begin{cases} 1, & y(k) > 0 \\ -1, & y(k) < 0 \end{cases}$$

If the channel distortion allows the data to be interpreted as *correct* more often than *wrong* (i.e. a value > 0 becomes a +1 and < 0 becomes a -1) then we should be able to

adapt in the mean. Hence the signal flow graph for our “blind” adaptive equalizer is:

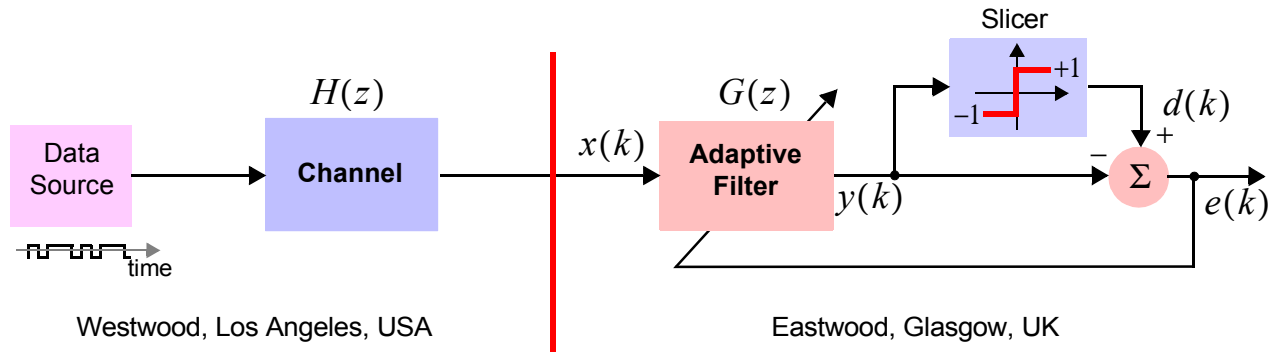


Figure 3.16: *Decision directed mode of adaptive data equalizer.*

- (a) Run the system, and note that the error does indeed adapt to zero. Note that even if the channel is non-minimum phase (or has a delay as in our channel) this is taken care of because our “desired signal”, $d(k)$ is derived from $y(k)$ which has the channel delay. Note also that the LMS token had initial conditions set up such that the first tap (or weight) of the filter was set to 1. Were this not done, the output of the adaptive filter would always be zero and the system may have difficulty “starting”.

Exercise 3.25 Data Equalisation with ASK Data

Open the system:

Digital Comm\ch_03\data_equalize_ddm4-ASK.svu

This system uses data symbols with 4 levels and hence four points in the constellation.

The “slicer” or decision device is this time a little more complex as it must make a decision as to which one of four levels was sent rather than the 2 levels as in the previous system. This 4 level decision is done by the following token connection:

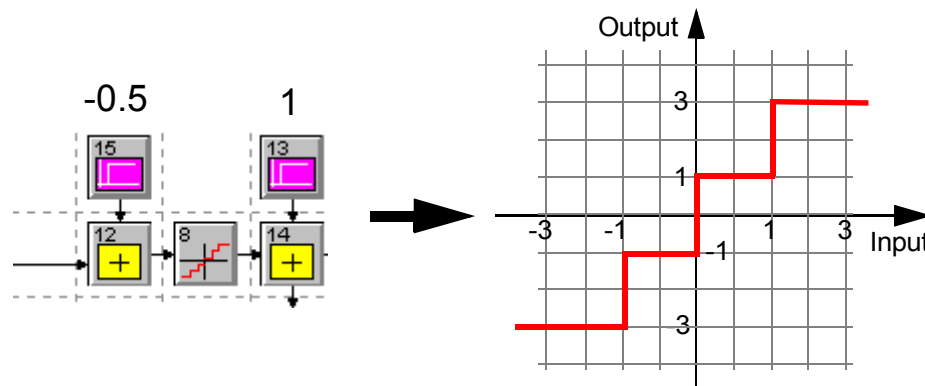
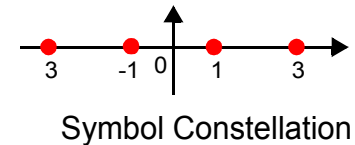


Figure 3.17: Four level slicer.

- Run the system and observe how it adapts.
- Modify the channel from a $\{1, 0.15\}$ to a $\{1, 0.5\}$. Note that the blind equalisation fails in this case.

Exercise 3.26 FeedForward Decision Feedback Equalizer FDFE

Open the system:

`Digital Comm\ch_03\ff_dfe.svu`

Run the system and confirm the ability to equalize the channel. Note the channel impulse response. Try changing to something a little more involved. Note that this system is a feedforward - decision feedback equalizer (FF-DFE).

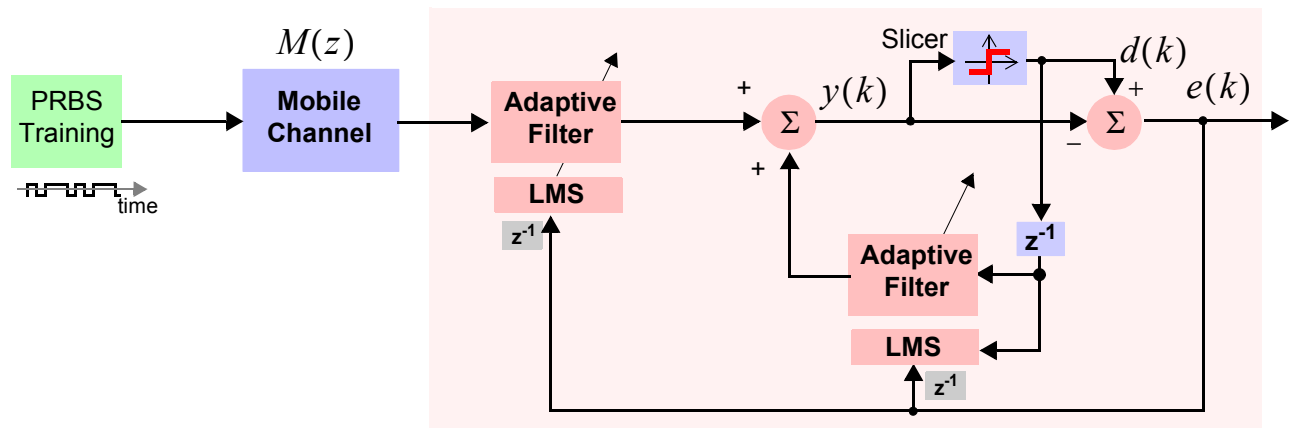


Figure 3.18: DDM mode Feedforward/decision feedback equalizer (FF-DFE).


4 Bandpass Modulation and Demodulation/Detection

Exercise 4.1 Basic Bandpass Modulation

Open the system:

`Digital Comm\ch_04\amp_mod_dsbsc.svu`

This system shows simple double-sideband suppressed carrier (DSB-SC) amplitude modulation.

- (a) Run the system and note the double sideband output.
- (b) Change the input baseband signal to a frequency sweep of 100Hz to 2000Hz over a period of 10msecs. Run the system again, and note the DSB-SC in the  window.

Exercise 4.2 Basic Quadrature Modulation

Open the system:

`Digital Comm\ch_04\gam_two_signals.svu`

This system multiplexes two signals onto a single carrier, and then demodulates, and low pass filters to produce the two signals at the receiver.

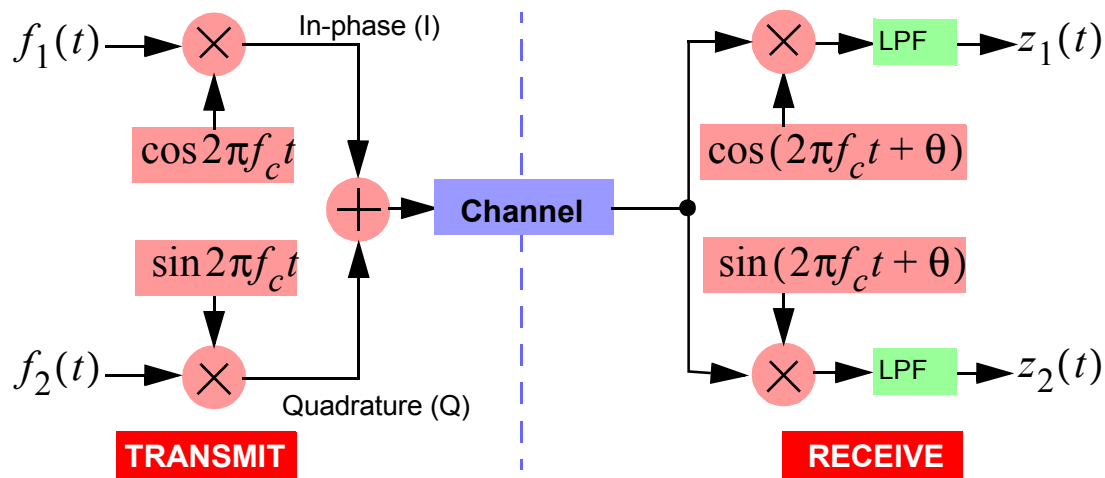


Figure 4.1: Quadrature modulation/demodulation.

- Run the system and note that the two inputs can be recovered at the receiver output.
- Put a small phase error of 4 degrees in the local oscillator at the receiver. What is the effect on the output?

- (c) Try adding some (Gaussian) channel noise and run the system again.
- (d) Open the system:

`Digital Comm\ch_03\gam_two_signals2.svu`

- (e) This system performs exactly the same as the previous system, except we are now using two tokens from the Communication Library. Identify the *Qad_Mod* token at the modulation side, and the token *IQ-mixer* at the receiver and check the parameters.
- (f) Run the system and confirm that the output is the same as before.
- (g) For the IQ-mixer, set the IQ rotate parameter to 4 degrees and confirm the effect is the same as putting a phase shift on the carrier as done above.

Hence we note that SystemView has two “convenient” tokens for quadrature modulation and demodulation.

Exercise 4.3 8 Phase Shift Keying (PSK)

Open the system:

[Digital Comm\ch_04\8psk.svu](#)

This exercise explores several features of 8-PSK modulation and detection. It also uses a BER token from the Communications Library, which was designed for making bit-error-rate measurements. To understand the operation of this token, click the F1 button on your keyboard. This brings up all of the SystemView manuals (using Acrobat Reader). Under Optional Libraries, click Communications, and scroll to page 152 (Appendix A).

- (a) Review the operation of the tokens in the exercise file to see how the modulator and demodulator accomplish the desired task. Write down a short description of each token in terms of what it is accomplishing in the overall simulation scheme.
- (b) Run the exercise file. What is the BER indicated by the final value sink?
- (c) Go to the analysis window. There are two I/Q signal constellation plots. One plot is for the input, and the other is at the output before the phase demodulator token. Explain the nature of the output I/Q plot, especially in terms of the BER value previously noted.
- (d) Run the exercise file several times. Each time increase the value of the noise source. Observe the BER value reported and discuss this value in terms of the output I/Q plot.

Exercise 4.4 8 Phase Shift Keying (PSK) Bit Error Rate Exercise

Open the system:

`Digital Comm\ch_04\8psk-ber.svu`

This example implements a complete end-to-end BER simulation of a coherent 8-PSK system. It is advised that you complete the [Exercise 4.3](#) before this one in order to understand the operations of the various elements of the simulation.

- (a) Compare this exercise file with the exercise file [Exercise 4.3](#). Note that we have added two tokens, a Gray encoder and a Gray decoder.
- (b) What is the purpose of the Gray coding as related to the system performance?
- (c) Run the simulation. Go to the analysis window to see the BER curve. Compare the values from the simulation with the theoretical results presented in Chapter 4.

Exercise 4.5 Design of a QPSK System

In the sequence of systems below, we will begin by setting up a matched filter QPSK link. Open each system in turn and observe the increasing complexity,

`Digital Comm\ch_04\data_generate.svu`

`Digital Comm\ch_04\raised_cosine.svu`

`Digital Comm\ch_04\root_raised_cosine.svu`

For the next system, use the SystemView probe (see [Getting Started](#) on use of the probe) with ChA on the X-axis and ChB on the Y-axis to produce a constellation diagram.

`Digital Comm\ch_04\qpsk_system.svu`

Change the phase of the carrier frequency and observe the rotation of the constellation. How could be correct for this rotation?

Try adding a slight offset to the carrier frequency (say 100Hz error). What happens to the constellation now?

Note that when you add noise, we see the constellation spreading:

`Digital Comm\ch_04\qpsk_system_noise.svu`

With the noise set to a “low” level, change the input signal from a 2-level to a 4-level input, observe the constellation and comment on your 16-QAM output.

Exercise 4.6 Complex FIR QAM Channel Modelling

In the sequence of systems below, we will produce a complex baseband FIR filter that is a (linear) model of QPSK or QAM link using an adaptive complex LMS system identification technique. In the first system we produce a simple multipath channel, and thereafter model this with a baseband filter.

Run the sequence of examples in order to model the real channel.

```
Digital Comm\ch_04\qpsk_system_multipath.svu
```

```
Digital Comm\ch_04\qpsk_system_impulse.svu
```

```
Digital Comm\ch_04\qpsk_system_clms.svu
```

```
Digital Comm\ch_04\qpsk_system_complex_model.svu
```

5 Communications Link Analysis


Exercise 5.1 Automatic Gain Control

Open the system:

`Digital Comm\ch_05\auto-gain-ctrl.svu`

The automatic gain control (AGC) circuit is a common element of a receiver. It is used to control the dynamic range of the signal as it is processed by the receiver components. There are many forms of AGC circuits.

This example shows one possible implementation. A square law detector determines the average power of the received signal. This result is compared to a desired result. The difference between the two signals is used to control the gain of an amplifier.

- (a) Run the system; note from the define system time  that three different loops with different input powers are shown. What should the output AGC power be in each case? Verify your answer.
- (b) Vary the gain of token 7 and observe its effect on the loop operation. Observe the loop response time as a function of the input amplitude. Explain the result.

Exercise 5.2 Noise Figure Problem

Open the system:

`Digital Comm\ch_05\noise-figure.svu`

You want to connect the antenna of your satellite dish to your TV receiver with a cable that has a loss of 10 dB. You buy an amplifier having a 10 dB gain and a 3 dB noise figure.

Problem: Where do you place the amplifier, at the antenna-end or the receiver-end of the cable?

- (a) The SystemView example shows an input signal and noise with the two configurations running in parallel. Calculate the signal and noise levels after each element of the system (refer to the equations in Chapter 5). What is the overall noise figure of each system?
- (b) Run the SystemView example and verify your calculations with the displayed spectral plots in the analysis window.
- (c) Write a short paragraph which gives a *physical* description as to why the two results are different.

6 Channel Coding: Part 1

Exercise 6.1 Signal Spectra

Open the system:

`Digital Comm\ch_06\signal-spectra.svu`

Consider a binary communication with the following signal assignment

$$\begin{aligned} \text{data} = 1, \text{ send } s_1(t) &= A, 0 \leq t \leq T \\ \text{data} = 0, \text{ send } s_2(t) &= \begin{cases} A, 0 \leq t \leq T/2 \\ -A, T/2 \leq t \leq T \end{cases} \end{aligned} \quad (6.1)$$

And T is the data period or inverse of the data rate R .

- (a) Show that the two signals are orthogonal over T sec.
- (b) For *any* arbitrary input data sequence examine the resulting data stream for the time increments:

$$0 \leq t \leq T/2$$

$$T \leq t \leq 3T/2$$

...

$$kT \leq t \leq (kT + 1)/2$$

what do you conclude about the signal in these time periods?

- (c) This exercise implements an orthogonal signalling system as described in Eq. 6.1 and Eq. 6.1. Signals $s_1(t)$ and $s_2(t)$ are generated by tokens 1 and 0 respectively. An antipodal signal is used as control signal of a switch, if it takes a logic 1 value, $s_1(t)$ is chosen, if it takes a logic 0 value, $s_2(t)$ is chosen. Verify your answers for (a) and (b) via the SystemView exercise.
- (d) Note that the binary communications system specified in Eq. 6.1 and Eq. 6.1 can also be represented by a sequence $s(t = kT)$ which is the sum of two terms as specified in the following equation

$$s(kT) = \sum_{k=-\infty}^{\infty} p(t - kT) + \sum_{k=-\infty}^{\infty} a_k p(t - kT - T/2) \quad (6.2)$$

where

$$p(t) = \begin{cases} 1, & 0 \leq t \leq T/2 \\ 0, & \text{otherwise} \end{cases} \quad (6.3)$$

and

$$a_k = \pm 1 \quad (6.4)$$

- (e) Take the Fourier integral of the Eq. 6.2 and analytically show that the spectra is comprised of two distinct parts: 1) a series of discrete lines of rate k/T arising from the first summation of $s(kT)$, and 2) a continuous part from the second term. Verify the answer via the accompanying SystemView example.
- (f) Of the total average power in the signal, what is the ratio of power in the continuous portion of the signal to the power in the line spectra?
- (g) What information is carried by line spectra that are on for all time?
- (h) Finally, what can you conclude about another view of why an orthogonal signal set performs 3 dB worse than an antipodal set, even when both signals have the same energy per bit?

Exercise 6.2 Golay Error Correction Code

Open the system:

`Digital Comm\ch_06\golay-correction.svu`

This exercise uses a BER token from the Communications Library, which was designed for making bit-error-rate measurements. To understand the operation of this token, click the F1 button on your keyboard. This brings up all of the SystemView manuals (using Acrobat Reader). Under Optional Libraries, click Communications, and scroll to page 152 (Appendix A).

The Golay block error correcting code has the parameters:

$$(n = 23, k = 12, t = 3) \quad (6.5)$$

- (a) Note in this system that there are two BER tokens, 7 and 10. Review the parameters of these tokens and their outputs to the sinks. What is being measured in each case?
- (b) Run the simulation and go to the analysis screen. Here you will see the plots of channel block errors, decoded block errors, and an overlay of the two. Review the results.
- (c) If the number of errors/block at the input is 3 or less, how many errors are in the decoded block? Verify your answer from the plot.

Exercise 6.3 Cyclic Redundancy Coding

Open the system:

`Digital Comm\ch_06\crc_encoder.svu`

This circuit performs (7,4) cyclic coding (7 bits are output for every 4 input). This coding scheme thus transmits the original 4 data bits plus an additional 3 parity bits. The circuit effectively generates the remainder of a polynomial division. During the first 4 shifts switch 1 allows the data to be loaded into the shift register and switch 2 directs the data to the output of the circuit. After four shifts the parity bits occupy the shift register. Switch 1 is then set to the open position so that the shift register is cleared while switch 2 directs the shift register output (parity bits) to the output of the circuit.

$$g(X) = 1 + X + X^3$$

Run the system and confirm its correct operation.

Exercise 6.4 Cyclic Redundancy Decoding

Open the system:

`Digital Comm\ch_06\crc_deccoder.svu`

This circuit performs a "syndrome" check on a received CRC coded data stream. The seven bits of the received codeword are shifted into the register with the feedback enabled (switch 1 closed). When all seven bits have been shifted then the result of the syndrome test will be stored in the shift register. These are then retrieved by disabling the feedback and closing switch 2. If the result of the syndrome test is not all zeros then an error has occurred.

- (a) Run the system and observe the output of the syndrome generator to verify that the input sequence to the channel has no errors.
- (b) Set the impulse token amplitude to 1, such that the 3rd current bit is inverted and rendered erroneous. Run the system to see if the syndrome test detects the error. Change the time of the impulse to see if it still works.

Exercise 6.5 Golay Codes

Open and run the sequence of systems:

`Digital Comm\ch_06\Golay_codes_1.svu`

`Digital Comm\ch_06\Golay_codes_2.svu`


and observe the error detecting capabilities of Golay codes.

Exercise 6.6 BCH-Coded Baseband System - BER Exercise

Open the system:

`Digital Comm\ch_06\BCH-chan-BER.svu`

In Chapter 6 of the textbook, block error correcting codes were described in terms of the BER input to the decoder (channel BER) and the output BER from the decoder, or system BER (see Figure 6.21 of the textbook). This example provides a simulation to illustrate this concept.

- (a) In the example file verify the simulation and the nature of the two BER curves generated.
- (b) Run the example and go to the analysis  window. The plot is similar to Figure 6.21 of the textbook. In SystemView, the abscissa values are plotted from low to high, while in the textbook they are plotted from high to low. Thus the orientation of the SystemView curve is "flipped" compared to the one in the textbook.
- (c) Equation (6.4.6) in the textbook relates the channel-bit error probability p to the decoded-bit error probability P_B . Substitute the simulation channel values for p into Equation (6.46) and compare the result with the actual simulation result.

Exercise 6.7 BCH BER Example

Open the system:

`Digital Comm\ch_06\BCH-BER-plot.svu`

This file measures the BER of a (63,30,6) block error-correcting code.

- (a) Run the example file, and go to the sink calculator to see the BER plot. Note that the BER for binary antipodal signalling is overlaid with the BCH BER curve. Binary antipodal signalling as was shown in the textbook is the theoretically best performing uncoded binary system. It can be used as a benchmark for comparing the performance of other systems.
- (b) Look at the nature of the two BER curves. In particular, when $E_b/N_0 < 4$ dB, which scheme, the coded or the uncoded, has the better BER performance?
- (c) Does correcting codes always give performance improvement?

Exercise 6.8 BCH Exercise: BER versus Code Rate

Open the system:

`Digital Comm\ch_06\BCH-code-rate.svu`

In the textbook, an important trade-off regarding error correcting codes was discussed. Basically, there are two competing forces which effect the overall bit-error rate. First, the error-correcting capability of the code improves with greater redundancy (decreasing code rate). Second, for a real-time communication system, greater redundancy results in less energy per code bit, and hence in a larger channel-bit error rate. As the channel-bit error rate increases, so does the bit error rate. As a general result the optimum code rate is about 1/2 to 1/3 (In the textbook, see Chapter 8, Figure 8.6, and Chapter 9, Example 9.4).

In the SystemView example there are three independent BCH encoded systems; (1) [63,57,1], (2) [63,30,6], and (3) [63,7,15]. What is the rate of each of these codes?

Run the example. Go to the analysis window and observe the BER curves. Which code has the best performance?

7 Channel Coding: Part 2

Exercise 7.1 Hard Vs. Soft Decision

Open the system:

`Digital Comm\ch_07\hard-vs-soft.svu`

The detection technique used here can be termed a repeat code. At the demodulator/detector, segments of each bit (11 in this exercise) can be considered to be code bits. Hard decisions are made on each of the code bits, and majority logic then determines the final bit decision. This problem explores the loss of system performance when using a hard decision error correcting code algorithm versus a soft decision algorithm as discussed in Chapter 7, Section 7.3.2 of the textbook. The optimum soft decoder will correlate the received code word with all possible code words and select the one with the closest match. This procedure becomes impractical for large codes. For example, how many code words are there for the BCH code [15, 11, 1]? The alternative is to make a hard decision on each received code bit (± 1), and employ a simpler but non optimum decoding algorithm. The 'price' that is paid for this simplification is a loss of 2.0-2.5 dB, that is, it takes about 2.0 dB more power to achieve the same BER for a hard decision as it does for a soft decision.

It is difficult to modify existing algorithms used for block codes to accommodate soft decisions. However, it is relatively easy to do so for convolutional codes, which is one of the reasons for their popularity.

Consider a simple binary communication system where we apply the following simple code:

$$\text{ones} = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\} \text{ (eleven 1's)} \quad (7.1)$$

$$\text{zeros} = \{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\} \text{ (eleven 0's)} \quad (7.2)$$

- (a) What is the rate of this code?
- (b) What is the Hamming distance of this code?
- (c) How many errors can it correct?
- (d) Run the system, noting that there are two simulations running side by side. Describe the nature of the simulations, especially their difference. What is the effect of the limiter token on the simulation results for its loop?
- (e) Show that the BER for the soft decision path is given by $Q\sqrt{2E_b/N_0}$
- (f) Show that the BER for the hard decision path is given by

$$c_6^{11} q^6 p^5 \quad (7.3)$$

$$q = Q(\sqrt{2E_b/11N_0}) \quad p = 1 - q \text{ and } c_6^{11} \text{ binomial coefficient}$$

Use the asymptotic form $Q(\sqrt{x}) \approx e^{-x}$ to show that the hard decision performs about 2.5 dB worse in this case.


- (g) Run the simulation and verify the answer you obtained.

Exercise 7.2 Convolutional Coder

Open the system:

`Digital Comm\ch_07\simple_coder.svu`

In this example a convolutional coder is implemented using discrete components, and using SystemView's standard convolutional coder token.

- (a) For the input sequence 1 0 1 0 0 0 predict what the output will be via “hand” calculation, either from the circuit representation or using a state diagram. Assume that the coders start in state 00.
- (b) Before running the system, note carefully the sampling rate  is set to 19.2 Hz, and the data is input at a rate of 9.6 kbits after the downsampler. Hence the coder(s) take in data at 9.6 kbits/s and outputs at 19.2 kbits/s. Observe the token parameters to confirm this is correct.
- (c) Run the system and observe the coded outputs for the given input.

Note that the standard convolutional coder token has a latency of two bits compared to the discrete implementation; this is simply a function of the token design. Also note that the coder “polynomials” are conveniently represented by octal values, i.e., $\{7_8, 5_8\} = \{111, 101\}$.

Exercise 7.3 Convolutional Coding and Decoding

Open the system:

`Digital Comm\ch_07\conv_coder_decoder.svu`

In this example the same 7,5 coder as in [Exercise 7.2](#) is used, and the convolutional decoder, or Viterbi decoder has a path length of 15.

(a) Run the system and confirm that the sequence is correctly decoded at the output.

Notice there is a latency of 17 bits. Can you account for this?

(b) Run the system and observe the coded outputs for the given input. Note that the standard convolutional coder token has a latency of two bits compared to the discrete implementation; this is simply a function of the token design.

8 Channel Coding: Part 3

Exercise 8.1 Reed-Solomon Code

Open the system:

`Digital Comm\ch_08\reed-solomon.svu`

This exercise shows the BER performance of a [15,11,2] Reed-Solomon error correcting code. The BER token from the Communications Library is used in this exercise, and was designed for making bit-error-rate measurements. To understand the operation of this token, click the F1 button on your keyboard. This brings up all of the SystemView manuals (using Acrobat Reader). Under Optional Libraries, click Communications, and scroll to page 152 (Appendix A).

- (a) How many input bits are needed to make a symbol for the encoder? The modulation is baseband antipodal signalling. Therefore the encoded symbols must be converted back to bits.
- (b) Review the simulation to understand the operation of each token. The noise source token is set to 0.25 W/Hz. What is the corresponding E_b/N_0 ?
- (c) Run the exercise file. Go to the analysis window and observe the BER curve. What do you conclude about the effectiveness of this code when the E_b/N_0 is relatively high?

Exercise 8.2 Interleaving

Open and run the sequence of systems:

```
Digital Comm\ch_08\interleaving1.svu
```

```
Digital Comm\ch_08\interleaving2.svu
```

```
Digital Comm\ch_08\interleaving3.svu
```

and observe the “error” spreading effects of interleaving, and the correction capability of the convolution coder when interleaving is present and when removed.

Exercise 8.3 Turbo Codes

Open and run the sequence of systems:

```
Digital Comm\ch_08\rsc_1.svu  
Digital Comm\ch_08\rsc_2.svu  
Digital Comm\ch_08\rsc_3.svu  
Digital Comm\ch_08\turbo_codes_1.svu  
Digital Comm\ch_08\turbo_codes_2.svu  
Digital Comm\ch_08\turbo_codes_3.svu  
Digital Comm\ch_08\turbo_codes_4.svu  
Digital Comm\ch_08\turbo_codes_5.svu
```

to view the construction and implementation of a turbo coder.

Exercise 8.4 Concatenated Code Exercise

The Viterbi decoding algorithm for convolutional codes is described in detail in Chapter 7 Section 7.3 of the textbook. The basic idea is to find the most likely path through a trellis structure. When two paths arrive at a node in the trellis, the one with a metric which is the closest agreement to the transmitted path is accepted and the other is rejected. One consequence of this algorithm is that when the wrong path is chosen, the number of bit errors can be high. Thus the errors in a Viterbi decoding algorithm appear in bursts rather than a random pattern. In Chapter 8 of the textbook, it is noted that the Reed-Solomon (R-S) codes are capable of processing burst errors.

A concatenated code is one with one error correcting code inside another:

[Digital Comm\ch_08\concatenated.svu](#)

This structure is commonly employed in wireless systems. Note that there is a convolutional code nested inside the R-S code.

- (a) There are two BER tokens (19 & 13); describe what each is measuring.
- (b) Run the example file. Note the burst error nature of the convolutional code. Compare the two BER values. Comment on their relative values.
- (c) To see an additional point on the BER curve, perform the following operations: (1) delete sink token 20, (2) open the SystemView master clock in the tool bar at the top of the design screen. Change the number of loops from 1 to 2. Increase the number of samples to 500,000, (3) run the example and observe the results. Note the dramatic improvement of BER. How does this code compare to standard uncoded antipodal signalling?

9 Modulation and Coding Trade-offs

Exercise 9.1 Minimum Shift Keying (MSK)

Open the system:

`Digital Comm\ch_09\msk.svu`

This example extends the discussion on MSK modulation given in Section 9.8.2 of the textbook. One method for producing an MSK signal is shown in Equation 9.46.

- (a) Show that this equation is equivalent to modulation of the NRZ data via a FM (VCO) modulator. What must the modulation gain be with respect to the input data rate R ?
- (b) Show that the modulated phase change over 1 data period is $\pm\pi/2$. Run the SystemView simulation and observe the phase trajectory plot in the analysis window.
- (c) As discussed in the textbook, MSK can be thought of as an FSK signal where the frequency is being shifted $\pm(1/4T)$ with phase continuity at the bit transitions. Consequently, a frequency discriminator can demodulate MSK. Since the frequency is the derivative of the phase, a simple d/dt operation after the $\arctan(\cdot)$ will recover the original data. Observe this result in the SystemView analysis window.
- (d) The $\arctan(\cdot)$ operation is difficult to implement, and is not needed. Show that if

$$\theta = \arctan[I(t)/Q(t)] \quad (9.1)$$

then, the frequency f , is given by

$$\begin{aligned}
 f &= \frac{1}{2\pi} \frac{d\theta}{dt} = \frac{1}{2\pi} \frac{d}{dt} \text{atan}[I(t)/Q(t)] \\
 &= \frac{1}{2\pi} \frac{Q \frac{dI}{dt} - I \frac{dQ}{dt}}{I^2 + Q^2}
 \end{aligned} \tag{9.2}$$

The denominator is the signal envelope that is constant and need not be calculated. Furthermore, the continuous derivative can be replaced by its numeric equivalent to obtain a simple relationship.

- (e) Note that the above result is the imaginary part of the complex conjugate product between

$$I_k + jQ_k \text{ and } I_{k-1} + jQ_{k-1}$$

where the subscript k refers to the k th sample time.

- (f) Verify this result from the simulation plot.
- (g) Finally, we note that another method of detecting MSK is to compare the phase plane trajectory with all possible combinations and choose the most likely one which has the highest cross correlation. An algorithm based on the Viterbi decoder can be used to reduce the calculation load. Such a scheme is called the Viterbi equalizer described in the textbook, Chapter 15, Section 15.7.1.

Exercise 9.2 Gaussian Minimum Shift Keying (GMSK)

Gaussian Minimum Shift Keying (GMSK) is a popular modulation format used in wireless communication systems. It is similar to standard MSK modulation except that a Gaussian shaped low-pass filter is inserted between the input data NRZ waveform and the modulating VCO.

Open the system:

`Digital Comm\ch_09\GMSK-baseband.svu`

Note that both a GMSK and an MSK modulator are implemented.

- (a) Run the example. Go to the analysis window and observe the overlay of the occupied signal spectra of the two modulation formats. Comment on the differences seen and why GMSK is preferred over MSK in a wireless environment.
- (b) The simulation also implements a simple GMSK demodulator/detector. It is a form of baseband FM discriminator. In the analysis window the 'eye diagram' of the modulation is shown. What does this diagram say about intersymbol interference?

Exercise 9.3 Offset QPSK Example

Open the system:

`Digital Comm\ch_09\offset-QPSK.svu`

The textbook describes offset QPSK in Chapter 9, Section 9.8.1. In normal QPSK the I and Q data changes simultaneously every T seconds. Thus a phase shift of 180 degrees is possible. In offset QPSK the Q channel is delayed by $T/2$ sec. Thus the signal phase can change at most by ± 90 degrees every $T/2$ seconds. The implications of this difference shows up when the signal is band limited.

- Note that two systems are running side by side: normal QPSK, and offset QPSK. Verify that the systems shown produce the correct modulation.
- Run the simulation and in the analysis window compare the two modulation types.
- In Chapter 9, Section 9.8.2 of the textbook, it is shown that the occupied power spectra of normal and offset QPSK is the same. Verify this by observing the overlay plot in the analysis window.
- However, the spectra of hard limited, filtered QPSK and Offset QPSK are quite different. Go to the sink calculator and choose 'Operators' and then select 'overlay plots'. Choose the plots 'hard limit QPSK', and 'hard limited OQPSK'. This will produce a new plot that is the overlay of the two selected plots. Now go to the sink calculator and select 'Spectrum', and select 'dBm into 50 ohms'. Select the overlay plot just generated. This will produce a new plot which is the overlay of the Spectra of the two signals. The plots are not the same now, observe the difference. Which spectra will interfere more with adjacent channels?

10 Synchronisation

Exercise 10.1 Delay Locked Loop

Open the system:

`Digital Comm\ch_10\delay-locked-loop.svu`

This is a variation of Figure 12.22 in the textbook. In this model it is assumed that the carrier frequency has been removed by some form of loop such as the Costas loop, and the resulting signal is at baseband. It is also assumed that the acquisition process has adjusted the receiver PN code to within one PN chip.

- (a) Run the system to see how the loop behaves.
- (b) Vary the loop parameters such as the initial offset, the gain of the VCO token, and the bandwidth of the low pass filter.

Exercise 10.2 Early Late Detectors

Open the sequence of systems:

Digital Comm\ch_10\early_late_symbol_sync_1.svu

Digital Comm\ch_10\early_late_symbol_sync_2.svu

Digital Comm\ch_10\early_late_symbol_sync_3.svu

Digital Comm\ch_10\early_late_symbol_sync_4.svu

and follow the on-screen instructions to set up an early late detector for a root raised cosine pulse shaped signal.

Exercise 10.3 Squaring Loop

Open the system:

`Digital Comm\ch_10\squaring_loops1.svu`

In this example, a BPSK signal is transmitted through an ideal communication channel. The BPSK signal used here is a 'suppressed carrier' signal and its carrier phase cannot be tracked by a simple phase locked loop.

One solution to this problem is to square the received signal. This produces a signal with a component at twice the carrier frequency. This component is not phase modulated like the original carrier and can be tracked by a phase locked loop.

- (a) Run the system.
- (b) Look at the input and output of the square law device, labelled 'received' and 'squared' respectively.
- (c) Note that the squared signal does not exhibit the sudden phase changes which are apparent on the received signal.
- (d) Go to the analysis window to see FFTs of the received and squared signals.
- (e) Now open the system

`Digital Comm\ch_10\squaring_loops2.svu`

which adds noise to the received signal. Follow the on-screen instructions.

Exercise 10.4 Squaring Loop and Phase Locked Loop

Open the system:

`Digital Comm\ch_10\squaring_loops3.svu`

The previous example showed that squaring a suppressed carrier signal produces a strong frequency component at twice the carrier frequency. In a squaring loop this component is tracked by a PLL. The output of the VCO in the PLL can then be used to generate a sinusoid at the carrier frequency.

- (a) Run the system and observe the “error” going to zero as the loop locks.
- (b) Change the phase of the carrier by altering the parameters of the PM token.
- (c) The SNR is currently set to 30 dBs. Reduce the SNR by altering the 'dB power' parameter of the gain token and observe the effect on the performance of the PLL. Remember that any fluctuations in the error cause a change of phase/frequency in the VCO. This is known as phase jitter.

Exercise 10.5 Phase Locked Loop (PLL)

Open the system:

`Digital Comm\ch_10\basic_pll_1.svu`

This problem explores the ideas pertinent to the operation of a PLL as described in Chapter 10 of the textbook. It is a basic loop with an input, phase detector, loop filter and VCO(FM) token.

Note that the SystemView FM token parameter 'modulation gain' μ is related to the 'gain' parameter K_0 used in the textbook via the relation:

$$K_0 = 2\pi\mu \quad (10.1)$$

For convenience, we rewrite the final error of the loop given in Equation 10.12 of the textbook as follows:

$$\lim_{t \rightarrow \infty} e(t) = \frac{\Delta\omega}{K_0 F(0)} = \frac{\Delta f}{\mu F(0)} \quad (10.2)$$

where Δf is the frequency offset in Hz and $\Delta\omega = 2\pi\Delta f$. The loop filter Fourier transform $F(\omega)$ is set to the form:

$$F(\omega) = \frac{1}{j\omega + 1} \quad (10.3)$$

- (a) The first task is to examine the tracking error when the input and VCO are at the same frequency of 10 Hz. In this case, what is Δf , and hence what is the final error?

- (b) Run the simulation to verify your answer. Take a detailed look at the error signal. Note the 'high' frequency oscillations. What is the source of these oscillations?
- (c) Now change the frequency of the input signal to 10.2 Hz. What is the steady state tracking error? Run the simulation and verify your answer.
- (d) Consider a loop filter $F(\omega)$ of the form

$$F(\omega) = 1 + \frac{1}{j\omega} = \frac{j\omega + 1}{j\omega} \quad (10.4)$$

In this case what is $F(0)$, and what is the steady state tracking error? Modify the loop filter.

Exercise 10.6 Phase Locked Loop (PLL) - Time Exercise

Open the system:

`Digital Comm\ch_10\basic_pll_2.svu`

- (a) This file is essentially the same as the one used in [Exercise 10.5](#). Open it and change the parameters of the input sinewave to a frequency of 10.8 Hz, and a new amplitude of 0.5 V. Run the simulation. Did the loop lock as it did in the previous exercise with the same input frequency?
- (b) Return to the basic derivations of a PLL starting with Equation (10.1) in the textbook. Instead of a unity amplitude, give the input sinewave an amplitude A_1 . Also in Equation (10.2) give the VCO an amplitude $2A_2$. Show that the effect of this change is to define a new constant $K_0' = A_1 A_2 K_0$.
- (c) Note that the loop dynamics depends only on the product of the individual terms. Run the example several times by varying the three terms such that the product is constant. Verify that the loop operation does not change.
- (d) What does the above say about the dynamics of a PLL with respect to the strength of the input signal?
- (e) What would you do to insure that the input power to the loop is constant?

Exercise 10.7 Costas Loop Exercise

Open the system:

`Digital Comm\ch_10\Costas-loop.svu`

The Costas loop is described in Figure 10.6 of the textbook, and is detailed in the SystemView file.

(a) Show that the signal in the upper or I arm of the loop is given by

$$I = a_k \cos \theta \quad (10.5)$$

where $a_k = \pm 1$ is the k -th data bit and θ is the unknown phase error.

(b) Similarly show that the signal in the Q arm is

$$Q = a_k \sin \theta \quad (10.6)$$

(c) The error signal is $e = IQ$. Show that $e = \sin(2\theta)$, independent of the data.

(d) For the values $0 \leq \theta \leq 2\pi$, at what value of θ does the error signal have a value of zero $e = 0$?

(e) Plot the parameter, e , as a function of θ . Look at the slope of this function at $(0, \pi)$ and at $(\pi/2, 3\pi/2)$. It can be shown that for the latter case, any minor perturbation will cause the loop to slip until one of the stable points $(0, \pi)$ is reached. The loop can then lock at the true value, or π radians out of phase. Unless a known reference

is sent, there is no way of distinguishing which occurs. For phase-based modulation schemes, what is the effect on the value of the demodulated data if the loop locks to π instead of 0?

- (f) Now open the SystemView example and run the simulation to get a feeling of the operation of the loop.
- (g) How would you encode the data so that this ambiguity does not effect the correct decision?

Exercise 10.8 Costas Loop, suppressed carrier synchronisation

Open the system:

`Digital Comm\ch_10\Costas-loop2.svu`

and follow the on-screen instructions.

Exercise 10.9 Open Loop Symbol Synchronisation

Open the sequence of systems:

Digital Comm\ch_10\open_symbol_sync_1.svu

Digital Comm\ch_10\open_symbol_sync_2.svu

Digital Comm\ch_10\open_symbol_sync_3.svu

and follow the on-screen instructions.

Exercise 10.10 MPSK Carrier Recovery

Open the system:

`Digital Comm\ch_10\mpsk-carrier-recovery.svu`

This file illustrates a technique for carrier recovery of an arbitrary MPSK signal. 8PSK was chosen as an example. The idea is simple. The actual received phase is quantized to the nearest possible phase (i.e. 0 , $\pi/4$, $\pi/2$, $3\pi/4$, etc.). The difference between the actual phase and the quantized version is used as an error signal to drive a tracking loop. The example file is a baseband simulation. Without loss of generality it is assumed that the carrier frequency is 0 Hz. In this way the maximum sample rate of the simulation can be reduced, improving the simulation time.

The first group of tokens (0,1,2, and 3) are used to generate the 8 phases of the modulation. A 22.5 degree offset is set in the sine token which represents the unknown phase α . The complex rotator token 4 is used to rotate the signal back into normal position under command of the output of the tracking loop. The $\arctan(\cdot)$ token is used to create the resulting phase which is used to derive the tracking error as mentioned.

- Run the system and watch the operation of the loop.
- What should the phase error become as the loop locks? The final value out of the integrator token is used to remove the unknown phase. What should this final value be? Verify your answer.
- Tokens 17 and 18 are the I and Q components after the phase rotation. The window w4 is the scatter plot of I vs. Q which is called the constellation. On the tool bar at the top, there is an icon called 'animate'. Select plot W2 and then click on this icon. Explain the action that results.

Exercise 10.11 Open Loop Symbol Synchronizer

Open the system:

`Digital Comm\ch_10\open-loop-symb-sync.svu`

Let $s(t)$ be a basic NRZ waveform with symbol time T .

Now compute the signal $z(t) = s(t)s(t - \tau)$. Note that if $\tau < T$, then there is an overlap of each bit with itself over a time $T - \tau$.

- (a) If $s(t) = \pm 1$, then what is the value of $z(t)$ during the overlap section.
- (b) Run the attached SystemView file. For simplicity we have chosen $T = 1$ and $\tau = 0.25$. Look at the waveform for the times

$$0.25 < t < 1$$

$$1.25 < t < 2$$

$$2.25 < t < 3$$

...

what is the value of $z(t)$ for these times? We conclude that $z(t)$ contains a periodic pulse train of period $T = 1$ and pulse width 0.75.

- (c) Now, the Fourier transform of a periodic sequence of period T has discrete frequencies at multiples of $R = 1/T$. In the SystemView example observe the spectra of $z(t)$ at frequencies 1 Hz, 2 Hz etc.

From Fourier analysis theory the transform of a delayed version of a signal is given by

$$\text{FT}\{s(t - \tau)\} = e^{j2\pi f\tau} \text{FT}\{s(t)\} \quad (10.7)$$

where $\text{FT}\{f(t)\}$ denotes Fourier transform of $f(t)$.

The combined result of this and (c) is that the **phase** of the spectral line at $f = 1/T$ contains the information on the timing of the input NRZ signal which is what we are after. Since the signal is a CW tone, we can lock on to it with a phase locked loop as shown in the SystemView example. The overall structure of the example is a symbol synchronizer.

- (d) To aid the PLL, it is desirable to maximize the power in the CW tone at $f = 1/T$. Run the SystemView example several times. Each time change the delay τ from 0 to $T = 1$. Look at the power in the line at 1 Hz for each case. For what value of τ is the power maximized?

Exercise 10.12 QPSK Carrier Recovery

Open the system:

`Digital Comm\ch_10\QPSK-carrier.svu`

This file is a carrier recovery loop using the phase detector model shown in Figure 9.36. The relative phase of the input carrier at $t = 0$ is 30 degrees, and there is a 40 kHz frequency offset. The error signal at token 13 is passed through a loop filter, token 6, whose output drives the FM VCO.

- (a) Run the simulation. Note that the loop locks in both phase and frequency. But the phase lock has a 4-fold ambiguity just as the Costas loop had a 2-fold ambiguity.
- (b) The VCO has a modulation gain of 40 kHz/v. What should the final steady state error of the loop filter be at lock? Verify your answer from the plots.

Look at the output waveforms in the analysis window and watch as the output signal goes into lock. Compare the input and output data signals. Are they the same? Why or why not?

Exercise 10.13 Early Late Bit Synchronizer

Open the system:

[Digital Comm\ch_10\early-late-bit-sync.svu](#)

The early/late gate closed-loop bit synchronizer is described in Section 10.2.2.2 of the textbook. Figure 10.13 is the associated block diagram. The SystemView exercise file represents this system in a somewhat modified form. The early and late samples can be combined to form the error discriminator curve which is sampled once per bit time T by the FMPT (VCO) token.

- (a) Run the simulation. The initial phase of the input data is controlled by the multiple run loop operation. Each loop advances the phase by 36 degrees starting from 0. Observe the loop as it locks up (error=0) for each case. What causes the 'oscillations' about 0 in the steady state error?
- (b) There are several parameters in the simulation that control the operation: the averaging time of the integrator, the early/late delay, and the modulation gain of the FMPT VCO token. Try various values of these parameters to see their effect on the loop operation.

Exercise 10.14 **Frame Synchronisation**

Open the sequence of systems:

`Digital Comm\ch_10\frame_sync_1.svu`

`Digital Comm\ch_10\frame_sync_2.svu`

and follow the on-screen instructions.


11 Multiplexing and Multiple Access

Exercise 11.1 Time Division Multiple Access

Open the system:

`Digital Comm\ch_11\time_division_ma.svu`

This system is taking four input signals and performing a time division duplex (or multiple access). Run the system and confirm the correct operation.


Use the probe  to view the signal at various points in the simulation.

Exercise 11.2 Frequency Division Multiple Access

Open the system:

`Digital Comm\ch_11\freq_division_ma.svu`

This system is taking four input signals and performing a frequency division duplex (or multiple access) using four frequency bands. Run the system and confirm the correct operation.


Use the probe  to view the signal at various points in the simulation.

Exercise 11.3 Code Division Multiple Access

Open the system:

`Digital Comm\ch_11\code_division_ma.svu`

This system is taking two input data streams and performing a code division duplex (or multiple access) orthogonal Walsh codes with a coding gain of 64 (i.e. 64 chips per input symbol).

- (a) Run the system and confirm the correct operation. Use the probe  to view the signal at various points in the simulation.
- (b) Change the orthogonal codes used (modify the selected row) and confirm the multiple access is still performing correctly.


12 Spread Spectrum Techniques

Exercise 12.1 Spread Spectrum

Open the system:

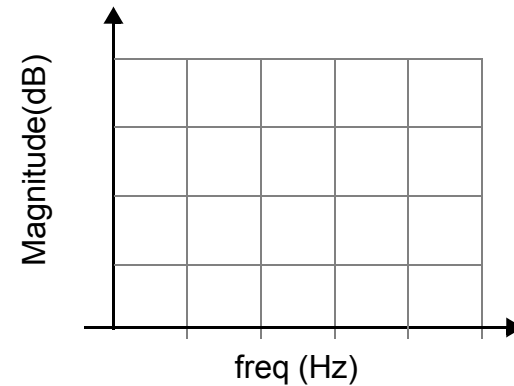
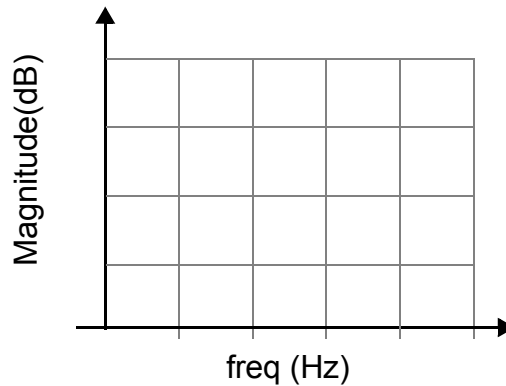
`Digital Comm\ch_12\spreadspectrum.svu`

The input signal to the system is a bit sequence at a rate of 19.2Kbps. It is mixed with a spreading sequence at a rate of 1.2288 Mcps per second (Mcps). Therefore the spreading factor is $1228800/19200=64$. The effect of this mixing operation is the spreading of the input sequence spectrum by a factor of 64. This kind of spreading is called direct sequence spread spectrum (DS-SS).

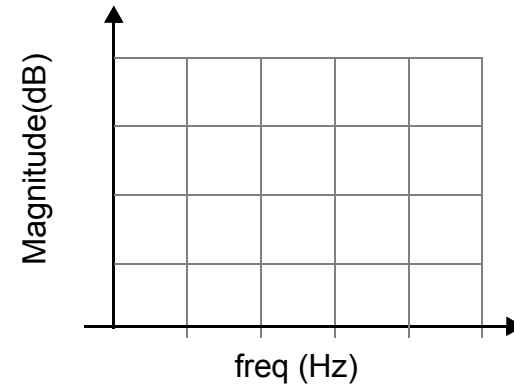
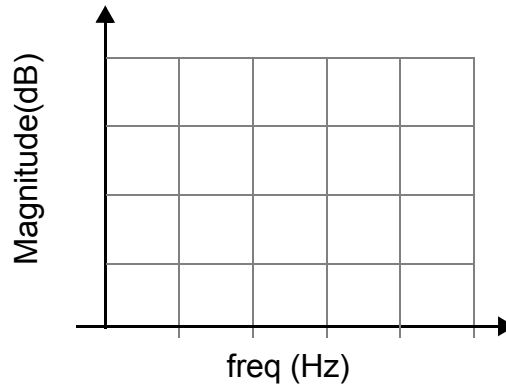
(a) Run the system and in the  **SystemView Analysis** window view the original data, and the chipped data.

Also note that the spectrum of the chipped signal occupied a much wider bandwidth.

Sketch the spectrum of the original and of the spread signal below.



- (b) Change the chip rate clock to 614400 Hz. Run the system again and sketch the spectrum of the input and output signals. What has happened to the bandwidth? Why should this be?



Exercise 12.2 Spread Spectrum IS95

Open and run the spread spectrum system:

`Digital Comm\ch_12\spreadspec_baseband.svu`

This system is performing a simple direct sequence spread spectrum (DS-SS). Note that the 19200 bits/sec data sequence is chipped with the spreading code, which is a pseudo random binary sequence (PRBS) at 1.2288MHz.

- Study the PRBS (or PN) generator function. Note the polynomial being used.
- Run the system and in the analysis window confirm the original data sequence can be recovered by the spreading & despreading functions at the transmit & receive.
- Change the delay in the channel from 0 to $1/1.228 \times 10^6$ ($= 8.1380208 \times 10^{-9}$) i.e. one chip duration. Run the system again. Are you able to recover the data now? The reason you cannot is the correlation is now one sample out of phase, hence there is no correlation for the PRBS function.
- Leaving the delay in the channel, synchronize your decorrelator (or matched filter) in the receiver by setting the delay also to $1/1.228 \times 10^6$. Does the system now successfully recover the original data?
- Inside the PRBS generator polynomial, change the seed value. Run the system again. Once again note that the data is NOT recovered. Why is this?
- Increase the no. of samples to 65636, run the system and in the analysis window perform a 20logFFT on the input data, and on the spread data. View the spectra and confirm that the main energy is contained between 0 and 9600 Hz for the input binary data and 0 and 1.288 MHz for the chipped data.

Exercise 12.3 Modulated Spread Spectrum

Open and run the spread spectrum system modulating onto a 10 MHz carrier.

`Digital Comm\ch_12\spreadspec_modulated.svu`

(Note this system may take a VERY long time to run.)

Exercise 12.4 Spread Spectrum with Noise

Open and run the system

`Digital Comm\ch_12\spreadspect_noise.svu`

- (a) Run the simulation and observe the effect of the noise at the output. The chip energy to noise spectral density ratio (E_c/N_0) is calculated according to the following equation

$$\frac{E_c}{N_0} = \frac{A_c^2}{R_c N_0} \quad (12.1)$$

where A_c^2 is the amplitude of the chip pulse, and R_c is the chip rate.

- (b) Increase the noise gain to 0dB, which is equivalent to having a chip energy to noise spectral density ratio $E_c/N_0=0$ dB. Run the system and note it is still possible to recover the original signal.
- (c) How would you obtain a “clean” signal at the output of the receiver? Open the system:

`Digital Comm\ch_12\spreadspect_noise_slice.svu`

for an answer.

Exercise 12.5 Integrate and Dump

The system implemented in:

`Digital Comm\ch_12\spreadspect_int&dump.svu`

perform three different kinds of low-pass filtering after the despreading operation in order to recover the transmitted signal

- Low pass FIR filter with a cut-off frequency of 19200 Hz and 99 taps
 - Averaging filter with an averaging period of $T_b=1/19200$ sec.
 - Integrate & Dump filter with an integration period of $T_b=1/19200$ sec.
- (a) Run the system and observe the output of the three filters. Note that the transmitted signal waveform can be discerned in the three of them. Note that the averaging operation and the integrate & dump filter are equivalent. The most effective way of performing the low pass filtering operation is the matched filter to the pulse shape, in our case a square pulse. The corresponding matched filter is to average or integrate & dump over a bit period $T_b=1/19200$ sec.
- (b) Open the system:


`Digital Comm\ch_12\spreadspect_int&dump_decimate.svu`

where a sampler has been placed after the integrate & dump token. This token provides one sample per bit. Compare the transmitted and the received signal. Place a slicer as suggested in the previous exercise in order to exactly recover the transmitted signal.

Exercise 12.6 CDMA Inherent Interference Rejection

In this example a interferer sinusoid is introduced in the channel. This could simulate an intentional jammer or a narrowband interference (NBI).

`Digital Comm\ch_12\spreadspect_interf_rejection.svu`

- (a) Run the system and in the  **SystemView Analysis** window observe the spectrum of the spread spectrum contaminated signal. Note the peak in the spectrum corresponding to the sinusoid. At which frequency does it appear? Does it correspond to the frequency of the sinusoid introduced in the channel?
- (b) Observe the spectrum of the despread signal. Is the interfering tone still present? Is it possible to recover the transmitted data? Note this is due to the low degree of correlation between the sinusoid and the spreading sequence.
- (c) Change the interfering sinusoid to an interfering frequency chirp from 650×10^3 Hz to 700×10^3 Hz. Choose the appropriate period considering your simulation time. Run the simulation again and check the spectrum of the spread spectrum contaminated signal and of the despread signal.

Can the receiver successfully detect the original signal?

Exercise 12.7 CDMA Frequency Diversity

By definition the spread spectrum signal has a much larger bandwidth than the original signal to be transmitted. Therefore, if a small enough part of that available bandwidth is lost it might still be possible to recover the transmitted signal if other frequencies are not affected. This is called frequency diversity.

In this example a part of the total available spread spectrum bandwidth is filtered using a stop band filter. Open the system:

`Digital Comm\ch_12\spreadspect_freq_diversity.svu`

- (a) Study the filter frequency response, what is the width of the stopband?
- (b) Run the simulation and check the spectrum of the transmitted signal, the received signal and the despread signal. Note that it is possible to recover the transmitted data?

Exercise 12.8 CDMA Multipath Diversity

In a spread spectrum system with a multipath channel, the transmitted signal travels following two different paths towards the receiver. A CDMA receiver can recover these two signals separately providing what is called multipath diversity. In a fading environment one of the multipath components might suffer a deep fade, but the transmitted information travelling on other multipath components might be preserved.

Open the system:

`Digital Comm\ch_12\spreadspec_multipath_diversity.svu`

- (a) Study the channel, how many multipaths are there? What is their attenuation?
- (b) Note the delays on branch 1 of the receiver, which multipath components are they synchronized to?

And the second branch?

- (c) Run the simulation and check that both branches can recover the transmitted signal.
- (d) Note there is no noise introduced in the channel, however the received signals seems to be noisy; where does this noise come from? Check exercise [Exercise 12.10](#) for an answer.

Exercise 12.10 Multiple Access Interference MAI

Open the system:

`Digital Comm\ch_12\spreadspect_two_users.svu`

In this system there are two users sharing the same frequency band and separated by the spreading codes.

- (a) Run the simulation and observe the recovered signal. Is it noisy?
- (b) Disconnecting the noise source in the channel and run the simulation again and observe the recovered signal. You should note that it is still noisy.

The interference introduced is due to the fact that the two users are sharing the same spectrum and even though they can be separated, they still create a level of interference on each other because the cross-correlation of the two spreading codes is not zero.

Exercise 12.11 Two Users CDMA

Open the system:

`Digital Comm\ch_12\multi-user-cdma.svu`

This example illustrates the operation of a CDMA system. In this case there are two signals sharing the same spectra at the same time, but each has a different maximal length sequence spreading code. The gain token 14, is defaulted to 1, and used to control the relative power between the two signals.

- (a) Run the simulation. Observe the recovered data streams with respect to the two original data signals.
- (b) Increase the power (gain token 13), of data 2 to see how much it can be increased until the data 1 channel output is degraded.
- (c) Try various combinations of the PN codes and repeat (b) above. Note that the PN token can generate arbitrary data patterns not necessarily limited to maximal length sequences.

Exercise 12.12 Spread Spectrum BER

Open the system:

`Digital Comm\ch_12\cdma-ber.svu`

In this system there are two systems running in parallel. The first (lower) is a basic baseband system which is equivalent to binary antipodal signalling. The second (upper) has the added feature of direct sequence spread spectrum (DS/SS) spreading and despreading. The system data rate is 10 Hz and the PN code is clocked at 50 Hz, providing a processing gain of 5 (7 dB). Remember that the processing gain is the number of chips per symbol.

This example uses the BER token from the Communications Library, which was designed for making bit-error-rate measurements. To understand the operation of this token, click the F1 button on your keyboard. This brings up all of the SystemView manuals (using Acrobat Reader). Under Optional Libraries, click Communications, and scroll to page 152 (Appendix A).

- (a) What do we expect from the difference in the BER performance of the two systems with additive Gaussian noise as the system impairment?
- (b) Run the simulation (it might take a minute or two).
- (c) Observe the BER performance of the two systems. Explain your result.
- (d) Change the chip rate from 50 Hz to 100 Hz, now the processing gain is 10 (10 dB). Run the simulation and observe the BER results obtained.
- (e) Can we say that DS/SS techniques provide any BER advantage against thermal noise?

Exercise 12.13 Orthogonal Codes

In this exercise, the same system of [Exercise 12.10](#) is implemented but using Walsh spreading codes, which are a kind of orthogonal spreading codes.

[Digital Comm\ch_12\spreadspectwo_users_orth.svu](#)

- (a) Study the system and check the token that generates the Walsh spreading code.
- (b) Run the system and note there is no MAI, this is because orthogonal spreading codes have been used.
- (c) Increase the amplitude of the data stream for user 2 (token 11) to 5 v. This is equivalent to increasing user's 2 power (near-far problem). Run the simulation. Can you observe any MAI? Keep on increasing user's 2 amplitude, can you observe any MAI? Is the orthogonality maintained?

Exercise 12.14 Orthogonal Codes in a Multipath Environment

The system:

`Digital Comm\ch_12\spreadspect_orth_multipath.svu`

is exactly the same as Exercise 12.13 but a multipath component has been added in the channel for the orthogonal and non-orthogonal cases.

- (a) Study the multipath channel. How many multipath components are there? What is the delay between them?
- (b) Run the simulation and observe the output of both systems (with and without orthogonal codes). Note that the output of the system with orthogonal sequences is much noisier than before, the multipath channel has reduced the orthogonality of the sequences used. Which system performs better under this conditions?
- (c) Disconnect the channel and connect the transmitter directly to the receiver (token 9 directly to tokens 3 and 18). Run the system, is orthogonality restored?

13 Source Coding

Exercise 13.1 Delta Modulator Exercise

Open the system:

`Digital Comm\ch_13\delta-modulator.svu`

The basic structure of a delta modulator is given in Figure 13.21 of the textbook. This exercise implements this modulator, where $a = 1$, and $\Delta s = 0.025$. The signal source is bandlimited white Gaussian noise.

- (a) Run the example and verify its operation.
- (b) Vary the value of the parameter, Δs , to see its effect on the delta modulator operation.
- (c) Vary the value of parameter, a , to see its effect on the delta modulator operation.
- (d) Try other input signals such as a sine wave.


Exercise 13.2 Dithered A/D

This exercise relates to the discussion of dithering in Section 13.2.4 of the textbook. The system:


`Digital Comm\ch_13\dithered-adc.svu`

illustrates the points made therein. For 10 bits of quantization, over the range ± 1 , the smallest quantization level is $1/512 = 1.953 \times 10^{-3}$ as stated. Note that the quantizer token rounds the input signal either up or down to the nearest quantized level. The output value of 0 is permitted. Thus the quantizer will 'trip' on a signal as small as $1/1024$.

This system implements two quantizers . The top one has an input with dithering noise, while the bottom one has no dithering.

(a) Run the system and go to the analysis window .


With the input signal at -63 dB, note that without the dither the output is 0, while the dithered signal produces a recognizable spectral line at 10 Hz. (You may have to zoom in to see this)

- (b) Return to the design space and connect the reconstruction filter  and check that the reconstructed sinusoid using dithering resembles the original sinusoid, while the recovered signal without dither is again just zero.
- (c) Now increase the signal size slowly by decreasing the amount of attenuation in the gain tokens. Explain your result.

Exercise 13.3 Sigma-Delta A/D Converter

Open the system:

`Digital Comm\ch_13\sigma-delta.svu`

- (a) Run the system and look at the plot in the analysis window . This shows an overlay of the input signal and the two quantization noise error signals in the frequency domain. Note how the spectrum of the sigma-delta quantization noise 'rolls off' as the frequency goes to zero. The normal A/D quantization error is flat as a function of frequency.
- (b) Change the number of bits in the sigma-delta quantizer as well as in the standard quantizer from 6 to 10 bits. Can you still observe the same effect?
- (c) Repeat the previous step but using now 2 bit quantizers. Is the amount of quantization noise still reduced in the band of interest?
- (d) What is the advantage of the sigma-delta converter?

Exercise 13.4 First Order Sigma Delta Simulation

Open the system:

`Digital Comm\ch_13\first_order.svu`

and study the circuit.

This system has a 5 kHz bandlimited input signal (100 Hz to 4800 Hz chirp) which has a Nyquist sampling rate of 10 kHz. The system is oversampling at 64 x's, i.e. $f_s = 640$ kHz. The system contains a 10 bit quantizer with the input/output characteristic shown below in Figure 13.1(a) and the sigma delta loop uses a 1 bit converter (or comparator) as shown in Figure 13.1(b). The 10 quantizer is simply provided for "resolution"

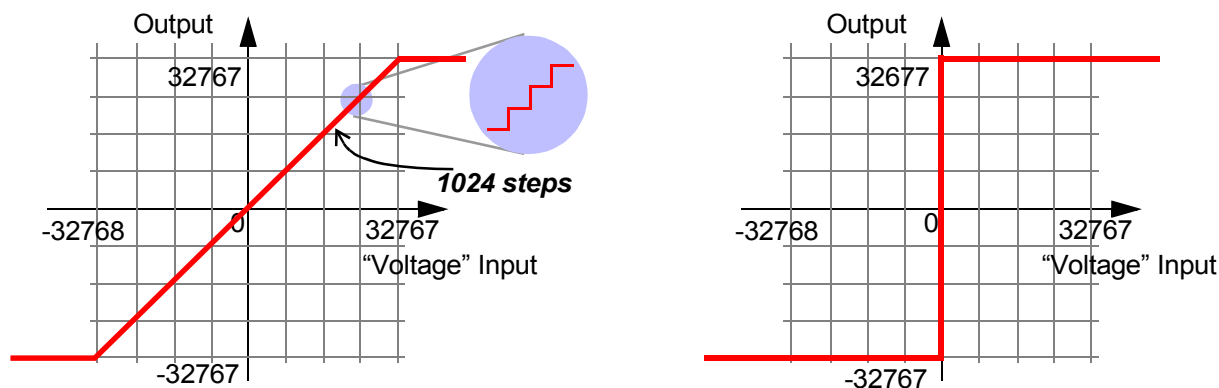


Figure 13.1: (a) 10 bit quantizer, (b) 1 bit quantizer (or comparator). Note the quantizers are quantizing from a real number to a real number output.

comparison purposes and is NOT part of the sigma delta loop. We would expect that using a first order sigma delta loop we can achieve the equivalent of about 8-9 bits of resolution of a Nyquist rate converter/quantizer.

- (a) Run the system and view the various time and frequency plots in the analysis window.
- (b) Compare the time domain versions of the 10 bit and 1 bit sigma delta signals; clearly the 1 bit signal has only 2 quantization levels, whereas the 10 bit signal has 1024 quantization levels. Also note the “noise shaping” that can be seen from the FFT of the 1 bit (oversampled) sigma delta signal (also illustrated in [Figure 13.2](#)).

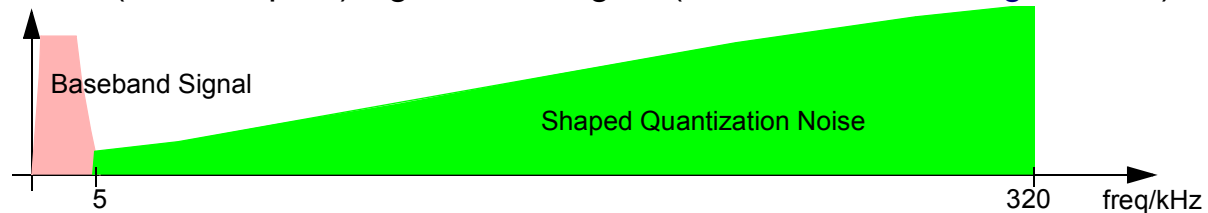


Figure 13.2: *First order sigma delta quantization noise shaping. FFT of a 1 bit resolution signal bandlimited to 5kHz and oversampled at 64 x's at $f_s = 640\text{kHz}$.*

- (c) Try running with another “distinctive” input such as a pseudo noise (at say 1000Hz), a sine wave.

Exercise 13.5 First Order Sigma Delta loop model

Open the system:

Digital Comm\ch_13\model.svu

This system includes a first order sigma delta loop, and also a simulation of the linear model often used for analysis, as shown in Figure 13.3.

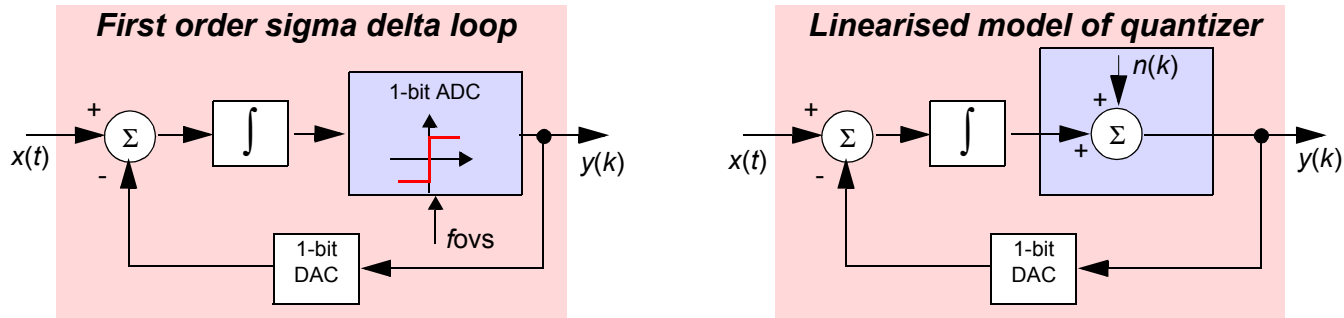


Figure 13.3: Linear model of a first order sigma delta noise loop. The 1 bit quantizer is modelled as an additive white noise source, $n(k)$, in the right hand figure.

Run the system and observe in the analysis window that the two high pass noise profiles are very similar. Therefore from this simulation, we can perhaps conclude that the linear model is “reasonable”!

Exercise 13.6 Decimated Output of 1st Order Sigma Delta

Open the system:

`Digital Comm\ch_13\decimate.svu`

This circuit is the same as in [Exercise 13.4](#) except that the sigma delta loop output is decimated by a lowpass filter which is downsampling by 64, and the input signal is bandlimited from 50 to 3000Hz. Open the decimating filter and confirm that the magnitude frequency response is as shown in [Figure 13.4](#), and note the “decimate” parameter is set to 64 (see below [Figure 13.4](#)) and therefore the output of the decimation filter is sampled at 10kHz.

Run the system and view the analysis window. Note by viewing both time and frequency domain that the output of the low pass filter reveals that we have reproduced the chirp with approximately 8-9 bits of true resolution from the 1 bit sigma delta signal which was input.

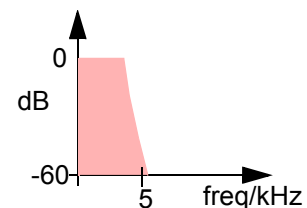
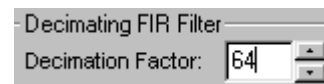


Figure 13.4: *Low pass decimation filter characteristic.*



Exercise 13.7 Quantizing Speech to 3 bits

Open the system:

`Digital Comm\ch_13\quantizing_3bits.svu`

- (a) Run the system and listen to the various outputs. In particular, note the speech *drop-out* from the mid-level quantizer. Observe the various signals in the frequency domain. The mid-riser has no drop-out. Ensure you know why this is.
- (b) Now open the system:

`Digital Comm\ch_13\quantizing_3bits_dither.svu`

This time we have included a dithering signal which breaks the *quantization error signal* and *speech signal* correlation.


Run the system and listen to the outputs and make a qualitative comment on the introduction of the dithering compared to not using dithering.

Exercise 13.8 Compressing and Expanding

Open the system:

`Digital Comm\ch_13\companding.svu`



This example takes 16 bit speech, converts to 13 bit resolution via a simple scaling consisting of multiplying by $2^{12} = 4096$ and dividing by the maximum 16 bit value of $2^{15} = 32768$, i.e, scaling by $1/8$. The signal is then passed through a μ -law compressor to give an 8 bit compressed signal, which is then expanded back to 13 bits. The process of first *compressing* then *expanding* is denoted *companding*.

- (a) Run the system and listen to the input signal and the 8 bit uniform quantized signal, and the 8 bit companded signal. (Very!) careful listening should reveal that the companded signal has better quality than the 8 bit linear quantized and is almost as good as the 13 bit signal.
- (b) In the analysis window  observe the amplitude range of the 8 bit uniform quantized signal, and the 8 bit compressed signal. Calculate the histograms for both and therefore comment on the effect on the segmental SNR.

Exercise 13.9 Differential Speech Coding


Open the system:

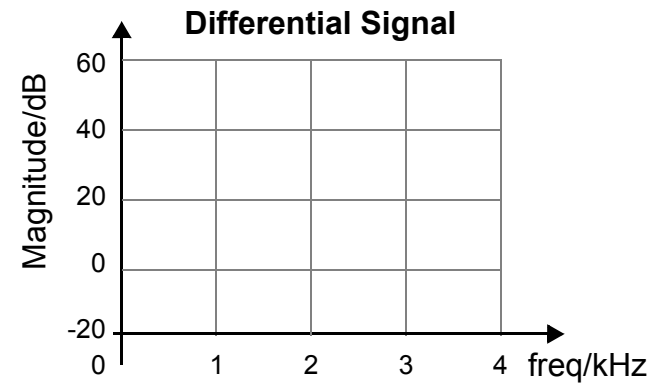
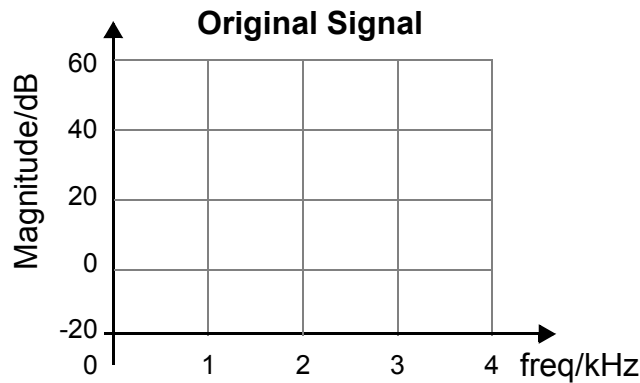
Digital Comm\ch_13\differential.svu

- Run the system and listen to the input signal and differentially coded signal. You should hear that the differentially encoded signal is high pass filtered.
- In the analysis window  observe the amplitude range of the original signal, and the *differentially encoded signal*.
- Using the statistics button  observe the parameters below for the two signals (recall that mean squared value is the square of the standard deviation).

	Original	Differentially Encoded
Mean squared value		
Mean value		
Maximum Value		
Minimum value		

Noting that the original signal had a linear dynamic range from around -23000 to 26000 and the differentially encoded signal had a range from -15000 to 13000 , then total dynamic range for the original signal and differentially encoded signal is approximately 49000 and 28000 respectively.

- (d) Using the calculator  choose **Spectrum** and generate the magnitude frequency spectrum **20 Log |FFT| (dB)** for both the original and differentially encoded signals. You should observe that the spectrum of the differentially encoded signal is somewhat flatter than that of the original spectrum.
- (e) Sketch the magnitude frequency spectra of the original and differential signals below:



Exercise 13.10 Difference Speech Coding and Decoding

Open the system:

`Digital Comm\ch_13\diff_coder_decoder.svu`

(a) Run the system and confirm that the decoding is essentially “perfect”.

(b) Now open the system

`Digital Comm\ch_13\diff_coder_decoder2.svu`

and for the speech signal confirm the output “sounds the same”.

Once a signal has been coded we require to produce the decoder. If we consider the z -transform of the differentiator, then:

$$\begin{aligned} D(z) &= X(z) - X(z)z^{-1} = X(z)[1 - z^{-1}] \\ \Rightarrow H(z) &= \frac{D(z)}{X(z)} = (1 - z^{-1}) \end{aligned} \quad (13.1)$$

Therefore to decode back to the original signal:

$$X(z) = \frac{D(z)}{1 - z^{-1}} \quad \Rightarrow \quad G(z) = \frac{1}{H(z)} = H^{-1}(z) = \frac{X(z)}{D(z)} = \frac{1}{1 - z^{-1}} \quad (13.2)$$

The difference equation for this is:

$$x(k) = d(k) + x(k-1) \quad (13.3)$$

which is a recursive equation with a single pole at $z = 1$, i.e. an *integrator*.

Exercise 13.11 Delta Modulation of Speech

Open the system:

`Digital Comm\ch_13\delta_modulator.svu`

The quantizer is set to levels of ± 1 , and the input signal is 16 bit speech signal sampled at 8000Hz

- (a) Run the system and note the output of the delta modulator and the decoded signal. It would appear that the decoded output bears no visual resemblance to the original and slope overload has occurred. (Note that the speech signal is still there in the decoded output....try playing at a very high volume, or rerun the system and add a linear gain of 100 before the audio output token!)
- (b) Increase the delta modulator step size to ± 100 . Run the system again and view the output and listen to the result. It would appear that this is better quality, but both slope overload and granularity effects are now audible.
- (c) Increase the step size further to ± 1000 , run again and listen to the output and view the input.
- (d) Modify the integrator to be "leaky" and run again.

Exercise 13.12 Linear Predictive Coding with the Residual

In this example we will perform a linear predictive coding and decoding of a voiced speech signal. Open the speech LPC “coder” system:

`Digital Comm\ch_13\lpc_weights_voiced.svu`

- (a) Run the system and listen to the original vowel “*ahhh*”. Also look at the FIR predictor filter weights, which are saved to file

`Digital Comm\External Files\general\lpc_weights.txt`

- (b):

`Digital Comm\ch_13\lpc_residual_voiced.svu`

In this file we generate the residual by passing the voiced speech (vowel “*ahhh*”) through an all-zero filter with the coefficients generated in the previous file and stored in

`Digital Comm\External Files\general\lpc_weights.txt`


Note the residual signal is save to an external file

`Digital Comm\External Files\sinks\residual_error.snk.`

- (c) In the analysis window observe the residual signal and also the original vowel waveform (note we are using 160 ms of sound to allow us to clearly hear results - typical single frames in speech coding are around 10 to 30 ms and too short to aurally appreciate).

- (d) Now open the system speech LPC “decoder”:

`Digital Comm\ch_13\lpc_synthesis_voiced.svu`

Copy the predictor filter weights produced from previous system `lpc_residual_voiced.svu` into the all pole model and check that the external file token  and is reading in the residual file `residual_error.snk`.

Run the system and listen to the synthesised output vowel. How does it sound?

- (e) This coder-decoder is LPC-25 because the filter has 25 LPC weights. More typically we use LPC-10. Modify to an LPC-10 system and run again from part (a).

Exercise 13.13 Linear Predictive Coding with Excitation Pulses

Open the system:

`Digital Comm\ch_13\lpc_synthesis_voiced_pulses.svu`

- (a) Run the system and in the analysis window calculate the pitch period. There are two ways to do this: (i) take the FFT of the residual signal, and (ii) take the autocorrelation of the residual signal and count the number of samples between peaks. Both should give the same answer (Check that 133 Hz is about right for the speech sample.)
- (b) Now in the decoder part of the system, read in the $A(z)$ filter weights to the all-pole model, set up the input pulse train to be impulses of frequency 133 Hz, (leave the amplitude at 5000 and the pulse width at $1/8000 = 125 \mu s$) and run the system. Listen to the synthetic vowel. How does it sound? In the analysis window look at the time waveform for the original and synthetic vowels and also calculate the magnitude frequency spectra and compare.
- (c) Try decoding with the wrong pitch period, say 150 Hz; how does the result sound.
- (d) Set the pitch period back to 133 Hz, run again and confirm the coder is still working. Now open the all pole weight linear system and modify the 5th weight (indexed as 4) from its value (probably around 0.887) to 0.4. This might represent a change due to a simple bit error. Run the system and listen to the output. It should fail completely. Therefore even a very small change in the predictor weights can lead to a significant change in the performance.
- (e) Repeat for the vowel “eee”.

Exercise 13.14 Linear Predictive Coding for Unvoiced Signals

Open the system:

```
Digital Comm\ch_13\lpc_weights_unvoiced.svu
```

(a) Run the system and follow the on-screen instructions to generate the LPC weights file `lpc_weights.txt`.

(b) Open the system:

```
Digital Comm\ch_13\lpc_residual_unvoiced.svu
```

using the filter weights `lpc_weights.txt` generated with the previous SystemView file, the residual error is generated and saved into the file `residual_error.snk`.

(c) Open the system:

```
Digital Comm\ch_13\lpc_synthesis_unvoiced.svu
```

This file synthesises the unvoiced sound by using the LPC weights stored in `lpc_weights.txt` and by exciting the system with the residual error stored in `residual_error.snk`.

(d) Open the system:

```
Digital Comm\ch_13\lpc_synthesis_unvoiced2.svu
```

This system is similar to the previous one, in this case the unvoiced signal is synthesised by using white noise to drive the system rather than the residual error.

Exercise 13.15 Subband Speech Coding

Using a two subband system, design a subband speech coder (compressor) for wideband audio coding, whereby the upper (and less “important”) subband from 4-8 kHz is linearly coded with only 4 bits compared to 12 bits encoding for the lower subband, 0-4 kHz.

Using the 16 bit speech signal

Digital Comm\External Files\Sources\Fssm16.wav

(16 bit resolution and sampled at 16kHz), listen to the result and compare with the uncoded signal.

Exercise 13.16 Frequency Domain Coding

Open the system:

`Digital Comm\ch_13\freqtoken4.svu`

and study the general principle of quantizing the no. of bits of a signal in the frequency domain rather than the time domain.

14 Encryption and Decryption

Exercise 14.1 RSA Encryption

Open the system:

`Digital Comm\ch_14\rsa_encryption.svu`

This example implements the RSA encryption algorithm using the 'square-and-multiply' computational procedure. The encryption algorithm computes a message M as:

$$M = [C^e] \bmod N$$

C = plain text message

$$e = 17$$

$$N = 2773$$

The decryption algorithm is similar:

$$C = [M^e] \bmod N$$

M = encrypted message

$$e = 17$$

$$N = 2773$$

In the system there is a custom source token (t12) that converts a text message (entered in the token parameter text window) to 7 bit ASCII words, $0 < C < 127$. This token has two input parameters: (1) "*Input Character Rate*" in chars/sec - this controls

the rate that the text is printed on the screen; (2) “*Pad Samples*” - this allows room to represent e as a binary number. The RSA algorithm then produces an encrypted text, M , in the form of a 7 bit ASCII representation. A second custom token (t13) converts the ASCII binary word back to an alphabetical symbol. The encrypted message M is then sent to the decryption token “*dll*” (t18) which produces the ASCII binary form or the original message. Finally, a custom token (t19) reproduces the original message.

- (a) Run the system and verify that the recovered message is the same as the original message.
- (b) Open the custom source (t16) parameter dialog box and enter another message. Rerun the simulation.
- (c) Review the ‘*square-and-multiply*’ algorithm as detailed in Table 14.13 of the textbook. Note that the algorithm has two modes depending on whether the binary bit of the parameter e is a 1 or a 0.
- (d) Review the simulation. See if you can verify its operation with respect to the ‘*square-and-multiply*’ algorithm.
- (e) Note that several of the parameters in the simulation are globally linked (pull down the “Tools” menu on menubar at the top to view parameters used in the encryption algorithm). Pick another set of parameters (N , e , etc.) and enter them. Rerun the simulation to verify proper operation.

Exercise 14.2 Encrypted Email!

Open the system:

`Digital Comm\ch_14\rsa_send.svu`

- (a) Type your message in the source token text parameter box. There is an external sink (token 16) which will store your message to the file `rsa_send.txt`.
- (b) Now open:

`Digital Comm\ch_14\rsa_receive.svu`

Run the simulation and decipher the message.

Now you are equipped to send encrypted emails to any colleagues who also run SystemView!

15 Fading Channels

Exercise 15.1 Rake Receiver Principles

Open the system:

`Digital Comm\ch_15\spread.svu`


This system spreads the spectrum of a data sequence at **9600bits/sec** with a **chip rate of 192000Hz**, i.e. a coding gain of 20. For the simulation, the sampling rate is set to 5x's the chip rate at 960000Hz.

(a) Run the system and note that in the presence of no noise we can perfectly recover the original data sequence.

(b) Now open the system:

`Digital Comm\ch_15\rayleigh.svu`

In this system a Rayleigh fading path is introduced. Note there is no delay added in and hence the signal power of the received signal is just varying according to the Rayleigh fading channel specification.


Run the system and note in the analysis window  that the received signal is somewhat recovered, but with varying power.

Put in a "slicer" or decision device at the final output to "recover" the binary levels of the data. For this simple power variation, a slicer will allow a zero error output to be produced.

(c) Now open the system:

`Digital Comm\ch_15\rayleigh_delay.svu`


where the Rayleigh fading path has a delay of 208.33 μ secs. Clearly to receive this correctly we need to synchronize the receiver with the same delay.

Run the system and note in the analysis window  that the received signal is similar to your observations above. Of course, if we are perfectly synchronized then we would expect the performance of this to be *identical* to the previous system.

(d) Next view the system:

`Digital Comm\ch_15\rayleigh_two_paths1.svu`


where there are two Rayleigh fading paths, the first being direct and the second having a delay of 208.33 μ secs. The receiver is only synchronized to the first path and hence the second path will contribute noise at the receiver.

Run the system and note in the analysis window  that the received signal can be recovered but with noise contributed by the second path. The second delay path may of course cause errors to occur now. (You can observe these or detect with a BER token.)

(e) In the system:

`Digital Comm\ch_15\rayleigh_two_paths2.svu`


where there are two Rayleigh fading paths, this time the receiver is synchronized to the second (or delayed) path and hence the first path will contribute noise at the receiver.

Run the system and note in the analysis window  that the received signal can be recovered from the second path only, but again there may be errors due to power from the first (direct) path.

(f) In the system:

`Digital Comm\ch_15\rayleigh_synch.svu`


there are now two matched filter receivers one synchronized to each of the two paths. Note that the output of each receiver is going to be different and depend on the precise attenuation of the path at a given time, and the level of power output from the other path. So sometimes one path will give an error, whereas the other path will not.

Run the system and note the analysis window  results.

(g) Finally in the system:

`Digital Comm\ch_15\rake_receiver.svu`

the outputs of the two receivers are combined (with equal gain).

Run the simulation and observe the analysis window  results. The combined output should produce fewer errors than the output of each individual branch.

This receiver is said to present diversity; if one of the branches of the receiver fails to decode the received signal properly, there is a probability that the other branch might recover it and vice versa.

Exercise 15.2 Spread Spectrum and Multipath Resistance

Open the system:


`Digital Comm\ch_15\spread_and_nospread.svu`

This system takes a data sequence at 10 bits/sec and spreads to 200 chips/bit (spreading gain of 20).

Run the system and observe that the spread output in both time and frequency domain.

(h) Now open the system:

`Digital Comm\ch_15\spread_despread.svu`

This system has a matched filter receiver (despreader). Observe in the analysis window  that we can perfectly recover the original data.

(i) Next open the system:

`Digital Comm\ch_15\spread_multipath.svu`

where a multipath channel has been added for both the spread on nonspread transmissions paths. The channel has 3 paths, one direct, and two delayed paths all with the same gain. The impulse response is then:

Given that one symbol period is 0.1 sec then the multipath will cause intersymbol interference for the symbol rate channel.

However for the chipped data, the path delays will produce sequences that are uncorrelated with the original and hence the delayed path signal will be largely ignored (averaged out) by the despreader or matched filter.

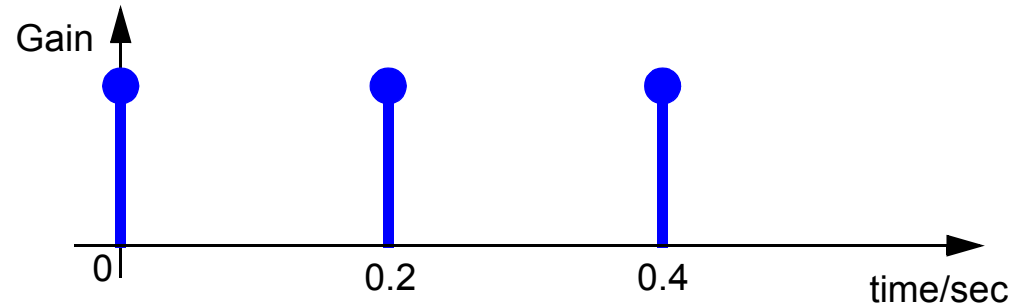



Figure 15.1: *Three path channel - all equal gain.*

Run the system and observe in the analysis window  that the spread signal is recovered with few errors, whereas the non-spread signal suffers greatly from ISI and has a large number of errors.

Exercise 15.3 Rayleigh and Rician pdf

Open the system:

[Digital Comm\ch_15\rayleigh-pdf.svu](#)

This exercise shows the details of generating a Rayleigh and Rician pdf. Consider a two dimensional Gaussian probability density function of the form

$$p(x, y)dxdy = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} dxdy \quad (15.1)$$

Now change variables from rectangular to polar coordinates

$$x = r\cos\theta \quad (15.2)$$

$$y = r\sin\theta \quad (15.3)$$

Then in polar coordinates the pdf becomes

$$p(r, \theta)rdrd\theta = \frac{1}{2\pi\sigma^2} e^{-\frac{r^2}{2\sigma^2}} rdrd\theta \quad (15.4)$$

The integral over the angle is trivial, leaving the net pdf for the r variable as

$$p(r) = \frac{1}{\sigma^2} r e^{-\frac{r^2}{2\sigma^2}} \quad (15.5)$$

which is the Rayleigh pdf. This means that we can easily generate a Rayleigh process by simply taking the envelope of two statistically independent GRV.

- (a) Run the system and verify that the operation labelled Rayleigh pdf will produce $p(r)$ as described above.
- (b) To produce the Rician pdf add a mean to the x variable as shown on the group labelled Rician pdf. The derivation for $p(r)$ is the same as the above, but the existence of the mean makes it more complicated. The net result is given in Equation (15.14) of the textbook.
- (c) Run the simulation and go to the Sink Calculator. Observe the overlay of the two pdf's.
- (d) Change the mean of the Rician distribution from 2 to 6 and run the simulation, what type of pdf does the resulting plot look like?

Exercise 15.4 Wideband Frequency-Selective Fading Channel Model

A general model for wideband frequency-selective fading systems is shown in Figure 15.2

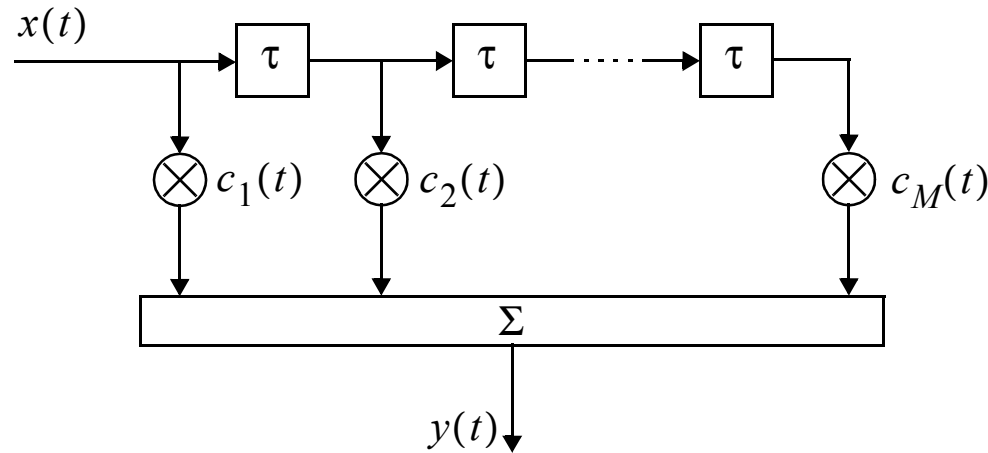


Figure 15.2: Wideband frequency fading system model

$$y(t) = \sum_m c_m(t) x(t - m\tau) \quad (15.6)$$


Thus the channel can be considered as an FIR filter with time varying coefficients. In the above equation, the reciprocal of the delay τ is less than the bandwidth of the signal $x(t)$. The coefficients $c_m(t)$ are generally chosen such that each individual tap of the

filter will produce a frequency flat fade.

Open the system:

[Digital Comm\ch_15\freq-select-fading.svu](#)

This system implements the model shown in [Figure 15.2](#). For simplicity the coefficients $c_m(t)$ are set to independent numbers uniformly distributed over ± 1 . Every time the simulation is run, the coefficients value change randomly as noted. This in turn changes the channel impulse response, and the channel frequency spectra. Verify that the basic signal processing shown is in accord with the figure and equation above.

Run the system several times, and after each run observe the channel frequency response in the . Note that each coefficient $c_m(t)$ simulates a multipath component with a certain delay and amplitude. Note that each time you run the simulation the specifics of the model will change. Compare the resulting spectra with [Figure 15.9\(a\)](#) in the textbook.

Exercise 15.5 Rayleigh Fading BER Exercise

Chapter 15 details the effects of multipath fading on a communication system. A result of fundamental importance is the bit error rate (BER) of a system when fading occurs. Table 15.1 tabulates the results for various types of modulation.

- (a) Compare the BER performance as detailed in Table 15.1 vs. the non-fading counterpart as a function of E_b/N_0 . How serious is the degradation in BER when there is fading?
- (b) Open the system:

`Digital Comm\ch_15\Rayleigh-fading-BER.svu`

The Rayleigh fading is simulated in this example using the Rician channel token using a factor $K = 0$, which produces a Rayleigh distribution. It simulates the BER performance of coherent PSK with and without fading. The theoretical value from Table 15.1 is overlaid. Run the simulation and verify the performance.

- (c) How seriously does the fading channel reduce the BER performance compared to a channel without fading?

Exercise 15.6 Coherent PSK vs. DPSK BER in a Rayleigh Fading Channel

Open the system:

`Digital Comm\ch_15\Rayleigh-coherent-BER.svu`

- (a) Compare the BER performance with fading for coherent PSK vs. DPSK as detailed in Table 15.1 in the textbook. What is the loss in dB of DPSK vs. coherent processing?

- (b) Make the same comparison for the case where there is no fading (see Sections 4.4.1 and 4.7.5 in the textbook). What do you think about the feasibility of sending a coherent reference signal along with the modulated data signal?

It simulates the BER performance for the two cases under consideration. Run the simulation. Verify the relative performance as a function of E_b/N_0 .

- (c) Equation 15.44 in the textbook details the improvement in SNR when a maximal ratio combining diversity (rake) receiver is used. The question remains, what is the effect of diversity on the system BER performance? It can be shown [see reference 10 in chapter 15], that the BER performance with L -path diversity can be written in the form

$$P_b = c(L) \frac{1}{[(E_b/N_0)E(\alpha^2)]^L} \quad (15.7)$$

where $c(L)$ is a constant depending on L ., and α is the fading envelope of the signal. From this equation, comment on the value gained by using diversity techniques.

Exercise 15.7 Diversity Combining for a 3-path Fading Signal

In the textbook, section 15.5.4.1 describes four methods for diversity combining: (1) selection, (2) feedback, (3) maximal ratio, and (4) equal gain. Here an example implementing selection, maximal ratio and equal gain combining is presented.

In selection combining, the largest multipath channel component magnitude is chosen to perform a decision. For maximal ratio, each tap is multiplied by the signal amplitude in that tap. In equal gain combining the individual tap outputs are added together with the same gain.

(a) Open the system:

`Digital Comm\ch_15\Diversity-combining.svu`

This example illustrates these three techniques side by side. Run the simulation several times in order to observe the action of each combiner under a variety of conditions of fading in each individual channel.

- (b) By observing the results obtained, which technique seems to work better than the others?
- (c) Which technique seems to reduce the number of deep fades or attenuations in the received signal amplitude?

Exercise 15.8 Rake Receiver

Open the system:

[Digital Comm\ch_15\Rake-receiver.svu](#)

This exercise illustrates the operation of a rake receiver. The input data signal is multiplied by the spreading PN code. The resulting signal is subject to two independently fading Rayleigh channels. Two PN correlators are used, one to recover each multipath channel. These demodulated signals are finally added together to produce the recovered data signal.

- (a) Run the example file. In the analysis window there are three plots: (1) the original data, (2) an overlay of the two individually recovered rake signals, and (3) the combined signal which is the sum of the two rake channels.
- (b) Run the simulation several times and observe the action of the receiver as the individual channels fade independently from each other.

Exercise 15.9 Rake Receiver Anti-multipath

In a communications system, if the channel model has individual paths separated in time greater than the data time, the resulting signal will exhibit inter symbol interference (ISI). One approach to mitigating this problem is to use some sort of equalizing filter. Another approach is to use direct sequence spread spectrum (DSPN) techniques combined with a rake receiver. If the DSPN chip time is short compared to the path delays, then each path can be individually resolved. The individual taps can then be combined. Thus as one path fades the other paths can maintain the signal to noise ratio. Open the system:

`Digital Comm\ch_15\Rake-receiver2.svu`

- (a) Run the exercise file and observe this effect. Observe the overlay of the transmitted and received signal for both systems, with and without rake receiver (labelled with and without PN). For the Rake receiver, transmitted and received signals are identical (therefore their overlay looks as just one single signal), meaning there are no errors in the transmission. For the case without the rake receiver, the transmitted and received signals are not identical and errors are introduced.
- (b) The processing gain of a DSPN system is the ratio of the data rate to the PN code clock rate. In this system the higher the processing gain, the greater the ability of the rake receiver to resolve closely spaced paths. Vary the clock token 11 from 200 Hz (spreading gain of 20) down to 10 Hz (spreading gain of 1, e.g. no spread spectrum) and observe the degradation of the system performance (observe the overlay of transmitted and received signals and check that they are not identical).