2. GOALS OF DATA COMPRESSION

General Objective in data compression:

Given a signal and a channel, find an encoder and decoder that give the "best possible" replica of the signal at the user end or the most reliable recommendation to an action, such as accept a person to a secure area from his/her finger print or deny.

To formulate the task of data compression as a precise problem, we need:

- probabilistic descriptions of signal and channel (parametric model or sample training data),
- possible structural constraints on form of codes (block, sliding block, etc.),
- Quantifiable notion of what "good" or "bad" reconstruction is in the form of quantifiable measurements.

Some important questions:

Mathematical: Quantify what "best" or optimal achievable performance is.

Practical: How do you build systems that are good, if not optimal?

How to measure performance and need to define them?

- Signal-to-Noise Ration (SNR) and Segmented SNR (SegSNR)
- Peak SNR
- Mean-Square Error (MSE)
- Probability of bit error (*Pe*)
- Information transmission or storage bit rates,
- Complexity of the system designed,
- Cost of coding or desing.
- Subjective measures, such as Mean Opinion Scores (MOS).

Shannon Theory

Two publications by C.E. Shannon, "A Mathematical Theory of Communication (1948)" and "Coding Theorems for a Discrete Source with a Fidelity Criterion (1957)" have provided basis for application of probability theory to solving problems in communications in addition to establishing the foundations of the Science of Modern Information Theory.

The primary goal of Shannon's Information Theory is to find the theoretical limits to achievable performance. That is, given a probabilistic model of an information source and a communication channel (for storage or transmission), how reliably can the source signals be communicated to a receiver though the given channel?

Note that it does not tell you how to do it?

It leaves it to smart engineers to come up with good designs!

Key to Shannon's formulation was a family of definitions of the ***information*** content of signals in terms of probabilities. These are:

- Entropy $H(X)$ of a random variable X.
- Entropy rate $\overline{H}(X)$ of a sequence of random numbers (random process) $\{X_n\}$
- Mutual Information $I(X;Y)$ between two random variables X and Y.
- Notion of a ***distortion*** (error) measure $d(X,\hat{X})$ representing the cost of reproducing a signal $\hat{X}$ at the receiver (decoder) instead of a signal X of the source (encoder) so that the average distortion provides a measure of average signal quality and thus the system performance.

Definitions:

Entropy:

- Assume a source S that can generate any of *L* possible messages.
- Let each of these messages has a probability: $P_S = \{p_k; k = 0,1,\cdots,L-1\}$
- Entropy of this source represent the theoretically achievable lower bound on compression Systems and it is defined by:

$$H(S) = \sum_{k=0}^{L-1} p_k.\log_2(1/p_k) \qquad \text{Bits/symbol} \qquad (2.1)$$

- Let us assume that a given source has *L* different messages possible (symbol)
- The k^{th} symbol occurs with probability P_k, *k= 0,1,...,L-1.*
- Consider an encoder, which assigns a codeword with l_k bits to a particular symbol s_k.
- Average length of this source coder is simply:

$$L_{ave} = \sum_{k=0}^{L-1} p_k.l_k \qquad \text{Bits/symbol} \qquad (2.2)$$

Example 2.1: Consider the following 6 source symbols to be used as messages: *S*={A,B,C,D,E,F} with occurrence probabilities {0.25, 0.20, 0.16, 0.15, 0.13, 0.11}.

Note that sum of probabilities is 1.0 since there are no other messages and possibilities left. One way to represent this symbol set is to assign $\overline{L} = 3 - bit$ long codewords to each symbol as it is shown in Table 2.1.

- 8 possible ways to arrange 3 bits: 000, 001,…, 111.
- Chose any 6 out of these combinations.
- Compute the entropy and average length of this coding procedure.

$$H(S) = -\sum_{k=1}^{6} p_k.\log_2(p_k) = 2.5309 \quad bit/symbol$$

- It is easy to see that the difference $\overline{L} - H(S) = 3.0 - 2.5309 = 0.4691\, bits/symbol$ is almost ½ bits per symbol away from the theoretical bound *H(S).*

- Another scheme is to use a variable-length bit assignment according to some nice rule! As also shown un the table. One such assignment procedure is known as Huffman Coding in literature, which will be discussed later.
- The average length of this code is:

$$\bar{L}_{Huffman} = 0.25x2 + 0.20x2 + 0.16x3 + 0.15x3 + 0.13x3 + 0.11x3 = 2.55 \quad bits/symbol$$

- Difference in this case is lowered to:

$$\bar{L} - H(S) = 2.55 - 2.5309 = 0.0191\, bits/symbol$$

- This scheme results almost perfect result.

Message ID	Probability	Binary Code	Code Length	Huffmann Code	Code Length
A	0.25	000	3	10	2
B	0.20	001	3	00	2
C	0.16	010	3	111	3
D	0.15	011	3	110	3
E	0.13	100	3	011	3
F	0.11	101	3	010	3

Shannon Source Coding Theorem:

Given a source *S* with entropy *H(S)* then

1. It is possible to encode this source with a distortionless (error free) source coder with an average length L_{ave} provided

$$L_{ave} \geq H(S) \tag{2.3a}$$

2. Conversely, there is no distortionless source coder to encode *S* if

$$L_{ave} < H(S) \tag{2.3b}$$

Channel Capacity:

- Let a source emit symbols $S_0, S_1, \ldots, S_{L-1}$ at a rate R bits per second.
- Transmitted over a channel with a bandwidth B Hz.
- Receiver detects signals coming from the channel with a signal-to-noise ratio SNR= S/N,
- Where *S* and *N* are the signal and noise powers at the input to the receiver, respectively.
- Receiver issues symbols $Y_0, Y_1, \ldots, Y_{K-1}$. (These symbols $\{Y_k\}$ may or may not be identical to the source set $\{S_k\}$ depending upon the nature of the receiver.) Furthermore,
- *L* and *K* may be of different size. (Some codewords might be totally lost in the channel or some codewords might be added by the channel itself.)

If the channel is noiseless then

- L and K are identical and
- Receiver symbols are also same as the source symbols. In this case,
- Reception of some symbol Y_k uniquely determines the source symbol S_k.
- In the noisy channels, however, there is a certain amount of uncertainty regarding the identity of the transmitted symbol when Y_k is received.
- If the information channel has a bandwidth of B Hz. and
- System is designed to operate at a signal-to-noise ratio SNR then

The highest rate we can reliably transmit binary information is called **Capacity of a noisy channel** and it is defined by:

$$C = B.\log_2(1+SNR) \qquad \text{Bits/second} \qquad (2.4)$$

Shannon Channel Coding Theorem:

Given a channel with a capacity C, then

1. It is possible to transmit symbols from a source emitting at a rate R bits per second with an arbitrarily small probability of error in over this channel if

$$C \geq R \qquad (2.5a)$$

2. Conversely, all systems transmitting at a rate R such that

$$C < R \qquad (2.5b)$$

are bound to have errors with probability one, i.e, with hundred percent certainty to have errors.

Implication: We can have good source coders, even perfect ones, if the source rate is under the channel capacity. We may not be able design such good coder but that is our problem, not the information theory's! On the other hand, all coders are destined to make errors if they operate at a rate above the channel capacity.

Combination of these theorems shows that in the problem of Point-to-Point communication, the encoder (and decoder) can be decomposed into two separate pieces:

1. Source coding is the conversion of an information source into an efficient binary (or other digital) representation of rate R bits/s, with no regard for the channel except that its capacity is $C > R$ bits/s.

2. Channel coding or error control coding, takes the binary data stream of R bits/s and sends it reliably across the channel. This result is usually called the source/channel separation theorem. This result is not true in general in network communications, e.g., many sources to one receiver (multi-user channel) or one source to many receivers (broadcast channel).

- Source coding reduces the number of bits in order to save on transmission time or storage space. Compression \removes redundancy" to gain efficiency.
- Channel coding typically increases the number of bits or chooses different bits in order to protect against channel errors. Error control "adds redundancy" to permit the detection and correction of errors by looking for violation of known structures of transmitted signals and picking a likely fix.
- Overall communication requires a balance of the above two effects.

1. Consider a bandlimited channel (Bandwidth= W Hz) is used for transmitting signal with a total power S watts.
2. Assume that this channel is subject to additive Gaussian noise with a total noise power $N = 2W.N_0/2 = W.N_0$ watts. Here the noise level (density) is constant in the band of interest as $N_0/2$ watts/Hz.

The capacity of this channel is given by:

$$C = W.\log_2(1+\frac{S}{W.N_0}) \text{ Bits/second} \tag{2.6}$$

Fundamental Question: What does all this theory have to do with the real world?

- The theorem supports the intuitive practice that good overall systems can be designed by *separately* focusing on the source coding (compression) and channel coding (error control) problems, without concern of the interaction between the two.
- Joint coders are of increasing interest in networks and because they can yield simpler implementations.
- Theory provides benchmarks for comparison for real communication systems. It is impossible to do better than Shannon's bounds (unless you cheat). Thus if you are operating close to the Shannon limit, hardly worth doing any more.
- Most systems of the time fell far short of Shannon. However, few believed that they could ever get near. Over the years, methods evolved to come close to these limits in many applications of both source coding and channel coding.
- Emphasis through first half of Shannon era was on channel coding.
- Compression has historically played a secondary role, partially because of apparently relatively small potential gains and highly nonlinear nature in comparison with error control coding methods.

Question: Why to compress?

Answers:

1.Theory: As systems get better, even small gains get more important.

2.Practice: Sometimes have no choice, must compress or lose all the data and Gains are sometimes significant.

Consider the following simple image compression tasks:

- Low-resolution, TV quality, color video with 512x512 pixels/color at 8 bits/pixel, and 3 colors =~ 6 $x10^6$ bits
- 24 x 36 mm (35-mm) negative photograph scanned at 12 μm which is 3000 x 2000 pixels/color, 8 bits/pixel, and 3 colors =~144 x 10^6 bits
- 14 x 17 inch radiograph scanned at 70 μm: 5000 x 6000 pixels, at resolution 12 bits/pixel =~ 360 x 10^6 bits. Medium size hospital generates tera bytes each year.
- LANDSAT Thematic Mapper scene: 6000x6000 pixels/spectral band, 8 bits/pixel, and 6 non-thernmal spectral bands =~1:7 x 10^9 bits.

Therefore, data compression is required for efficient **transmission**

- to send more data in available bandwidth.
- to send the same data in less bandwidth

- more users on same same bandwidth

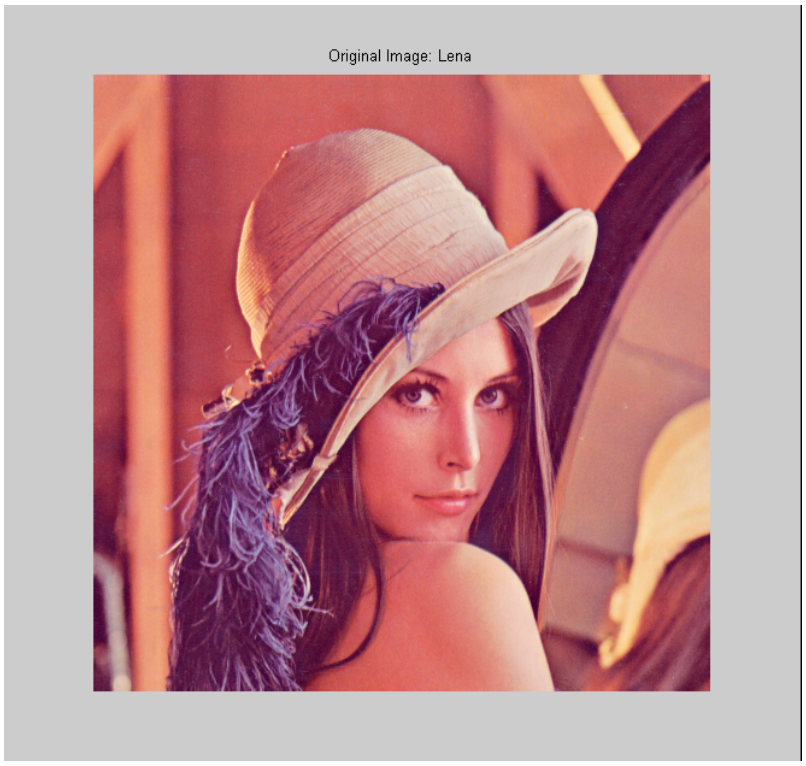
and **storage**
- to store more data
- to compress for local storage, and
- to put details on cheaper media

In both cases choice may be to compress or lose, e.g., you are allotted a given storage or bandwidth by an overall systems design, and your data rate is larger than the available storage or transmission capacity.
- Compression is also useful for progressive reconstruction, scalable delivery, browsing.
- It can also speed other signal processing by reducing the number of bits to be crunched (provided we have not lost essential information) or by combining with the compression algorithm, e.g.,
  - Enhancement
  - classification/detection
  - regression/estimation
  - filtering.

Example 2.2: Let us compute entropies of few grayscale (monochromatic, B/W) images as well as plotting histograms (computed probabilities) using a set of images from the Matlab database, which exhibit 256-levels of gray levels (8-bits), 0 being black and 255 white.

```
% Example using entropy.m to plot the histogram and compute entropy  of sample images.
% Note these 8-bit grayscale images are from the Matlab library.
% H is measure in bits per pixel
% Author: K. S. Thyagarajan; Modified by: H. Abut

A= imread('lena.ras');
figure, imshow(A), title('Original Image: Lena');

p=histogram(A);
figure, stem(p), title('Image Histogram')
xlabel('Pixel Value'),ylabel('Relative Frequency')

H=sum(-p .* log10(p+1e-06))/log10(2);
disp('Image Entropy in Bits/pixel: ');
disp(H);
```

```
Image Entropy in Bits/pixel:  7.7498
```


Original Image: Lena

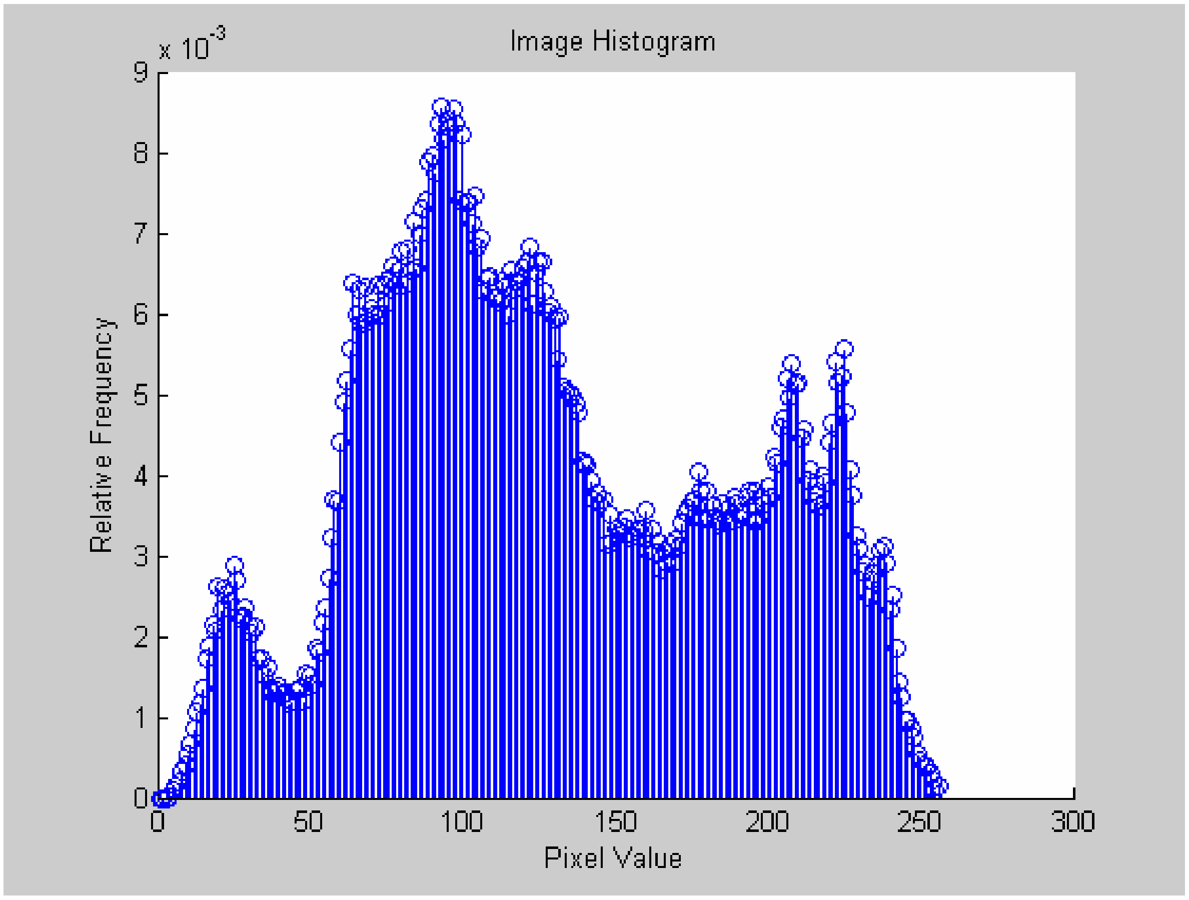


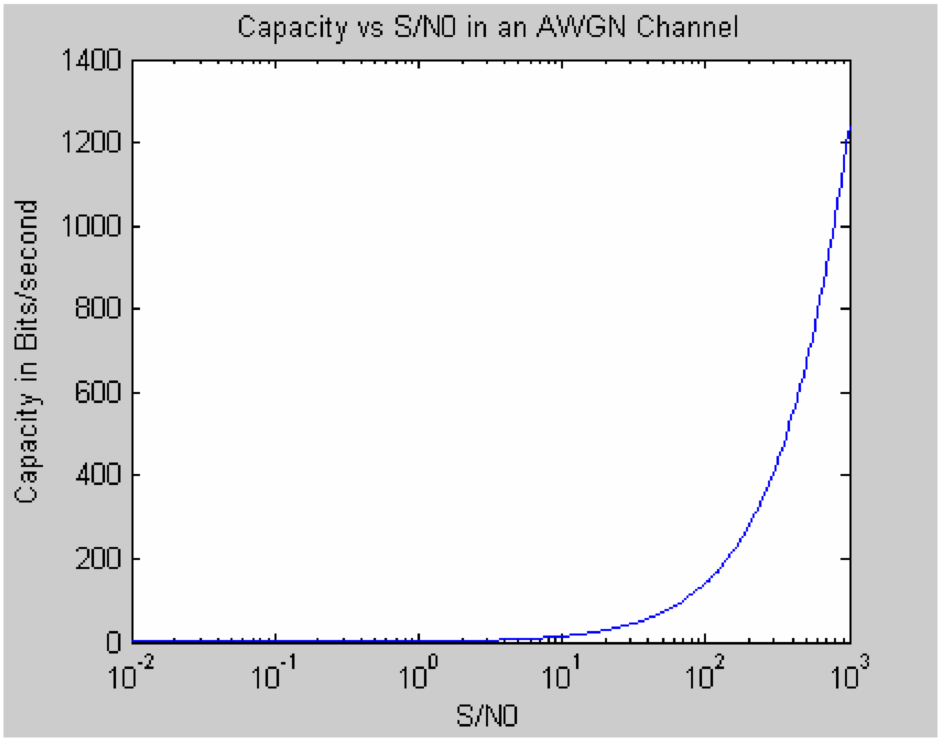
Image Histogram
x 10^-3
Relative Frequency
Pixel Value
0
1
2
3
4
5
6
7
8
9
0
50
100
150
200
250
300

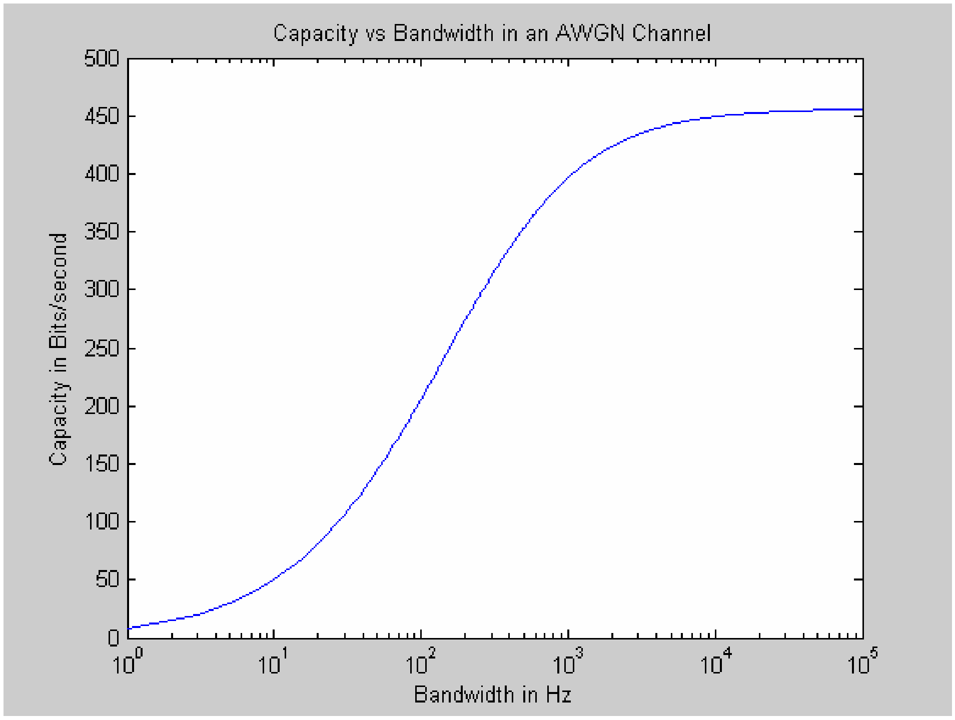
```
function [h,H]=histogram(A,L);
% HISTOGRAM Compute the histogram and cumulative histogram of A.
%    h = HISTOGRAM(A) computes the histogram assuming there are
%    256 integer grey-levels in A.
%    h = HISTOGRAM(A,L) computes the histogram using L integer
%    grey-levels in A.
%    [h,H] = HISTOGRAM(...) returns the histogram h and the
%    cumulative histogram H.
%    h is normalised so that the sum of the histogram is 1.
%    The maximum value of H is 1.
% M. Andrews 9 August 2000; Modified by H. Abut

% Create histogram bins
        if nargin == 1
          Bins=0:255;
        else
          Bins=0:L-1;
        end;

% Compute and normalise histogram
h=hist(A(:),Bins);
h=h/sum(h);

% Compute cumulative histogram
        if nargout == 2
          H=cumsum(h);
        end;
```

Example 2.3: Let us plot the capacity of an additive Gaussian channel with a bandwidth 3000 Hz as a function of $-20\,dB \le S/N_0 \le 30\,dB$. Repeat the experiment for a constant $S/N_0 = 25\,dB$ as a function of the bandwidth.

```
% Example 2.3 Channel Capacity Plots for a Additive Gaussian Noise Channel
% Programmed: Proakis, Salehi, Bauch
% Modified: H. Abut
echo on
bandwidth = 3000                    % Bandwith in Hz.
sovern0 = [-20:0.2:30];             % S/N0 in dB
sovern0 = 10 .^ (sovern0 ./ 10);    % S/N0 in none dB representation

capacity = bandwidth .* log2(1+sovern0/bandwidth);
semilogx(sovern0, capacity)
title('Capacity vs S/N0 in an AWGN Channel')
xlabel('S/N0')
ylabel('Capacity in Bits/second');

clear;
```

```
w=[1:10,12:2:100,105:5:500,510:10:5000,5025:25:20000,20050:50:100000];

sovern0=25;
sovern0 = 10 .^ (sovern0 ./ 10);

capacity = w .* log2(1+sovern0 ./ w);
figure;
semilogx(w,capacity)
title('Capacity vs Bandwidth in an AWGN Channel')
xlabel('Bandwidth in Hz')
ylabel('Capacity in Bits/second');

% Example 2.3 Channel Capacity Plots for a Additive Gaussian Noise Channel
% Programmed: Proakis, Salehi, Bauch
% Modified: H. Abut
echo on
bandwidth = 3000                    % Bandwith in Hz.
sovern0 = [-20:0.2:30];             % S/N0 in dB
sovern0 = 10 .^ (sovern0 ./ 10);    % S/N0 in none dB representation

capacity = bandwidth .* log2(1+sovern0/bandwidth);
semilogx(sovern0, capacity)
title('Capacity vs S/N0 in an AWGN Channel')
xlabel('S/N0')
ylabel('Capacity in Bits/second');

clear;
w=[1:10,12:2:100,105:5:500,510:10:5000,5025:25:20000,20050:50:100000];

sovern0=25;
sovern0 = 10 .^ (sovern0 ./ 10);

capacity = w .* log2(1+sovern0 ./ w);
figure;
semilogx(w,capacity)
title('Capacity vs Bandwidth in an AWGN Channel')
xlabel('Bandwidth in Hz')
ylabel('Capacity in Bits/second');
```


Capacity vs S/N0 in an AWGN Channel
Capacity in Bits/second
1400
1200
1000
800
600
400
200
0
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
S/N0


Capacity vs Bandwidth in an AWGN Channel
Capacity in Bits/second
500
450
400
350
300
250
200
150
100
50
0
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
Bandwidth in Hz