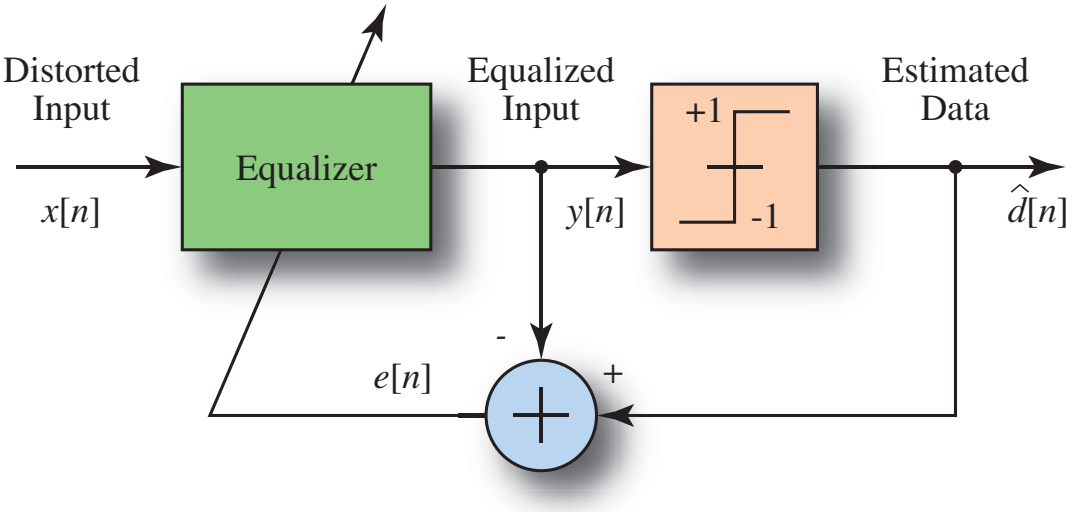
Statistical Signal Processing

ECE 5615 Lecture Notes

Spring 2019



Chapter 1

Course Introduction/Overview

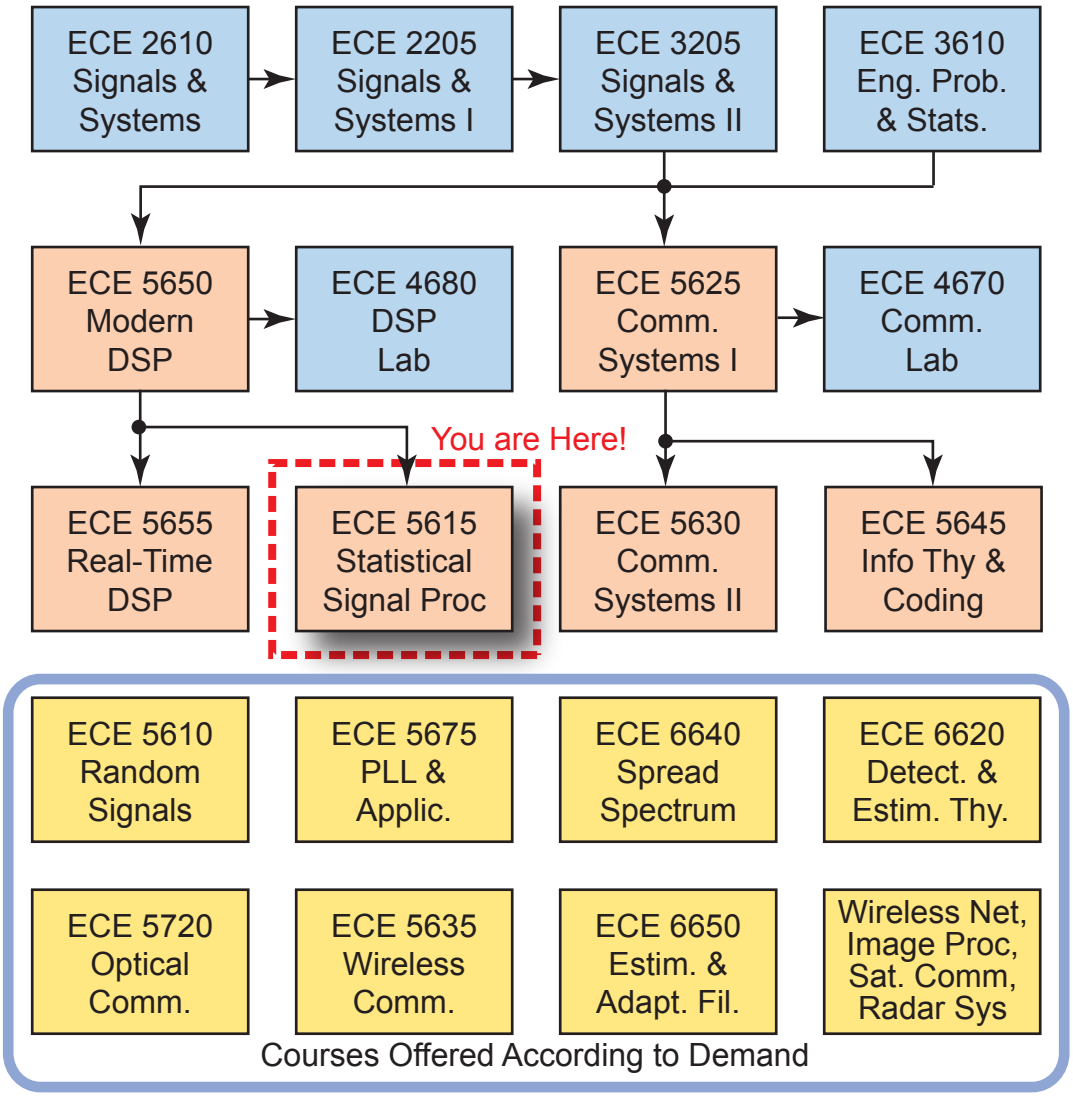
Contents

.

1.1 Introduction

- Course perspective
- Course syllabus
- Instructor policies
- Software tools for this course
- Required background
- Statistical signal processing overview

1.2 Where are we in the Comm/DSP Curriculum?



1.3 Instructor Policies

- Working homework problems will be a very important aspect of this course
- Each student is to his/her own work and be diligent in keeping up with problems assignments
- Homework papers are due at the start of class
- If work travel keeps you from attending class on some evening, please inform me ahead of time so I can plan accordingly, and you can make arrangements for turning in papers
- The course web site

  `http://www.eas.uccs.edu/wickert/ece5615/`

  will serve as an information source between weekly class meetings
- Please check the web site updated course notes, assignments, hints pages, and other important course news; particularly on days when weather may result in a late afternoon closing of the campus
- Grading is done on a straight 90, 80, 70, ... scale with curving below these thresholds if needed
- Homework solutions will be placed on the course Web site in PDF format with security password required; hints pages may also be provided

1.4 The Role of Computer Analysis/ Simulation Tools

- In working homework problems pencil and paper type solutions are mostly all that is needed
  - It may be that problems will be worked at the board by students
  - In any case pencil and paper solutions are still required to be turned in later
- Occasionally an analytical expression may need to be plotted, here a visualization tool such as Python (via the Jupyter qtconsole or notebook), Mathematica or MATLAB/Octave[1] will be very helpful
- Simple simulations can be useful in enhancing your understanding of mathematical concepts
- The use of Python for computer work is encouraged since it is fast and efficient at evaluating mathematical models and running Monte-Carlo system simulations
- There are multiple Python based simulation projects embedded throughout the course, as the Hayes text supports this with MATLAB based exercises at the end of each chapter
  - Text MATLAB examples are easily converted to Python

[1] https://www.gnu.org/software/octave/

1.5 Required Background

- Undergraduate probability and random variables (ECE 3610 or similar)
- Linear systems theory
  - A course in discrete-time linear systems such as, ECE 5650 Modern DSP, is highly recommend (a review is provided in Chapter 2)
- Familiarity with linear algebra and matrix theory, as matrix notation will be used throughout the course (a review is provided in Chapter 2)
- A desire to use the Jupyter/IPython Notebook, but using Mathematica or back to MATLAB/Octave as needed
- Homework problems will require the use of some vector/matrix computational tool (prefer Python at this point in time), and a reasonable degree of proficiency will allow your work to proceed quickly
- A desire to dig in!

1.6 Statistical Signal Processing?

- The author points out that the text title is not unique, in fact *A Second Course in Discrete-Time Signal Processing* is also appropriate
- The Hayes text covers:
  - Review of discrete-time signal processing and matrix theory for statistical signal processing
  - Discrete-time random processes
  - Signal modeling
  - The Levinson and Related Recursions
  - Wiener and Kalman filtering
  - Spectrum estimation
  - Adaptive filters
- The intent of this course is not entirely aligned with the text topics, as this course is also attempting to fill the void left by *Random Signals*, *Detection and Extraction of Signals from Noise*, and *Estimation Theory and Adaptive Filtering*
- A second edition to the classic detection and estimation theory text by Van Trees is another optional text for 2015; This is the text I first learned from
- Three Steven Kay books on detection and estimation are now optional texts, and may take the place of the Hayes book in the future

1.7 Course Syllabus-1

ECE 5615/4615
Statistical Signal Processing

Spring Semester 2019

Instructor: Dr. Mark Wickert **Office:** EN292 **Phone:** 255-3500
mwickert@uccs.edu **Fax:** 255-3589
http://www.eas.uccs.edu/~mwickert/ece5615/

Office Hrs: Wednesday 10:40–11:45 AM or alternative?, other times by appointment.

Required Texts: Monson H. Hayes, *Statistical Digital Signal Processing and Modeling*, John Wiley, 1996. Also Python Machine Learning from Packt for $5.

Optional Texts: H.L. Van Trees, K.L. Bell, and Z. Tian, *Detection, Estimation, and Modulation Theory, Part I*, 2nd. ed., Wiley, 2013. Steven Kay, *Fundamentals of Statistical Signal Processing, Vol I: Estimation Theory, Vol II: Detection Theory*, *Vol III: Practical Algorithm Development*, Prentice Hall, 1993/1998. ISBN-978-0-13-345711-7/978-0-13-504135-2/978-0-13-280803-3.

Optional Software: Scientific Python via the Jupyter Notebook/Lab (installed with https://anaconda.org/anaconda/python). See the second page of the syllabus. Mathematica, available to UCCS students (see course Web Site), is also very useful.

Grading:
1.) Graded homework assignments and computer projects worth 55%.
2.) Midterm exam (takehome) worth 20%.
3.) Final exam with computer sim problems (takehome) worth 25%.

Topics	Text Sections
1. Introduction and course overview	Notes ch 1
2. Background: discrete-time signal processing, linear algebra	2.1–2.4 (N ch2)
3. Random variables & discrete-time random processes	3.1–3.7 (N ch3)
4. Classical detection and estimation theory	N ch4
5. Overview of Rashka & Mirjalili, Python Machine Learning	Packt Pub.
6. The Levinson recursion	5.1–5.5 (N ch6)
7. Optimal filters including the Kalman filter; consider Phil Kim, Kalman Filtering for Beginners with MATLAB Examples.	7.1–7.5 (N ch8)
8. Spectrum estimation	8.1–8.8 (N ch9)
9. Adaptive filtering	9.1–9.4 (N ch10)

Drop Date The last day to drop is April 5, the Friday after *Spring break* week.

1.8 Course Syllabus-2

Installing Python & scikit-dsp-comm

Download and Install the Anaconda Python 3.6 Distribution: https://anaconda.org/anaconda/python. Do not choose the defaults! I recommend you have Anaconda configure paths to the tools. This is not the default.

Jupyter Notebook and QT Console will be installed with the Anaconda distribution, but more formation can be found at: http://ipython.org/.

Download and Install the distributed version control application git: https://git-scm.com/ on Windows systems (included on macOS and Linux).

Optionally Download and Install a code development environment that also integrates git version control such as Visual Studio Code (VS Code): https://code.visualstudio.com/ or PyCharm Community Edition IDE for Python: https://www.jetbrains.com/pycharm/.

Clone and Install the package scikit-dsp-comm by following the GitHub README page at: https://github.com/mwickert/scikit-dsp-comm.

Maintain scikit-dsp-comm using git pull origin master (see the information in the GitHub link).

Optional Jupyter Notebook to PDF Conversion

The Jupyter notebook is the perfect place to write code, document code, write text using markdown, import figures, and typeset math equations using LaTeX syntax. To render a Jupyter notebook as a PDF document a few more open source software components are needed:

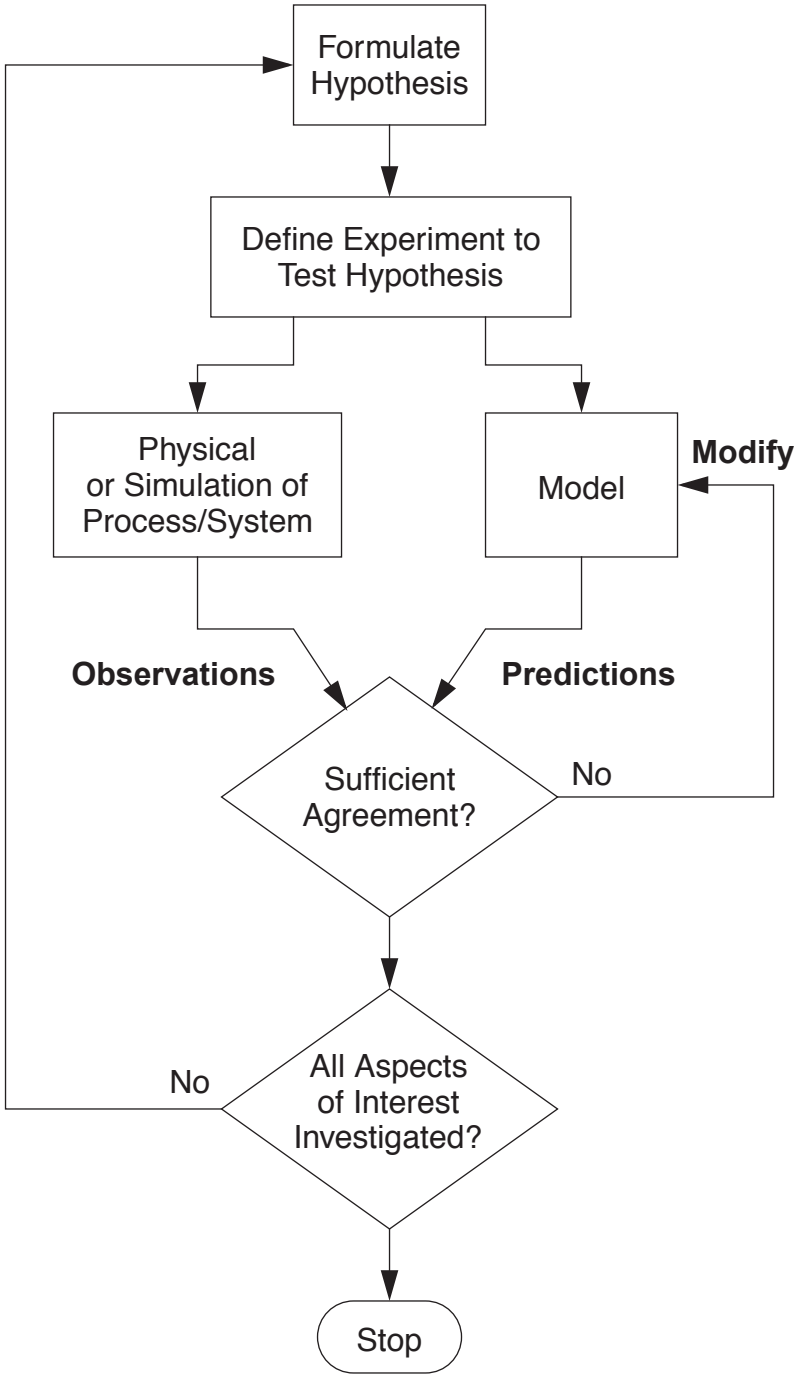
- **Install Pandoc** for file conversion to LaTeX and other formats: https://pandoc.org/index.html
- **Install MikTeX** for converting LaTeX documents to PDF on *Windows*: https://miktex.org/
- When installing MikTeX be sure to choose the option to automatically download needed LaTeX packages on-the-fly
- **Install TeXLive** for converting LaTeX documents to PDF on *macOS* and *Linux*: https://www.tug.org/texlive/
- **Install Inkscape** for converting embedded SVG graphics in Jupyter notebooks via Pandoc to LaTeX and then PDF: https://inkscape.org/en/release/0.92.2/. This gives you the ability to have nice looking graphics in the notebook and easily convert to a PDF, using just the File: Download Notebook menu item. On macOS you just install Inkscape. On Windows you may have to manually tweak the registry to get Inkscape to launch by the build script.

Install Typora: As an alternative to installing LaTeX (MikTeX or TeXLive), install the markdown editor Typora: https://typora.io/. Now you can export as *.md and then open the file in Typora and save to PDF directly. You can also do some nice file editing if need be.

1.9 Mathematical Models

- Mathematical models serve as tools in the analysis and design of complex systems
- A mathematical *model* is used to represent, in an approximate way, a physical process or system where measurable quantities are involved
- Typically a computer program is written to evaluate the mathematical model of the system and plot performance curves
  - The model can more rapidly answer questions about system performance than building expensive hardware prototypes
- Mathematical models may be developed with differing degrees of fidelity
- A system prototype is ultimately needed, but a computer simulation model may be the first step in this process
- A computer simulation model tries to accurately represent all relevant aspects of the system under study
- Digital signal processing (DSP) often plays an important role in the implementation of the simulation model
- If the system being simulated is to be DSP based itself, the simulation model may share code with the actual hardware prototype

- The mathematical model may employ both deterministic and random signal models



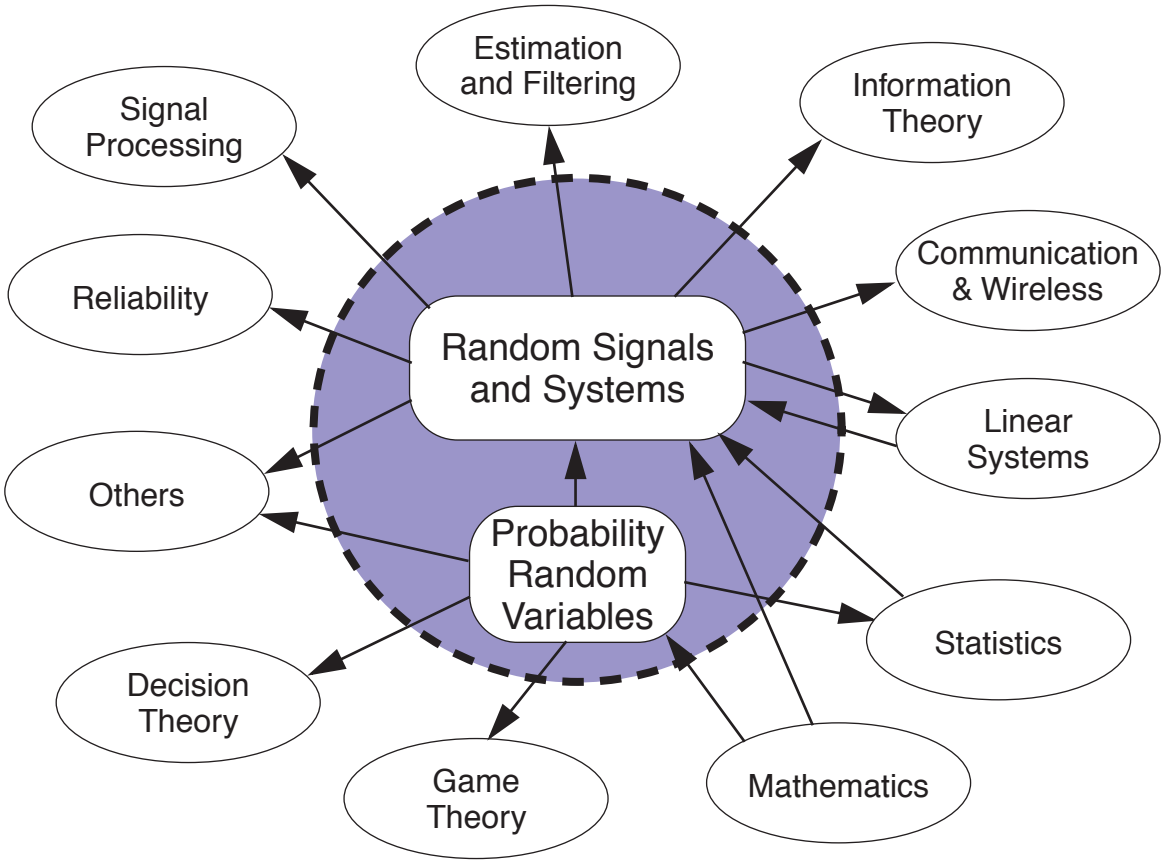
The Mathematical Modeling Process[2]

[2]Alberto Leon-Garcia, *Probability and Random Processes for Electrical Engineering*,

1.10 Engineering Applications

Communications, Computer networks, Decision theory and decision making, Estimation and filtering, Information processing, Power engineering, Quality control, Reliability, Signal detection, Signal and data processing, Stochastic systems, and others.

Relation to Other Subjects[3]



Addison-Wesley, Reading, MA, 1989

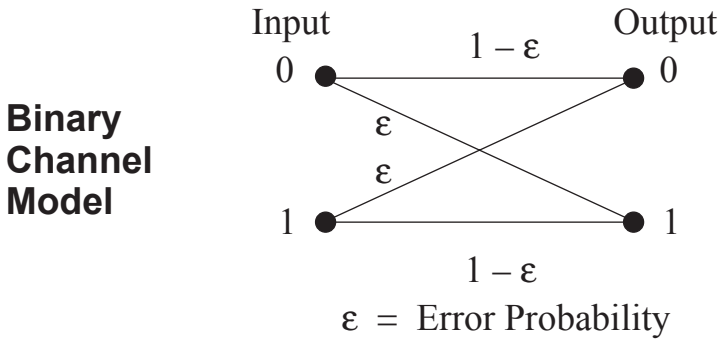
[3]X. Rong Li, *Probability, Random Signals, and Statistics*, CRC Press, Boca Raton, FL, 1999

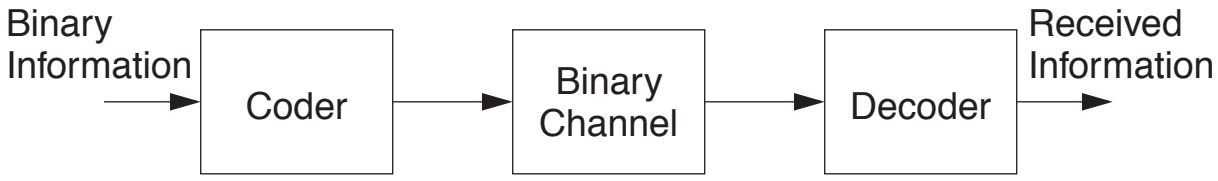
1.11 Random Signals and Statistical Signal Processing in Practice

- A typical application of random signals concepts involves one or more of the following:
  - Probability
  - Random variables
  - Random (stochastic) processes

Example 1.1: Modeling with Probability

- Consider a digital communication system with a *binary symmetric channel* and a coder and decoder





Communication System with Error Control

A data link with error correction

- The channel introduces bit errors with probability $P_e(\text{bit}) = \epsilon$
- A simple code scheme to combat channel errors is to *repetition code* the input bits by say sending each bit three times
- The decoder then decides which bit was sent by using a *majority vote* decision rule
- The system can tolerate one channel bit error without the decoder making an error
- The probability of a symbol error is given by

$$P_e(\text{symbol}) = P(2 \text{ bit errors}) + P(3 \text{ bit errors})$$

- Assuming bit errors are *statistically independent* we can write

$$\begin{aligned} P(2 \text{ bit errors}) &= \epsilon \cdot \epsilon \cdot (1-\epsilon) + \epsilon \cdot (1-\epsilon) \cdot \epsilon \\ &\quad + (1-\epsilon) \cdot \epsilon \cdot \epsilon = 3\epsilon^2(1-\epsilon) \\ P(3 \text{ errors}) &= \epsilon \cdot \epsilon \cdot \epsilon = \epsilon^3 \end{aligned}$$

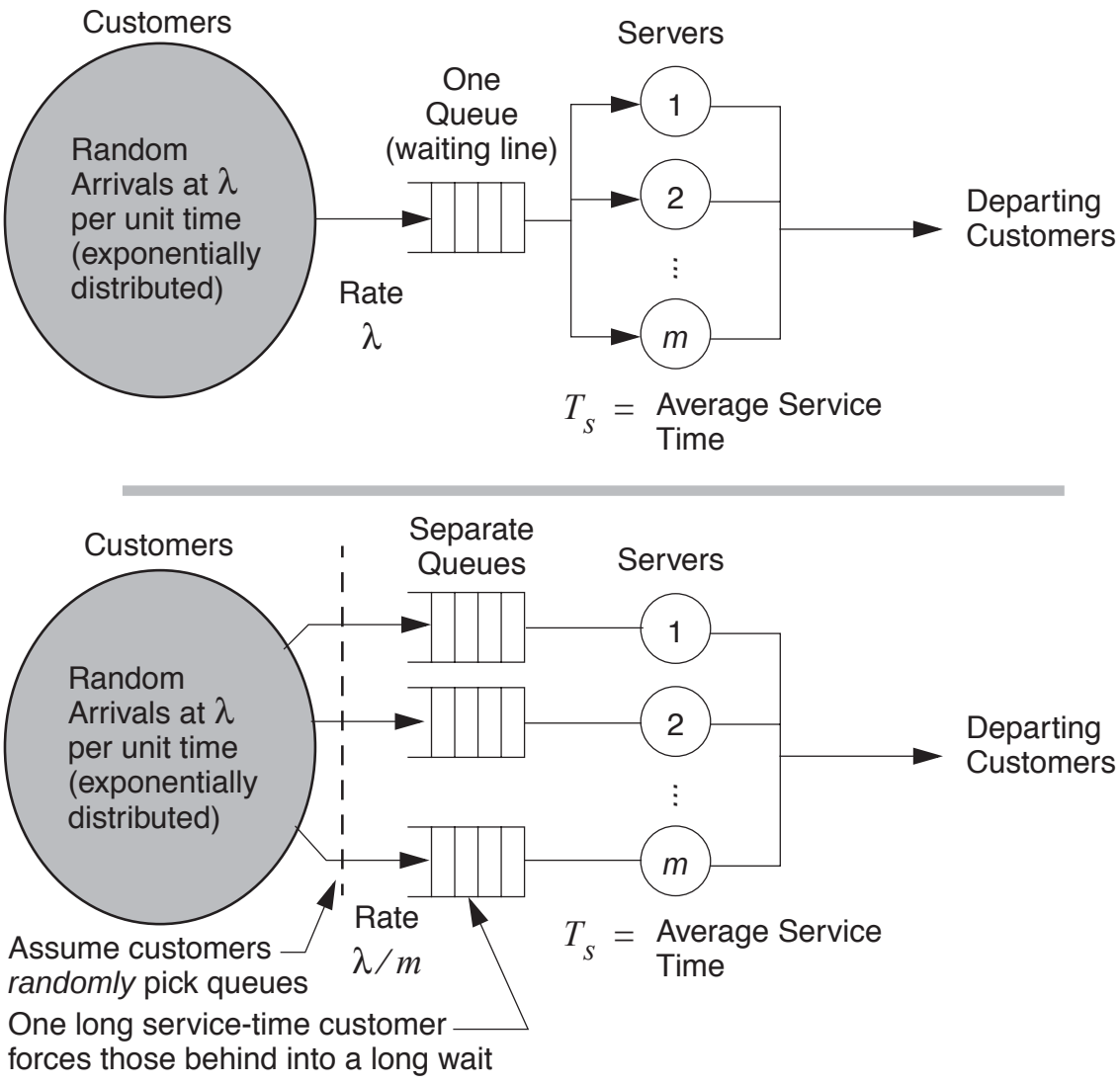
- The symbol error probability is thus

$$P_e(\text{symbol}) = 3\epsilon^2 - 2\epsilon^3$$

- Suppose $P_e(\text{bit}) = \epsilon = 10^{-3}$, then $P_e(\text{symbol}) = 2.998 \times 10^{-6}$
- The error probability is reduced by three orders of magnitude, but the coding reduces the throughput by a factor of three

Example 1.2: Separate Queues vs A Common Queue

A well known queuing theory result[4] is that multiple servers, with a common queue for all servers, gives better performance than multiple servers each having their own queue. It is interesting to see probability theory in action modeling a scenario we all deal with in our lives.



Common queue versus separate queues

[4] Mike Tanner, *Practical Queuing Analysis*, The IBM-McGraw-Hill Series, New York, 1995.

Common Queue Analysis

- The number of servers is defined to be m, the mean customer arrival rate is λ per unit of time, and the mean customer service time is T_s units of time
- In the single queue case we let $u = \lambda T_s =$ traffic intensity
- Let $\rho = u/m =$ server utilization
- For stability we must have $u < m$ and $\rho < 1$
- As a customer we are usually interested in the average time in the queue, which is defined as the waiting time plus the service time (Tanner)

$$T_Q^{\text{CQ}} = T_w + T_s = \frac{E_c(m,u)T_s}{m(1-\rho)} + T_s$$
$$= \frac{\left[E_c(m,u) + m(1-\rho)\right]T_s}{m(1-\rho)}$$

  where

$$E_c(m,u) = \frac{u^m/m!}{u^m/m! + (1-\rho)\sum_{k=0}^{m-1} u^k/k!}$$

  is known as the *Erlang-C formula*

- To keep this problem in terms of normalized time units, we will plot T_Q/T_s versus the traffic intensity $u = \lambda T_s$
- The normalized queuing time is

$$\frac{T_Q^{\text{CQ}}}{T_s} = \frac{E_c(m,u) + (m-u)}{m-u} = \frac{E_c(m,u)}{m-u} + 1$$

Separate Queue Analysis

- Since the customers randomly choose a queue, arrival rate into each queue is just λ/m
- The server utilization is $\rho = (\lambda/m) \cdot T_s$ which is the same as the single queue case
- The average queuing time is (Tanner)

$$T_Q^{\text{SQ}} = \frac{T_s}{1-\rho} = \frac{T_s}{1-\frac{\lambda T_s}{m}} = \frac{mT_s}{m-\lambda T_s}$$

- The normalized queuing time is

$$\frac{T_Q^{\text{SQ}}}{T_s} = \frac{m}{m-\lambda T_s} = \frac{m}{m-u}$$

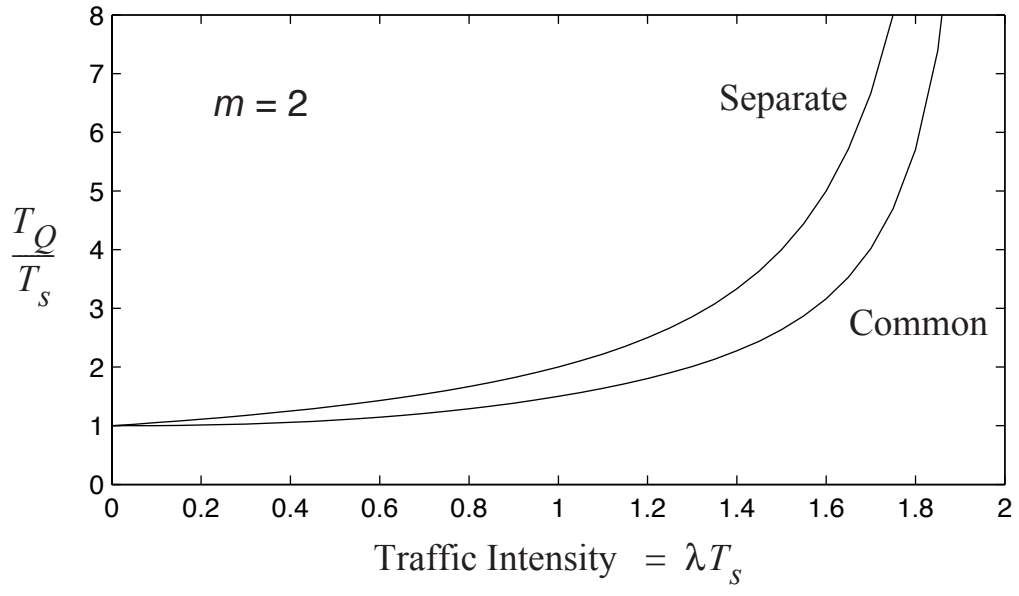
- Create the Erlang-C function in MATLAB:

```
function Ec = erlang_c(m,u);
% Ec = erlangc(m,u)
%
% The Erlang-C formula
%
% Mark Wickert 2001

s = zeros(size(u));
for k=0,
    s = s + u.^k/factorial(k);
end

Ec = (u.^m)/factorial(m)./(u.^m/factorial(m) + (1 - u/m).*s);
```

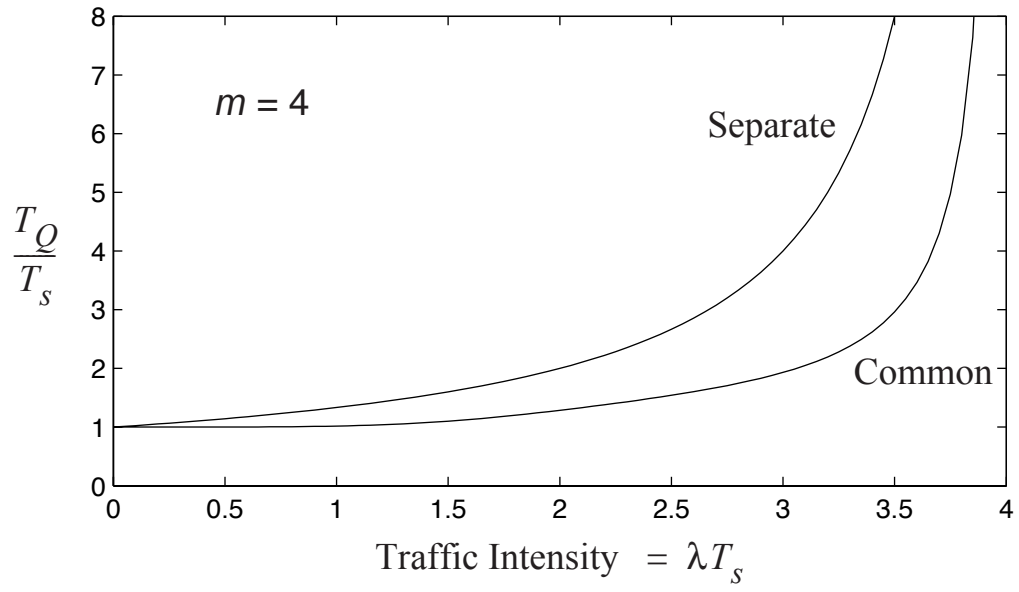
- We will plot T_Q/T_s versus $u = \lambda T_s$ for $m = 2$ and 4

```
>> % m = 2 case
>> u = 0:.05:1.9;
>> m = 2;
>> Tqsq = m./(m - u);
>> Tqcq = erlang_c(m,u)./(m-u) + 1;
>> % m = 4 case
>> u = 0:.05:3.9;
>> m = 4;
>> Tqsq = m./(m - u);
>> Tqcq = erlang_c(m,u)./(m-u) + 1;
```

- The plots are shown below:



Queuing time of common queue and separate queues for $m = 2$ servers



Queuing time of common queue and separate queues for $m = 4$ servers

Example 1.3: **Power Spectrum Estimation**

- The second generation wireless system *Global System for Mobile Communications* (GSM), uses the Gaussian minimum shift-keying (GMSK) modulation scheme

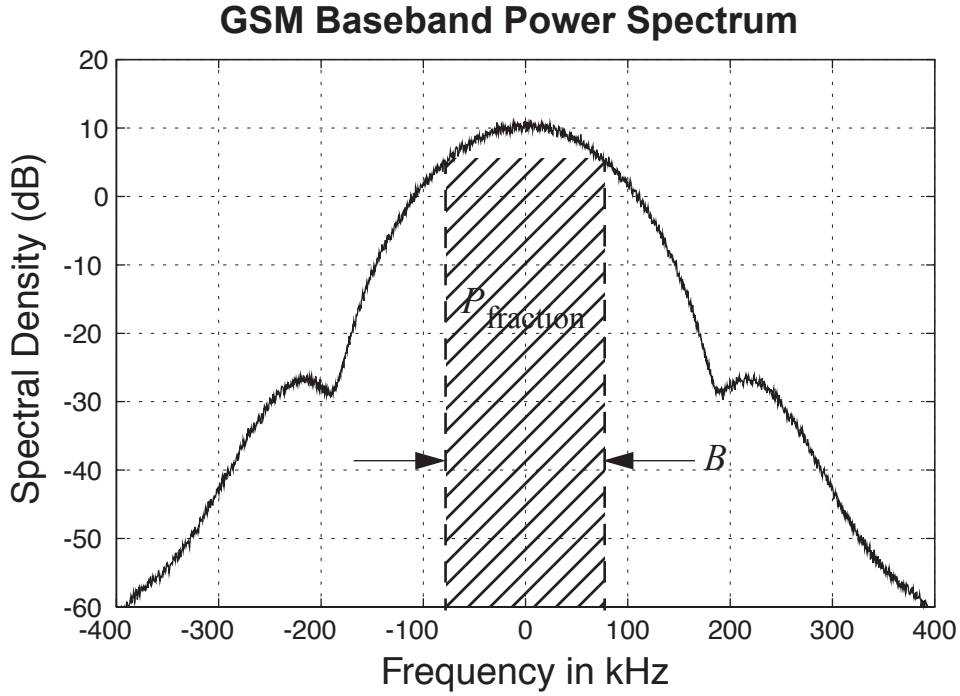
$$x_c(t) = \sqrt{2P_c}\cos\left[2\pi f_0 t + 2\pi f_d \sum_{n=-\infty}^{\infty} a_n g(t - nT_b)\right]$$

where

$$g(t) = \frac{1}{2}\left\{\text{erf}\left[-\sqrt{\frac{2}{\ln 2}}\pi B T_b\left(\frac{t}{T_b} - \frac{1}{2}\right)\right] + \text{erf}\left[\sqrt{\frac{2}{\ln 2}}\pi B T_b\left(\frac{t}{T_b} + \frac{1}{2}\right)\right]\right\}$$

- The GMSK shaping factor is $BT_b = 0.3$ and the bit rate is $R_b = 1/T_b = 270.833$ kbps
- We can model the baseband GSM signal as a complex random process
- Suppose we would like to obtain the fraction of GSM signal power contained in an RF bandwidth of B Hz centered about the carrier frequency
- There is no closed form expression for the power spectrum of a GMSK signal, but a simulation, constructed in MATLAB, can be used to produce a *complex baseband* version of the GSM signal

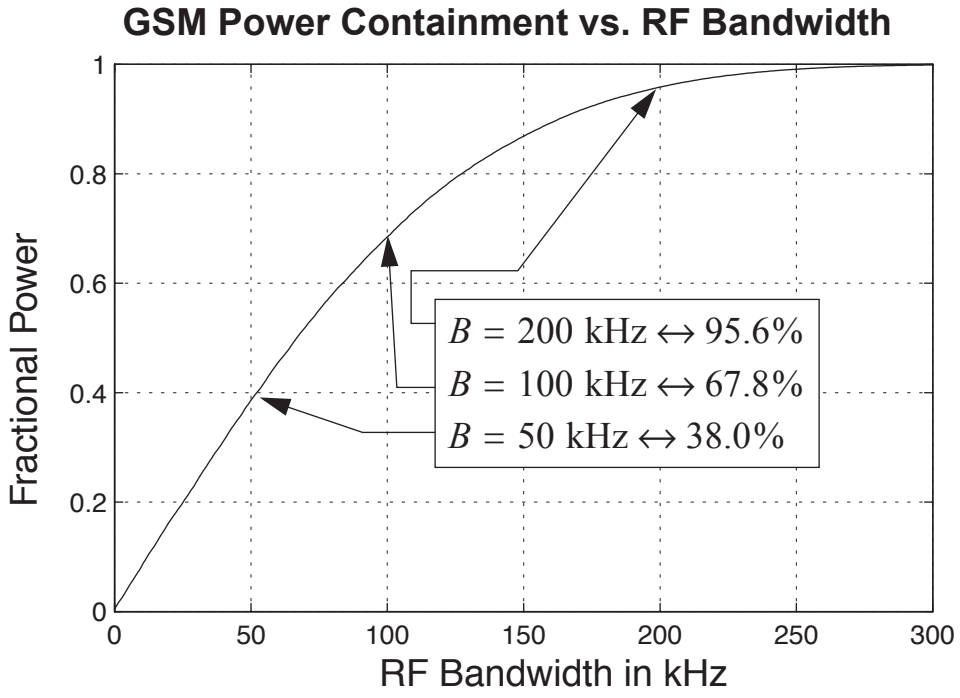
```
>> [x,data] = gmsk(0.3, 10000, 6, 16, 1);
>> [Px,F] = psd(x,1024,16);
>> [Pbb,Fbb] = bb_spec(Px,F,16);
>> plot(Fbb,10*log10(Pbb));  axis([-400 400 -60 20]);
```



- Using averaged periodogram spectral estimation we can estimate $S_{\text{GMSK}}(f)$ and then find the fractional power in any RF bandwidth, B, centered on the carrier

$$P_{\text{fraction}} = \frac{\int_{-B/2}^{B/2} S_{\text{GMSK}}(f)\,df}{\int_{-\infty}^{\infty} S_{\text{GMSK}}(f)\,df}$$

  - The integrals become finite sums in the MATLAB calculation



Fractional GSM signal power in a centered B Hz RF bandwidth

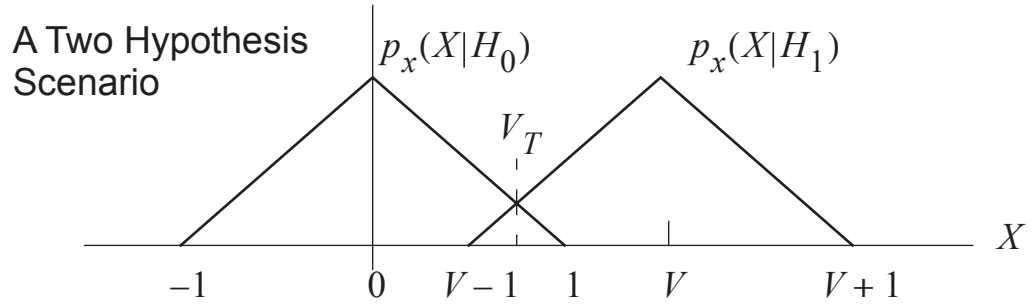
- An expected result is that most of the signal power (95%) is contained in a 200 kHz bandwidth, since the GSM channel spacing is 200 kHz

Example 1.4: A Simple Binary Detection Problem

- Signal x is measured as noise alone, or noise plus signal

$$x = \begin{cases} n, & \text{only noise for hypothesis } H_0 \\ V + n, & \text{noise + signal for hypothesis } H_1 \end{cases}$$

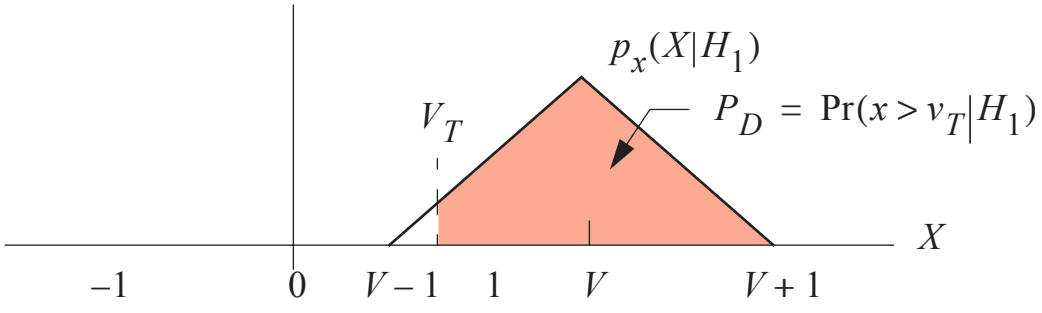
- We model x as a random variable with a *probability density function* dependent upon which hypothesis is present



- We decide that the hypothesis H_1 is present if $x > V_T$, where V_T is the so-called *decision threshold*

- The probability of detection is given by

$$P_D = \Pr(x > V_T|H_1) = \int_{V_T}^{\infty} p_x(X|H_1)\, dX$$

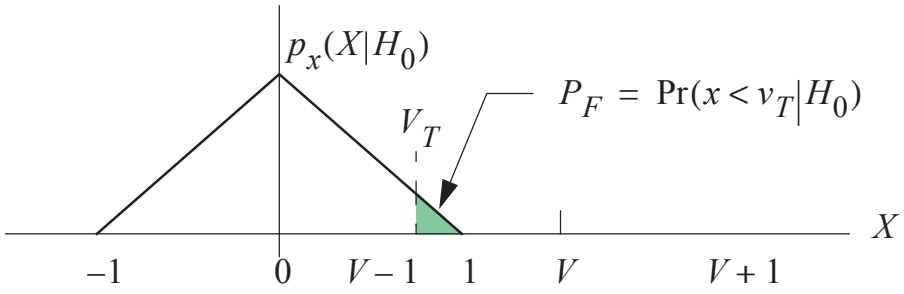


Area corresponding to P_D

- We may choose V_T such that the probability of *false alarm*, defined as

$$P_F = \Pr(x < V_T | H_0) = \int_{V_T}^{\infty} p_x(X|H_0)\, dX$$

is some desired value (typically small)



Area corresponding to P_F

Example 1.5: A Waveform Estimation Theory Problem

- Consider a received phase modulated signal of the form

$$r(t) = s(t) + n(t) = A_c \cos(2\pi f_c t + \theta(t)) + n(t)$$

where $\theta(t) = k_p m(t)$, k_p is the modulator phase deviation constant, $m(t)$ the modulation, and $n(t)$ is additive white Gaussian noise

- The signal we wish to estimate is $m(t)$, information phase modulated onto the carrier

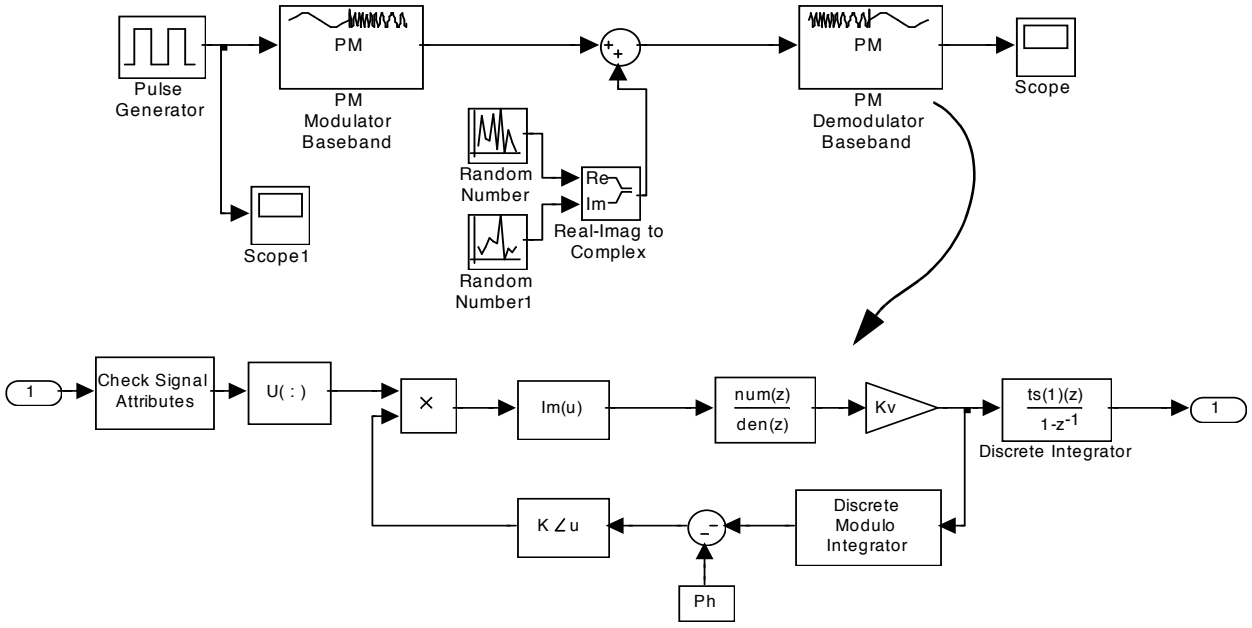
- We may choose to develop an estimation procedure that obtains $\hat{m}(t)$, the estimate of $m(t)$ such that the mean square error is minimized, i.e.,

$$\text{MSE} = E\{|m(t) - \hat{m}(t)|^2\}$$

- Another approach is *maximum likelihood estimation*

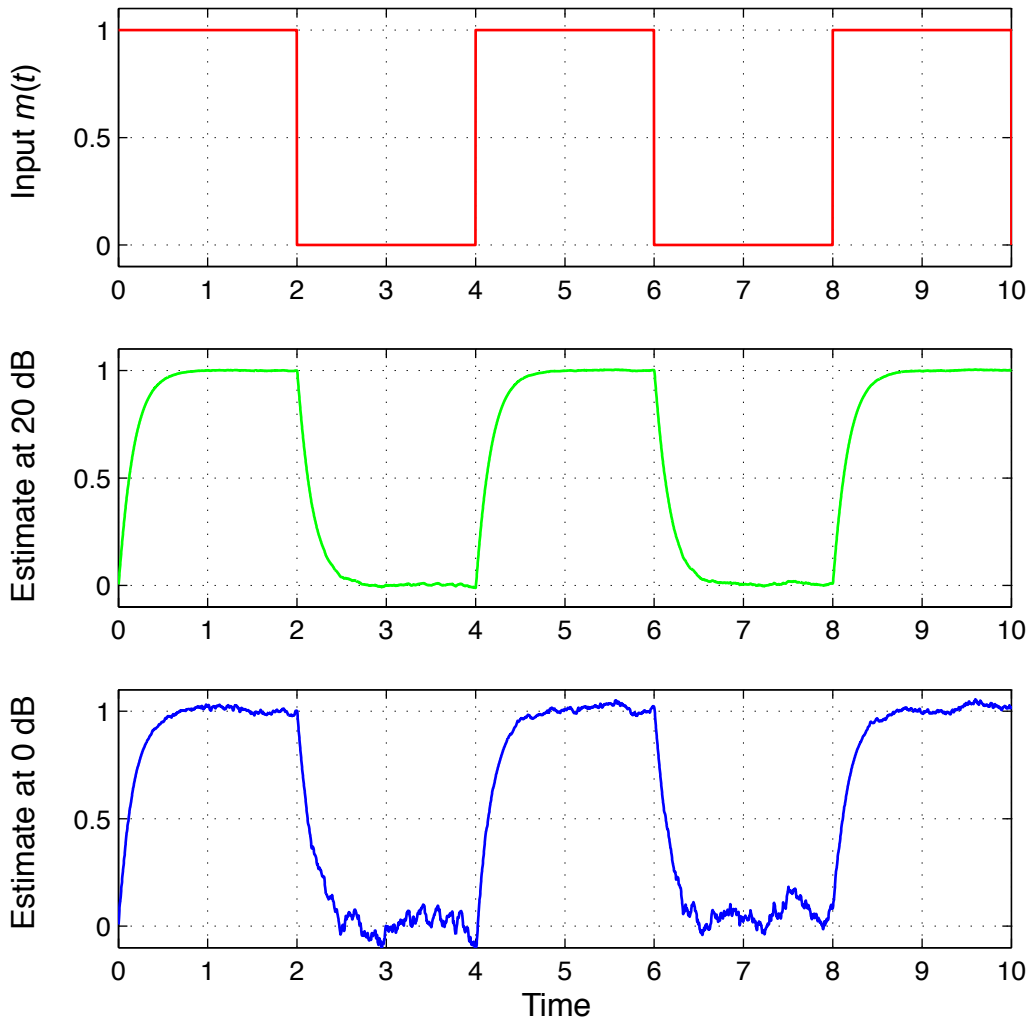
- A practical implementation of a maximum likelihood based scheme is a phase-locked loop (PLL)

- As a specific example here we consider the MATLAB Simulink block diagram shown below:



Complex baseband Simulink block diagram

- Results from the simulation using a squarewave input for two different signal-to-noise power ratios (20 dB and 0 dB) are shown below



Input/output waveforms

Example 1.6: Nonrandom Parameter in White Gaussian Noise

We are given observations

$$r_i = A + n_i, \; i = 1, \ldots, N$$

with the n_i iid of the form $N(0, \sigma_n^2)$

- The joint pdf is

$$p_{\mathbf{r}|a}(\mathbf{R}|A) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left[-\frac{(R_i - A)^2}{2\sigma_n^2}\right]$$

- The log likelihood function is

$$\ln p_{\mathbf{r}|a}(\mathbf{R}|A)] = \frac{N}{2} \ln\left(\frac{1}{2\pi\sigma_n^2}\right) - \frac{1}{2\sigma_n^2} \sum_{i=1}^{N} (R_i - A)^2$$

- Solve the likelihood equation

$$\left.\frac{\partial \ln[p_{\mathbf{r}|a}(\mathbf{R}|A)]}{\partial A}\right|_{A=\hat{a}_{\text{ml}}} = \left.\frac{1}{\sigma_n^2} \sum_{i=1}^{N} (R_i - A)\right|_{A=\hat{a}_{\text{ml}}} = 0$$

$$\boxed{\Longrightarrow \quad \hat{a}_{\text{ml}} = \frac{1}{N} \sum_{i=1}^{N} R_i}$$

- Check the bias:

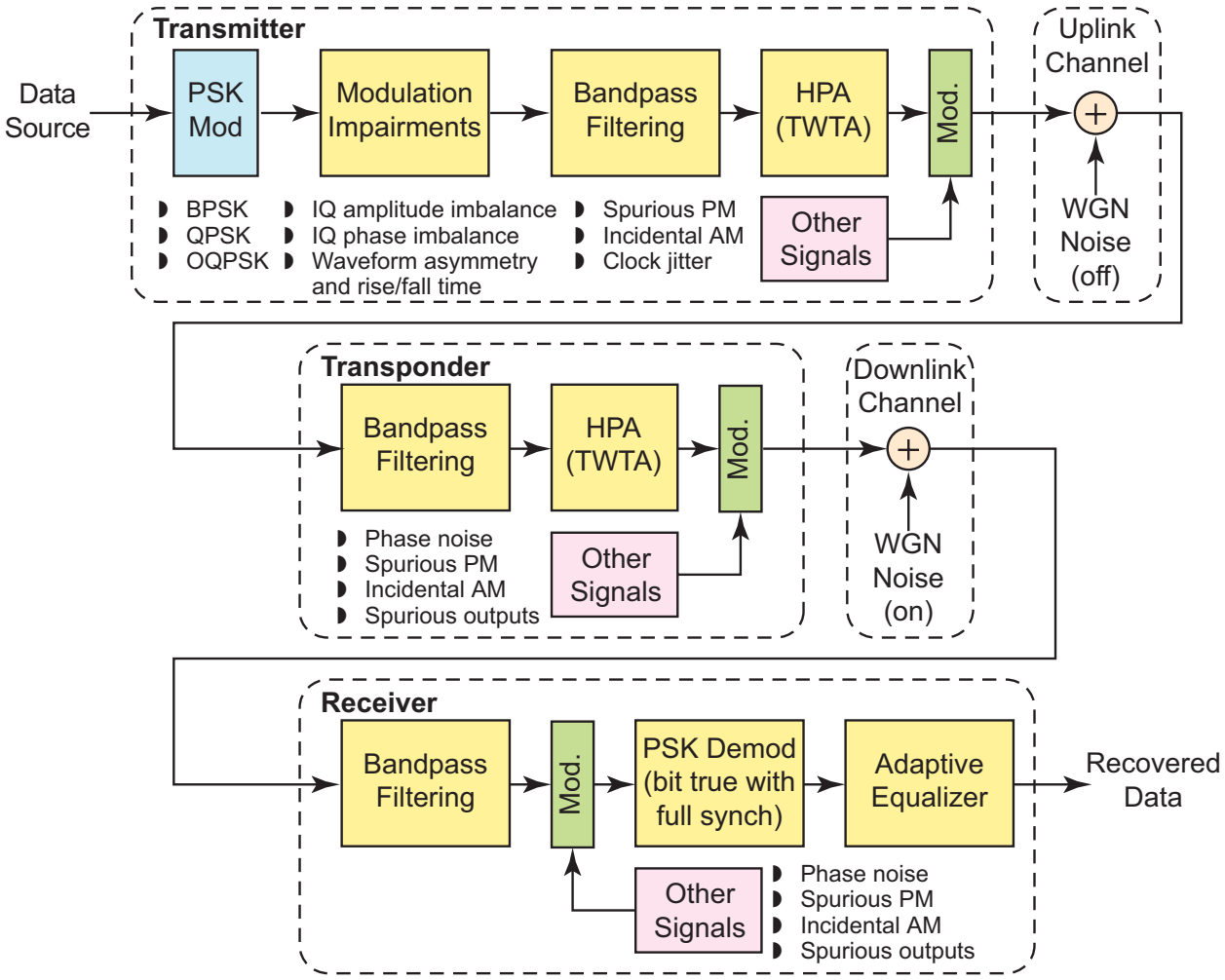
$$E\{\hat{a}_{\text{ml}}(\mathbf{r}) - A\} = E\left\{\frac{1}{N} \sum_{i=1}^{N} r_i - A\right\}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \underbrace{E\{r_i\}}_{A} - A = 0$$

No Bias!

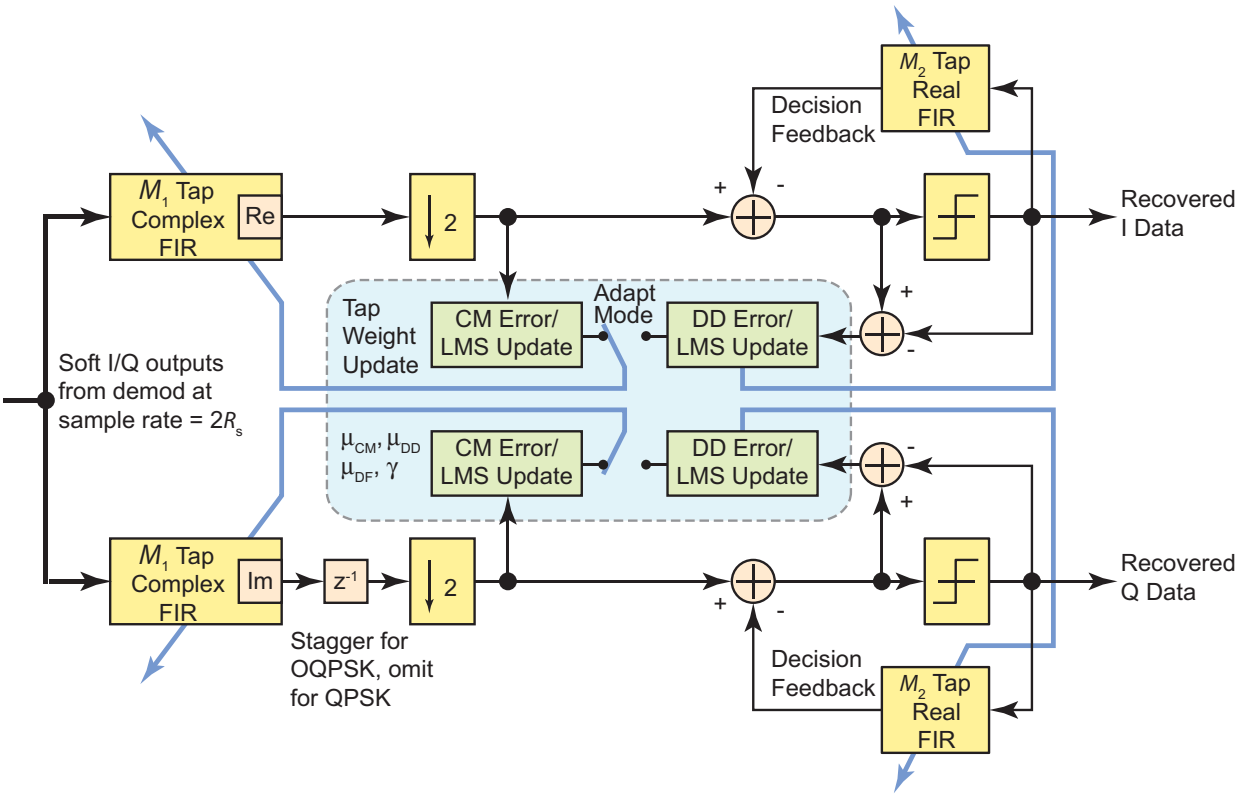
Example 1.7: Sat-Com Adaptive Equalizer

- Wideband satellite communication channels are subject to both linear and non-linear distortion



Wideband Sat-Comm simulation model

- An adaptive filter can be used to estimate the channel distortion, for example a technique known as *decision feedback equalization*

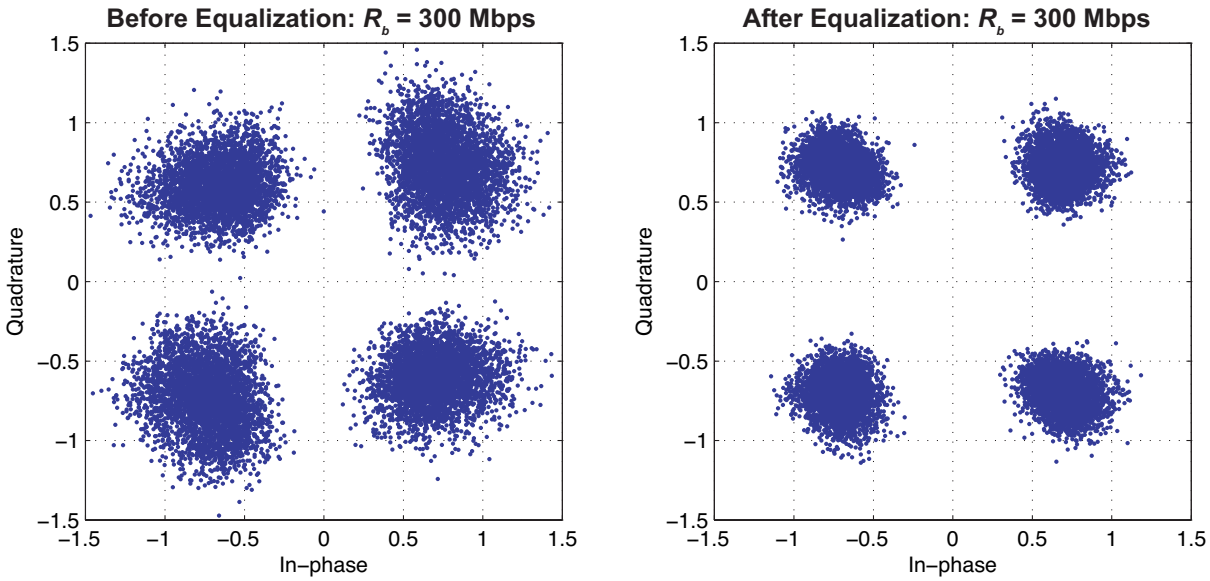


An adaptive baseband equalizer[5]

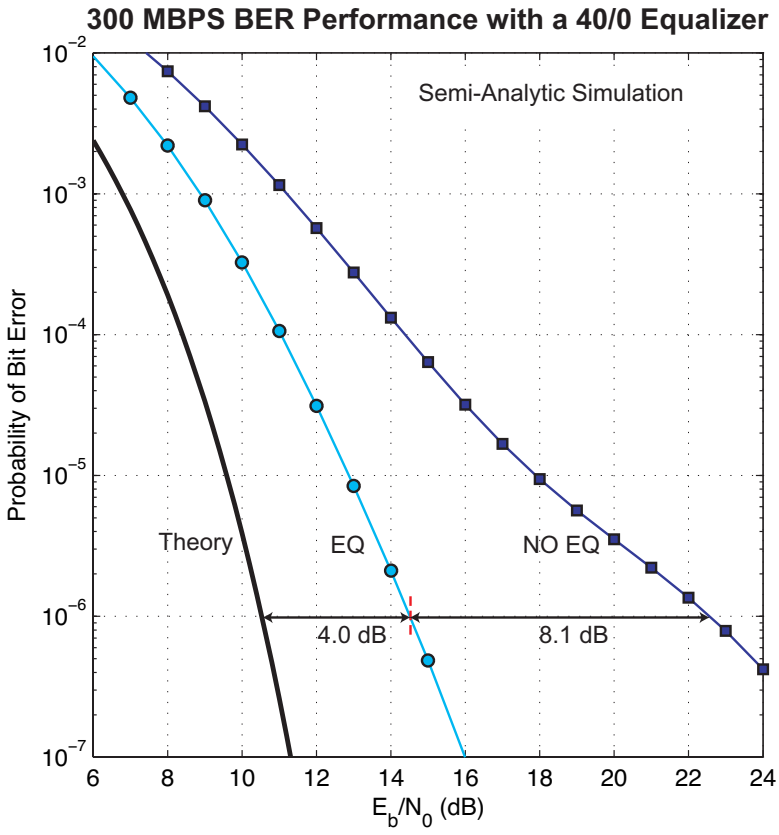
- Since the distortion is both linear (bandlimiting) and nonlinear (amplifiers and other interference), the distortion cannot be completely eliminated

- The following two figures show first the modulation 4-phase signal points with and with out the equalizer, and then the bit error probability (BEP) versus received energy per bit to noise power spectral density ratio (E_b/N_0)

[5]Mark Wickert, Shaheen Samad, and Bryan Butler. "An Adaptive Baseband Equalizer for High Data Rate Bandlimited Channels,ï£¡ *Proceedings 2006 International Telemetry Conference*, Session 5, paper 06–5-03.



OQPSK scatter plots with and without the equalizer



BEP versus E_b/N_0 in dB