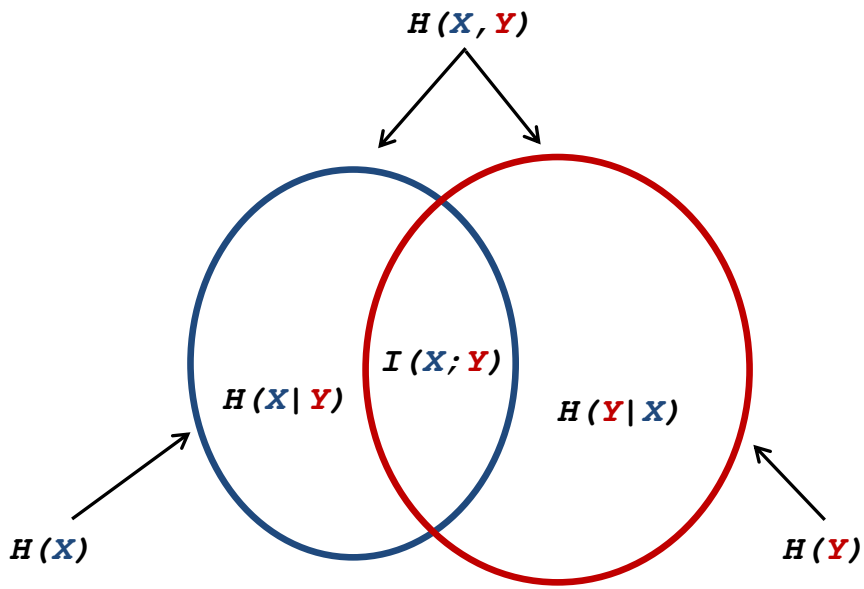
Information Theory

Professor John Daugman

University of Cambridge

Computer Science Tripos, Part II
Michaelmas Term 2020/21



Outline

1. **Foundations: probability, uncertainty, information.**
2. **Entropies defined, and why they are measures of information.**
3. **Source coding theorem; prefix, variable-, and fixed-length codes.**
4. **Discrete channel properties, noise, and channel capacity.**
5. **Information represented in vector spaces by projections.**
6. **Fourier representations of information.**
7. **Spectral properties of continuous-time signals and channels.**
8. **Continuous information; density; noisy channel coding theorem.**
9. **Signal coding and transmission schemes using Fourier theorems.**
10. **The quantised degrees-of-freedom in a continuous signal.**
11. **Gabor-Heisenberg-Weyl uncertainty relation. Optimal "Logons".**
12. **Data compression codes and protocols.**
13. **Discrete, and Fast Fourier Transform algorithms.**
14. **Wavelet representations of information.**
15. **Kolmogorov complexity. Minimal description length.**
16. **Applications of information theory in other sciences.**

Reference book

Cover, T & Thomas, J (2006) *Elements of Information Theory (2nd ed)*. Wiley.

Overview: what is information theory?

Key idea: The movements and transformations of information, just like those of a fluid, are constrained by mathematical and physical laws. These laws have deep connections with:

- probability theory, statistics, and combinatorics
- thermodynamics (statistical physics)
- spectral analysis, Fourier (and other) transforms
- sampling theory, estimation, prediction, and machine learning
- electrical engineering (bandwidth; signal-to-noise ratio)
- complexity theory (minimal description length)
- signal processing, compressibility, and pattern analysis

Two questions limiting all data encoding and communication systems:

1. What is the ultimate data compression?
   (answer: the entropy of the data, H, is its compression limit.)
2. What is the ultimate rate of reliable communication?
   (answer: the channel capacity, C, is its transmission rate limit.)

Information theory studies ways to achieve these two theoretical limits; but of greater relevance today are connections with machine learning.

Important questions to which information theory offers answers:

- How should information be measured?
- What can be learned from data, and how much additional information is gained by some reduction in uncertainty?
- How do the *a priori* probabilities of possible messages determine the informativeness of receiving them?
- What is the information content of a random variable?
- How does the noise level in a communication channel limit its capacity to transmit information?
- How does the bandwidth (in Hertz) of a communication channel limit its capacity to transmit information?
- By what formalism should prior knowledge be combined with incoming data to draw formally justifiable inferences from both?
- How resistant is a cryptographic key to a brute force attack?
- How much information in contained in a strand of DNA?
- How much information is there in the firing pattern of a neurone?

1. Foundations: probability, uncertainty, information

- *How the concepts of randomness, redundancy, compressibility, noise, bandwidth, and uncertainty are related to information.*
- *Ensembles, random variables, marginal and conditional probabilities.*
- *How metrics of information are grounded in the rules of probability.*

The entropy of a random variable is the average level of uncertainty that is removed, or information gained, inherently in its possible outcomes.

Random variables are just variables that take on values determined by probability distributions. They may be discrete or continuous, in their domain or in their range.

For example, a stream of ASCII encoded text characters in a transmitted message is a discrete random variable, taking on discrete values that have a known probability distribution for any given natural language.

An analog speech signal represented by a voltage or sound pressure waveform as a function of time (perhaps with added noise), is an example of a continuous random variable described by a continuous probability density function.

Most of Information Theory involves probability distributions of random variables, and conjoint or conditional probabilities defined over ensembles of random variables. Indeed, the information content of a symbol or event is defined in terms of its (im)probability.

Classically, there are two different points of view about what probability actually means:

- *relative frequency:* sample the random variable a great many times and tally up the fraction of times that each of its different possible values occurs, to arrive at the probability of each.

- *degree-of-belief:* probability is the plausibility of a proposition or the likelihood that a particular state (or value of a random variable) might occur, even if its outcome can only be decided once (such as the outcome of a particular horse-race).

The first view, the "frequentist" or operationalist view, is the one that predominates in statistics and in information theory. However, it does not capture the full meaning of probability.

For example, the proposition `"The moon is made of green cheese"` is one which surely has a probability that we should be able to attach to it. We could assess its probability by degree-of-belief calculations which combine our prior knowledge about physics, geology, and dairy products.

Yet it seems the "frequentist" definition of probability could only assign a probability to this proposition by performing (say) a large number of repeated trips to the moon, and tallying up the fraction of trips in which the moon turned out to be a dairy product....

In either case, it seems sensible that the less probable an event is, the more information is gained by noting its occurrence. (Surely discovering that the moon IS made of green cheese would be more "informative" than merely learning that it is made of earth-like minerals and rocks.)

Probability rules

Most of probability theory was laid down by theologians: Blaise PASCAL (1623-1662) who gave it the axiomatization that we accept today; and Thomas BAYES (1702-1761) who expressed one of its most important and widely-applied propositions relating conditional probabilities.

Probability Theory rests upon two rules:

Product Rule:

$$p(A, B) = \text{“joint probability of } \textit{both } A \textit{ and } B\text{”} = p(A|B)p(B)$$

or equivalently,

$$= p(B|A)p(A)$$

Clearly, in case A and B are *independent* events, they are not conditionalised on each other and so

$$p(A|B) = p(A) \quad \text{and} \quad p(B|A) = p(B),$$

in which case their joint probability is simply $p(A, B) = p(A)p(B)$.

Sum Rule:

If event A is conditionalised on a number of other events B, then the total probability of A is the sum of its joint probabilities with all B:

$$p(A) = \sum_B p(A, B) = \sum_B p(A|B)p(B)$$

From the Product Rule and the symmetry that $p(A, B) = p(B, A)$, it is clear that $p(A|B)p(B) = p(B|A)p(A)$. Bayes' Theorem then follows:

Bayes' Theorem:

$$p(B|A) = \frac{p(A|B)p(B)}{p(A)}$$

The importance of Bayes' Theorem is that it allows us to reverse the conditionalising of events, and to compute $p(B|A)$ from knowledge of $p(A|B)$, $p(A)$, and $p(B)$. Often these are expressed as *prior* and *posterior* probabilities, or as the conditionalising of hypotheses upon data.

Worked Example:

Suppose that a dread disease affects 1/1000th of all people. If you actually have the disease, a test for it is positive 95% of the time, and negative 5% of the time. But if you don't have the disease, the test is positive 5% of the time. We wish to know how to interpret test results.

Suppose you take the test and it is positive. What is the likelihood that you actually have the disease?

We use the above rules, with the following substitutions of "data" D and "hypothesis" H instead of A and B:

D = data: the test is positive
H = hypothesis: you have the disease
$\bar{H}$ = the other hypothesis: you do not have the disease

Before acquiring the data, we know only that the *a priori* probability of having the disease is .001, which gives us $p(H)$. This is called a *prior*. We also need to know $p(D)$.

From the Sum Rule, we can calculate that the *a priori* probability $p(D)$ of testing positive, whatever the truth may actually be, is:

$$p(D) = p(D|H)p(H) + p(D|\bar{H})p(\bar{H}) = (.95)(.001) + (.05)(.999) = .051$$

and from Bayes' Rule, we can conclude that the probability that you actually have the disease given that you tested positive for it, is much smaller than you may have thought:

$$p(H|D) = \frac{p(D|H)p(H)}{p(D)} = \frac{(.95)(.001)}{(.051)} = \boxed{0.019} \qquad \text{(less than 2\%)}.$$

This quantity is called the *posterior probability* because it is computed after the observation of data; it tells us how likely the hypothesis is, given what we have observed.

(Note: it is an extremely common human fallacy to confound $p(H|D)$ with $p(D|H)$. In the example given, many people would react to the positive test result by concluding that the likelihood that they have the disease is .95, since that is the "hit rate" of the test. They confound $p(D|H) = .95$ with $p(H|D) = .019$, which is what actually matters.)

Interchanging two conditional probabilities is one cause of common prejudices. For example, consider the random variables T and M:

$$T := \text{“this person is a terrorist”}$$
$$M := \text{“this person is a Muslim”}$$

Tabloid newspapers and their readers confound *“if terrorist, then Muslim”* with: *“if Muslim, then terrorist”*. Approximate frequencies today may be: $p(M|T) \approx 0.95$ while $p(T|M) \approx 0.0000001$ (about 100 in a billion).

A nice feature of Bayes' Theorem is that it provides a simple mechanism for repeatedly updating our assessment of some hypothesis as more data continues to arrive. We can apply the rule recursively, using the latest *posterior* as the new *prior* for interpreting the next set of data. In Artificial Intelligence, this feature is important because it allows the systematic and real-time construction of interpretations that can be updated continuously as more data arrive in a time series, such as a stream of images or spoken sounds that we wish to understand.

Information Theory allows us to analyse quantitatively the amount of uncertainty that is reduced, *i.e.* the amount of information that is gained, from an inference using Bayes' Theorem. Now we must develop such metrics that operate on probabilities.

2. Entropies defined, and why they are measures of information.

- *Marginal entropy, joint entropy, conditional entropies, and the Chain Rule for entropy. The "distance" between random variables.*
- *Mutual information between random variables. Independence.*

The information measure I of a single event or message is defined as the base-2 logarithm of its probability p of occuring:

$$I = \log_2 p$$

and its *entropy* H is considered the inverse: $H = -I$. Entropy can be regarded intuitively as "uncertainty," or "disorder." To gain information is to lose uncertainty by the same amount, so I and H differ only in sign. Entropy and information have units of *bits.*

Note that $I = \log_2 p$ is never positive: it ranges from 0 to $-\infty$, and thus H ranges positively from 0 to $+\infty$, as p varies from 1 to 0.

No information is gained (no uncertainty is reduced) by the appearance of an event or the receipt of a message that was completely certain anyway: $p = 1$, therefore $H = I = 0$. Intuitively, the more improbable an event is, the more significant it is; so the monotonic behaviour seems appropriate.

But why the logarithm?

The logarithmic measure is justified by the desire that information be additive. We want the algebra of our measures to reflect the rules of probability. When independent packets of information arrive, we would like to say that the total information received is the sum of the individual pieces. But the probabilities of independent events multiply to give their combined probabilities, and so we must take logarithms in order for the joint probability of independent events or messages to be combined additively into the total information gained.

This principle can also be understood in terms of the combinatorics of state spaces. Suppose we have two independent problems, one with n possible solutions (or states) each having probability p_n, and the other with m possible solutions (or states) each having probability p_m. Then the number of combined states is mn, and each of these has probability $p_m p_n$. We would like to say that the information gained by specifying the solution to *both* problems is the *sum* of that gained from each one. This desired property is achieved:

$$I_{mn} = \log_2(p_m p_n) = \log_2 p_m + \log_2 p_n = I_m + I_n$$

A note on logarithms:

In information theory we often wish to compute the base-2 logarithms of quantities, but many calculating tools only offer Napierian (base 2.718...) or decimal (base 10) logarithms. So the following conversions are useful:

$$\log_2 X = 1.443 \log_e X = 3.322 \log_{10} X$$

It may be noted in passing that occasionally the "natural" or Naperian (base-e) logarithm is invoked, in which case the information measure is the "nat" or the "nit" (for Naperian bit). Thus 1 bit $\approx$ 0.693 nits.

Henceforward we will omit the subscript; base-2 is always presumed.

You will find it very beneficial to commit to memory now all of the powers of 2 from about -8 to +8 (*i.e.* $\frac{1}{256}$, $\frac{1}{128}$, $\frac{1}{64}$, $\frac{1}{32}$, ..., 1, 2, 4, ..., 128, 256) because we will frequently encounter such numbers, and their base-2 logarithms should be immediately at your fingertips.

Intuitive Example of the Information Measure:

Suppose I select at random one of the 26 letters of the alphabet, and we play the game of "26 questions" in which you try to discover which letter I have chosen. I will only answer 'yes' or 'no,' always truthfully.

What is the minimum number of such questions that you must ask me, in order to guarantee finding the answer? - Perhaps 25 questions?

What form should such questions take? *e.g.*, "Is it A?" ... "Is it B?" ... or, is there some more intelligent way to solve this problem?

The answer to a Yes/No question having equal probabilities conveys one bit worth of information. In the above example with equiprobable states, you never need to ask more than 5 (well-phrased!) questions to discover the answer, even though there are 26 possibilities.

The information measure tells us that the uncertainty removed as a result of solving this problem corresponds to about 4.7 bits.

Entropy of ensembles

We now move from considering the information content of a single event or message, to that of an *ensemble.* An ensemble is the set of outcomes of one or more random variables. The outcomes have known probabilities attached to them. In general these probabilities are non-uniform, with event i having probability p_i, but they must sum to 1 because all possible outcomes are included; hence they form a discrete probability distribution:

$$\sum_i p_i = 1$$

The *entropy of an ensemble* is simply the average entropy of all of the elements in it. We can compute their average entropy by weighting each of the $\log p_i$ contributions by its probability p_i of occuring:

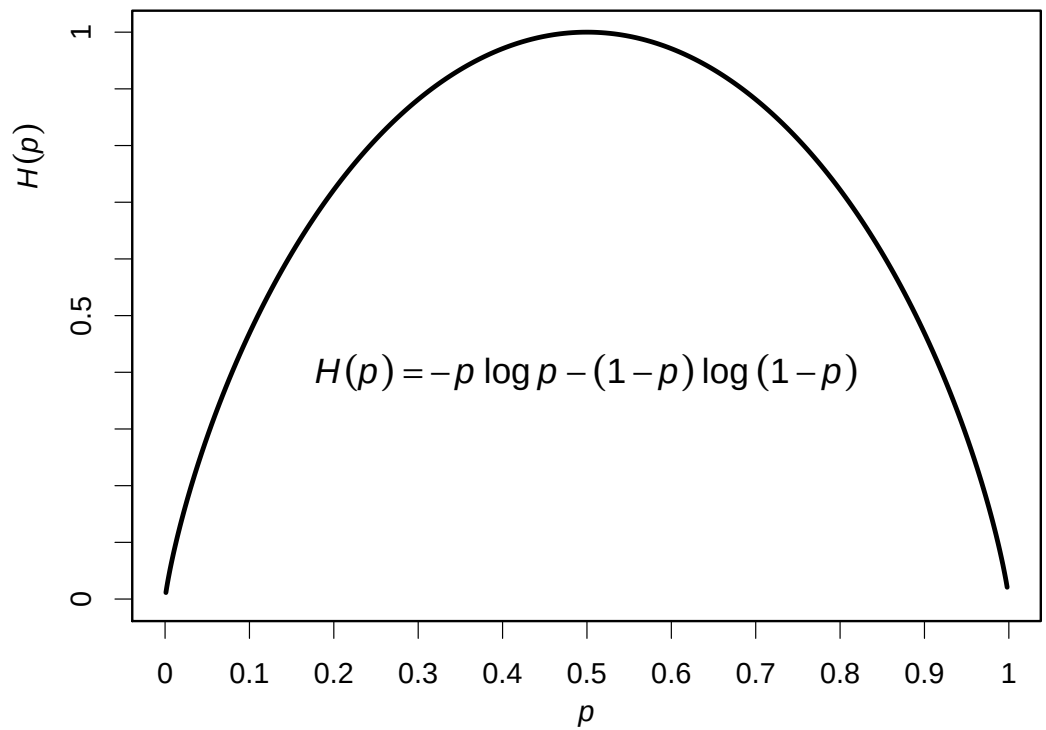
$$H = -I = -\sum_i p_i \log p_i$$

Thus we can speak of the information content or the entropy of a random variable, from knowledge of the probability distribution that it obeys. *(Entropy does not depend upon the actual values taken by the random variable! – Only upon their relative probabilities.)*

Entropy of a binary random variable with outcome probability p

Let us consider a random variable that takes only two values, one with probability p, and the other one with probability $1 - p$. (This is called a Bernoulli process, with parameter p.) How does the entropy of this binary random variable depend on the value of p?

Plotting $H = -\sum_i p_i \log p_i$ where the index i spans the two possible outcomes, as a function of p shows that the entropy is a symmetric concave function that equals 0 if $p = 0$ or if $p = 1$, and it reaches a maximum value of 1 bit when $p = \frac{1}{2}$:



Entropy of a discrete random variable with non-uniform probabilities

The various letters of the written English language have the following relative frequencies (probabilities), in descending order:

E	T	O	A	N	I	R	S	H	D	L	C	..
.105	.072	.066	.063	.059	.055	.054	.052	.047	.035	.029	.023	..

If all 26 were equiprobable, the entropy of the ensemble would have been $H = -\sum_1^{26}(\frac{1}{26})\log_2(\frac{1}{26}) = 4.7$ bits. But their non-uniform probabilities reveal that, for example, an E is nearly five times more likely than a C. Surely such prior knowledge must reduce the uncertainty of this random variable, and so the letter-guessing game should be even more efficient.

In fact, the distribution of English letters has an entropy of only 4.0 bits. This means that, on average, only four 'Yes/No' questions are necessary, to discover the secretly chosen letter of the 26 letters in the alphabet.

How can this possibly be true?

That is the subject of Claude Shannon's Source Coding Theorem (so named because it uses the "statistics of the source," the *a priori* probabilities of the message generator, to construct an optimal code.) Note the important assumption: that the "source statistics" are known!

Shannon's seminal contributions, which essentially created the field of Information Theory, include two other key theorems that we will study: the Channel Coding Theorem (capacity for error-correcting codes); and the Noisy Channel Coding Theorem (channel capacity in Gaussian noise).

Several further measures of entropy first need to be defined, involving the marginal, joint, and conditional probabilities of random variables. Some key relationships will then emerge, that we can apply to the analysis of communication channels and codes.

More concepts and notation

We use capital letters X and Y to name random variables, and we use lower case letters x and y for instances of their respective outcomes. These are drawn from particular sets $\mathcal{A}$ and $\mathcal{B}$: $x \in \{a_1, a_2, ...a_J\}$, and $y \in \{b_1, b_2, ...b_K\}$. The probability of any particular outcome $p(x = a_i)$ is denoted p_i, for $0 \leq p_i \leq 1$ and with $\sum_i p_i = 1$.

An *ensemble* is just a random variable X. A *joint ensemble* 'XY' is an ensemble whose outcomes are ordered pairs x, y with $x \in \{a_1, a_2, ...a_J\}$ and $y \in \{b_1, b_2, ...b_K\}$. The joint ensemble XY defines a probability distribution $p(x, y)$ over all the JK possible joint outcomes x, y.

Marginal probability: From the Sum Rule, we can see that the probability of X taking on any particular value $x = a_i$ equals the sum of the joint probabilities of this outcome for X and all possible outcomes for Y: $p(x = a_i) = \sum_y p(x = a_i, y)$. We usually simplify this notation for the marginal probabilities to: $p(x) = \sum_y p(x, y)$ and $p(y) = \sum_x p(x, y)$.

Conditional probability: From the Product Rule, we can easily see that the conditional probability that $x = a_i$, given that $y = b_j$, is:

$$p(x = a_i | y = b_j) = \frac{p(x = a_i, y = b_j)}{p(y = b_j)}$$

We usually simplify this notation for conditional probability to:

$$p(x|y) = \frac{p(x, y)}{p(y)} \quad \text{and similarly} \quad p(y|x) = \frac{p(x, y)}{p(x)}.$$

It is now possible to define various entropy measures for joint ensembles.

The key thing to notice about all of them is that they are instances of the basic $H = -\sum_i p_i \log p_i$ concept except that the p_i will now be instead a joint probability distribution, or a conditional probability distribution.

For convenience we will often absorb the minus sign into the logarithm by taking the reciprocal inside of the log; and also for convenience sometimes we will replace terms with others by applying the Sum or Product Rules.

Joint entropy of XY

$$H(X, Y) = \sum_{x,y} p(x, y) \log \frac{1}{p(x, y)}$$

Note that we have replaced the usual minus sign in front by taking the reciprocal of $p(x, y)$ inside the logarithm.

From this definition, it follows that joint entropy is additive if X and Y are independent random variables:

$$H(X, Y) = H(X) + H(Y) \quad \text{iff} \quad p(x, y) = p(x)p(y)$$

Otherwise, the joint entropy of the joint ensemble XY is less than the sum of the entropies $H(X)$ and $H(Y)$ of the individual random variables. The amount of that difference (see the Venn diagram on the first slide) will be one of the most important quantities that we encounter.

Conditional entropy of an ensemble X, given that for Y, $y = b_j$

...measures the uncertainty remaining about random variable X after specifying that random variable Y has taken on some particular value $y = b_j$. It is defined naturally as just the entropy of that conditional probability distribution $p(x|y = b_j)$:

$$H(X|y = b_j) = \sum_x p(x|y = b_j) \log \frac{1}{p(x|y = b_j)}$$

If we now consider the above quantity *averaged* over all the possible outcomes that random variable Y might have, each weighted by its corresponding probability $p(y)$, then we arrive at the...

Conditional entropy of an ensemble X, given an ensemble Y:

$$H(X|Y) = \sum_y p(y) \left[\sum_x p(x|y) \log \frac{1}{p(x|y)} \right]$$

and we know from the Sum Rule that if we move the $p(y)$ term from the outer summation over y, to inside the inner summation over x, the two probability terms combine and become just $p(x, y)$ summed over all x, y.

Hence a simpler expression for this conditional entropy is:

$$H(X|Y) = \sum_{x,y} p(x, y) \log \frac{1}{p(x|y)}$$

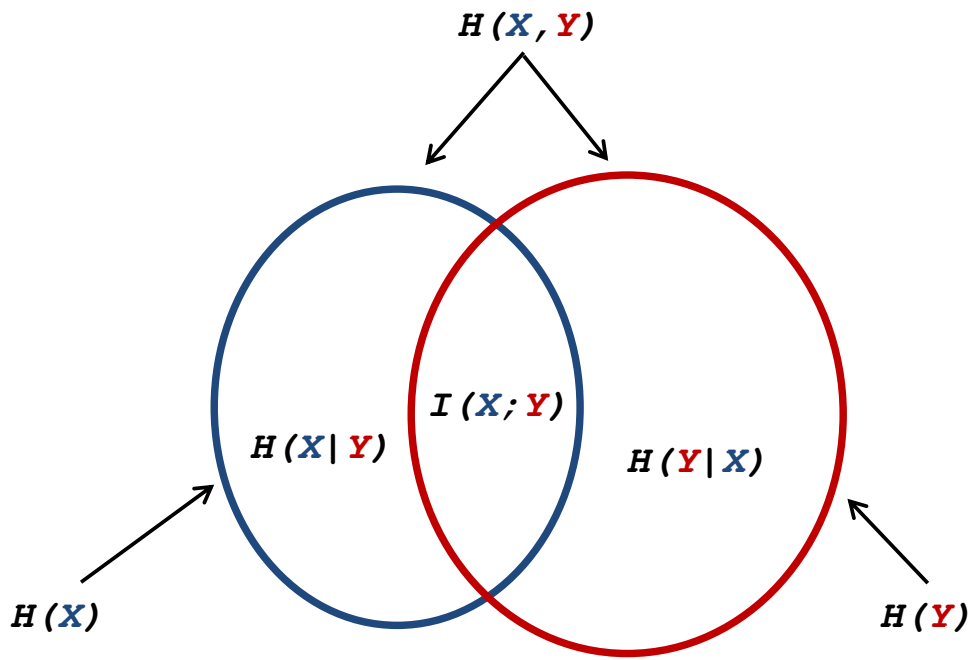
This measures the uncertainty that remains about X, when Y is known, averaged over all possible values of both random variables.

Chain Rule for Entropy

The joint entropy, conditional entropy, and marginal entropy for two random variables X and Y are related by:

$$H(X, Y) = H(X) + H(Y|X) = H(Y) + H(X|Y)$$

It should seem natural and intuitive that the joint entropy of a pair of random variables is the entropy of one plus the conditional entropy of the other (the uncertainty that it adds once its dependence on the first one has been discounted by conditionalizing on it).



"Independence Bound on Entropy"

A consequence of the Chain Rule for Entropy is that if we have many different random variables $X_1, X_2, ..., X_n$, then the sum of all their individual entropies must be an upper bound on their joint entropy:

$$H(X_1, X_2, ..., X_n) \leq \sum_{i=1}^{n} H(X_i)$$

Their joint entropy achieves this upper bound only if all of these n random variables are independent.

Another upper bound to note is that for any single random variable X which has N possible values, its entropy $H(X)$ is maximised when all of those values have the same probability $p_i = 1/N$. In that case,

$$H(X) = -\sum_{i} p_i \log_2 p_i = -\sum_{1}^{N} \frac{1}{N} \log_2 \frac{1}{N} = \log_2 N\,.$$

We shall use this fact when evaluating the efficiency of coding schemes.

Mutual Information between X and Y

The *mutual information* between two random variables measures the amount of information that one conveys about the other. Equivalently, it measures the average reduction in uncertainty about X that results from learning about Y. It is defined:

$$I(X;Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

Clearly, X says as much about Y, as Y says about X. Note that in case X and Y are independent random variables, then the numerator inside the logarithm equals the denominator. Then the log term vanishes to 0, and so the mutual information equals zero, as one should expect when random variables are independent.

Mutual information can be related to entropies in three alternative ways, as is apparent from the Venn diagram, and it has important properties. We will see soon that it corresponds to the relative entropy between the joint distribution $p(x,y)$ and the product distribution $p(x)p(y)$.

Non-negativity: mutual information is always ≥ 0. In the event that the random variables are perfectly correlated, then their mutual information is the entropy of either one alone. (Another way to say this is simply: $I(X;X) = H(X)$: the mutual information of a random variable with itself is its entropy. For this reason, the entropy $H(X)$ of a random variable X is sometimes referred to as its *self-information*.)

These properties are reflected in three equivalent definitions for the mutual information between random variables X and Y:

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ I(X;Y) &= H(Y) - H(Y|X) = I(Y;X) \\ I(X;Y) &= H(X) + H(Y) - H(X,Y) \end{aligned}$$

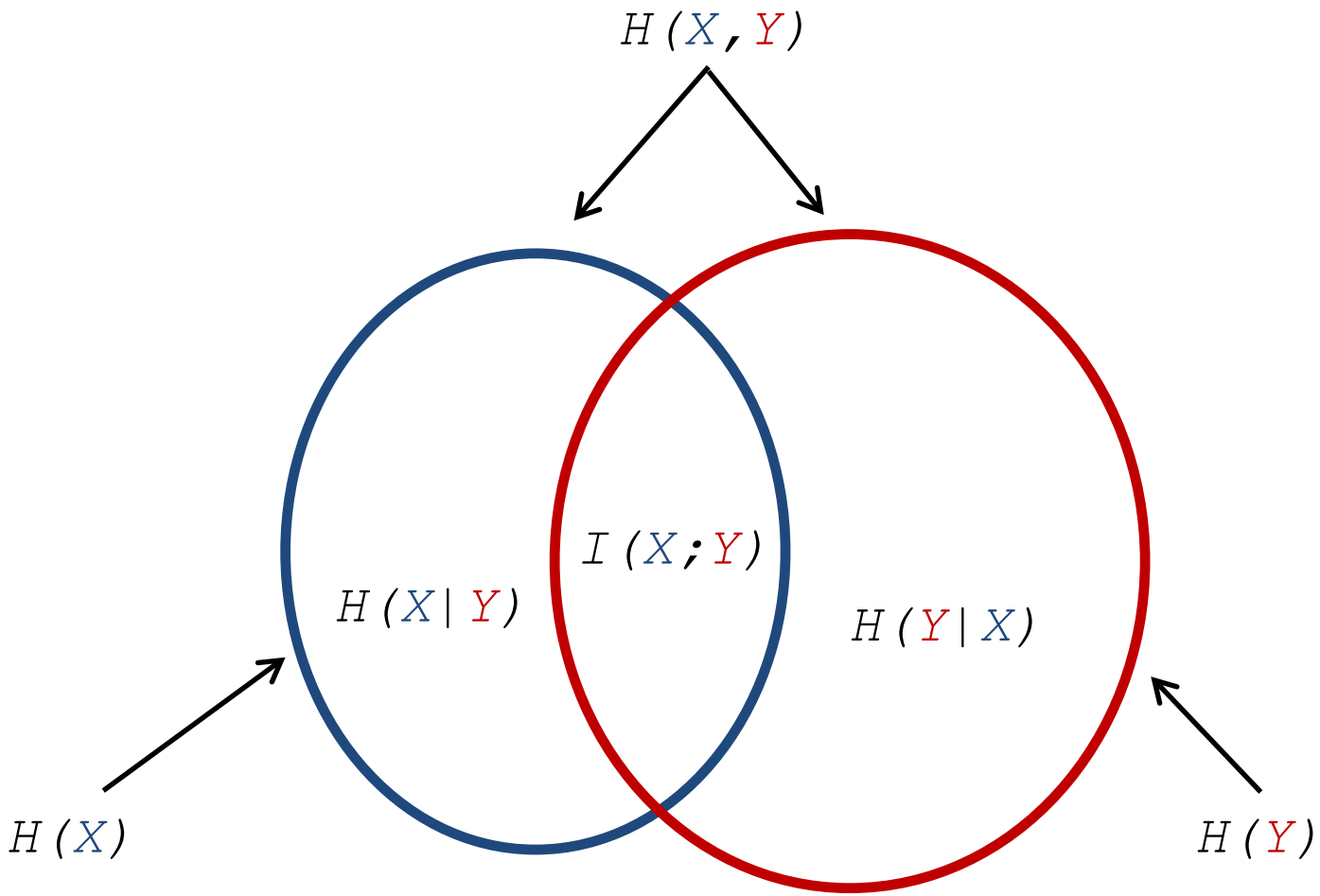
In a sense the mutual information $I(X;Y)$ is the intersection between $H(X)$ and $H(Y)$, since it represents their statistical dependence.

In the following Venn diagram, the portion of $H(X)$ that does not lie within $I(X;Y)$ is just $H(X|Y)$.

The portion of $H(Y)$ that does not lie within $I(X;Y)$ is just $H(Y|X)$.

Venn diagram summary of concepts and relationships

- Entropy $H(X),\ H(Y)$
- Joint entropy $H(X, Y) = H(X) \cup H(Y)$
- Conditional entropy $H(X|Y),\ H(Y|X)$
- Mutual information $I(X; Y) = H(X) \cap H(Y)$



Mutual information in pattern matching and inference

Mutual information helps measure the statistical similarity, or "distance", between two random variables. This is important in pattern recognition.

The Doctrine of Suspicious Coincidences

When the recurrence of patterns just by chance is a highly improbable explanation, it is unlikely to be a coincidence.

Cross-entropy, and "distances between distributions"

Consider two different probability distributions $p(x)$ and $q(x)$ over the same set of outcomes x for random variable X. We define cross-entropy between these distributions as:

$$H(p,q) = -\sum_x p(x) \log q(x) \ .$$

It resembles the familiar definition for the entropy $H(X)$ of a random variable X obeying distribution $q(x)$, except that now the expectation (probability weighting) is taken over a different distribution $p(x)$.

In coding theory it reveals the cost of making the false assumption that X is described by distribution $q(x)$, when in fact it is $p(x)$. If a coding scheme has been designed under the assumption of $q(x)$, then $H(p,q)$ predicts the costlier length of codewords expected on average, given the actual $p(x)$. In machine learning if a model is created from a training set, but some test set has different statistics, then cross-entropy assesses how accurately the learned model predicts this different test data.

Note that cross-entropy is asymmetric: $H(p,q) \neq H(q,p)$, and also that cross-entropy is minimised if $p(x) = q(x)$: then $H(p,q) = H(q) = H(p)$.

"Distance" $D(X, Y)$ between two random variables X and Y

The amount by which the joint entropy of two random variables exceeds their mutual information is a measure of the "*distance*" between them:

$$D(X, Y) = H(X, Y) - I(X; Y)$$

Note that this quantity satisfies the standard axioms for a distance:

- $D(X, Y) \geq 0$ (distances are non-negative),
- $D(X, X) = 0$ (distance between something and itself is 0),
- $D(X, Y) = D(Y, X)$ (symmetry), and
- $D(X, Z) \leq D(X, Y) + D(Y, Z)$ (triangle inequality for distances).

Relative entropy, or Kullback-Leibler distance

An important measure of the "distance" between two random variables that does *not* satisfy the above axioms for a distance metric, is the *relative entropy*, or *Kullback-Leibler (KL) distance* or divergence. It is also called the *information for discrimination*.

If $p(x)$ and $q(x)$ are two different probability distributions defined over the same set of outcomes x, then their relative entropy is:

$$D_{KL}(p\|q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

Note that $D_{KL}(p\|q) \geq 0$, and in case $p(x) = q(x)$ then their distance $D_{KL}(p\|q) = 0$, as one might hope. However, this metric is not strictly a "distance," since in general it lacks symmetry: $D_{KL}(p\|q) \neq D_{KL}(q\|p)$. Note also that major problems arise if there are any outcomes x for which $q(x)$ is vanishingly small relative to $p(x)$, thereby dominating the metric.

The relative entropy $D_{KL}(p\|q)$ is a measure of the "inefficiency" of assuming that a distribution is $q(x)$ when in fact it is $p(x)$. It is closely related to the cross-entropy $H(p, q)$ defined earlier; it is easy to see that

$$D_{KL}(p\|q) = H(p, q) - H(p)$$

If we have an optimal code for a distribution $p(x)$ (implying we need to use $H(p)$ bits, its entropy, in its codewords on average), then the number of additional bits that we would need to use if we instead described $p(x)$ using an optimal code for $q(x)$, would be their relative entropy $D_{KL}(p\|q)$.

Fano's Inequality

We know that conditioning reduces entropy: $H(X|Y) \leq H(X)$. It is clear that if X and Y are perfectly correlated, then their conditional entropies $H(X|Y) = 0$ and $H(Y|X) = 0$. It should also be clear that if X is any deterministic function of Y, then again, there remains no uncertainty about X once Y is known, so their conditional entropy $H(X|Y) = 0$.

Fano's Inequality relates the probability of error P_e in guessing X from knowledge of Y to their conditional entropy $H(X|Y)$, when the number of possible outcomes is $|\mathcal{A}|$ (*e.g.* the length of a symbol alphabet):

$$P_e \geq \frac{H(X|Y) - 1}{\log |\mathcal{A}|}$$

The lower bound on P_e is a linearly increasing function of $H(X|Y)$.

The "Data Processing Inequality"

If random variables X, Y, and Z form a Markov chain (which means that the conditional distribution of Z depends only on Y and is independent of X), normally denoted as $X \to Y \to Z$, then the mutual information must be monotonically decreasing over steps along the chain:

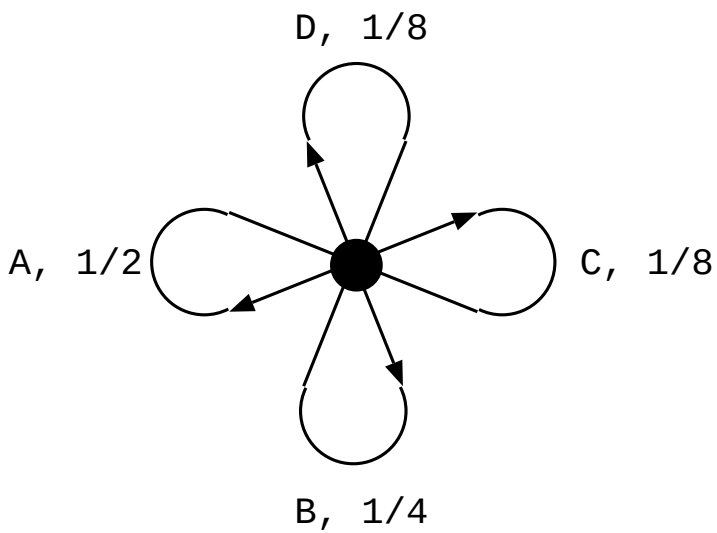
$$I(X;Y) \geq I(X;Z)$$

We turn next to applying these measures and relationships to the study of communication channels, symbol sources, codes, error correction, and channel capacity under various conditions.

3. Source coding theorem; variable-length and prefix codes

- *Discrete symbol sources as Markov processes, with one or many states.*
- *Entropy of sources. Code rates and compression. Fixed-length codes.*
- *Capacity of a noiseless channel. Huffman codes and the prefix property.*

We model a source of symbols as a Markov process, in which letters are emitted with known probabilities. Initially we consider just a one-state Markov process. (In a two-state Markov process, after the emission of certain symbols, the state may change to a different one having different emission probabilities for the symbols.)



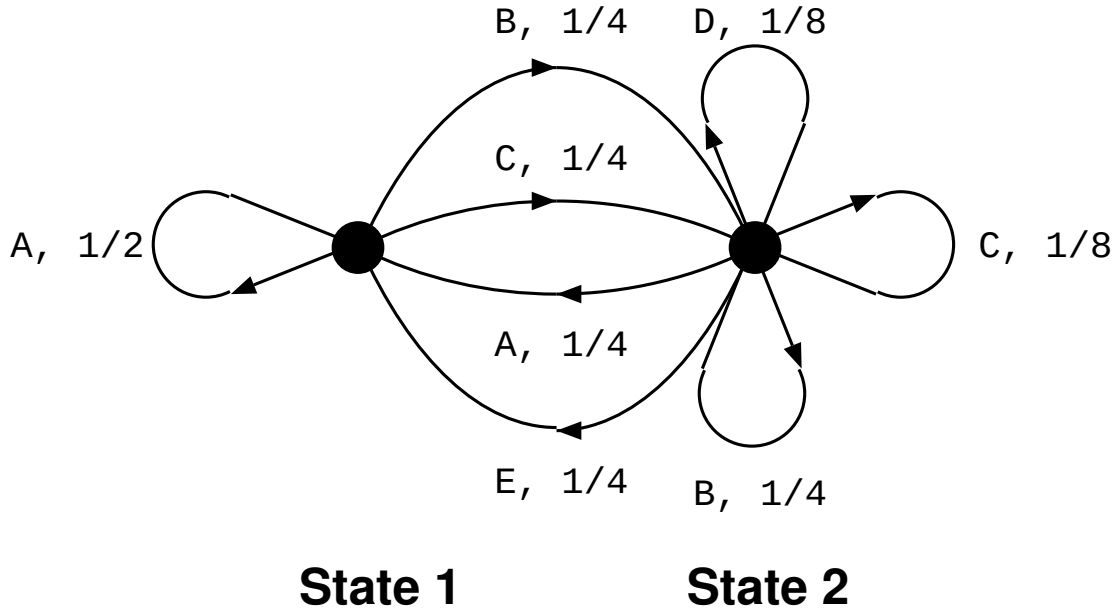
Such a Markov process (having any number of states) has an entropy. For the one-state Markov process above this is easily calculated from the probability distribution for letter emissions, and it is:

$$H = -\sum_i p_i \log p_i = (\tfrac{1}{2})(1) + (\tfrac{1}{4})(2) + (\tfrac{1}{8})(3) + (\tfrac{1}{8})(3) = 1.75 \text{ bits.}$$

Note that this entropy is based only on the symbol emission probabilities, regardless of the pace at which the source emits symbols, and so it is expressed as bits per symbol. If the symbols are emitted at a known rate, we may also characterise this symbol source in units of bits per second: (bits/symbol) $\times$ (symbols/second).

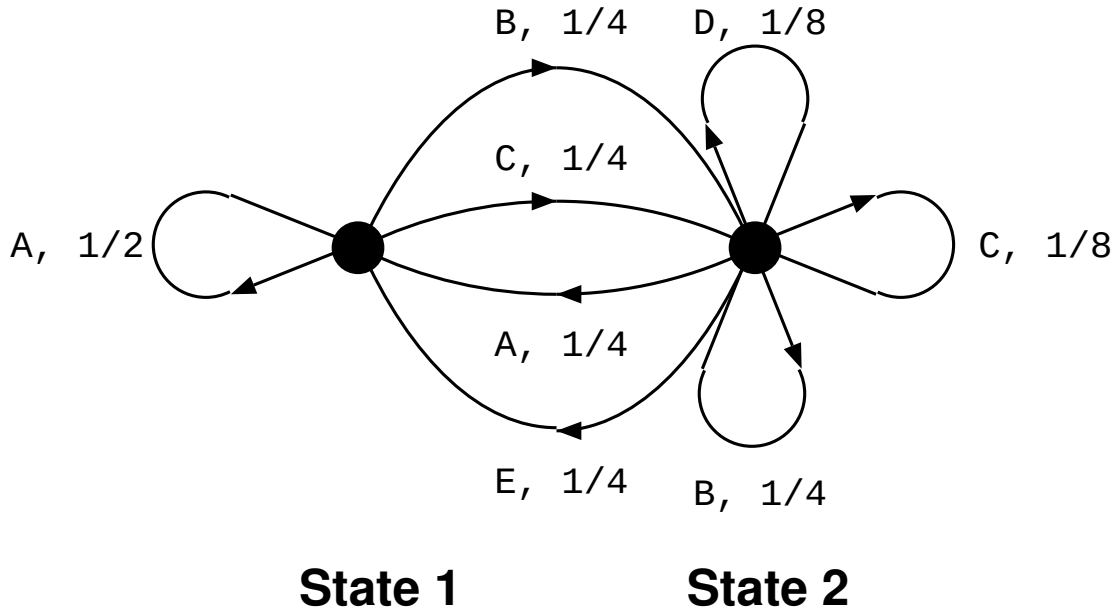
Now suppose that the Markov process has multiple states, and that transitions occur between states when certain symbols are emitted. For example, in English the letter 'q' is almost always followed by the letter 'u', but in other states (*e.g.* after 'z') the letter 'u' is far less probable.

First we can calculate the entropy associated with each state, as above; but then we also want to characterise the entropy of the entire process by taking into account the occupancy probabilities of these various states.



In the above two-state Markov process, which has "memory," there are two different sets of probabilities for the emission of symbols.

If a Markov process has several states $\{S_1, S_2, \ldots, S_n\}$, with associated emission probabilities $p_i(j)$ being the probability of emitting symbol j when in State S_i, then we first define the entropy of each of these states H_i in the normal manner, and then take the weighted average of those, using the occupancy probabilities P_i for the various states to arrive at an overall entropy $H = \sum_i P_i H_i$ for the multi-state Markov process.



$$H = \sum_i P_i H_i = -\sum_i P_i \sum_j p_i(j) \log p_i(j)$$

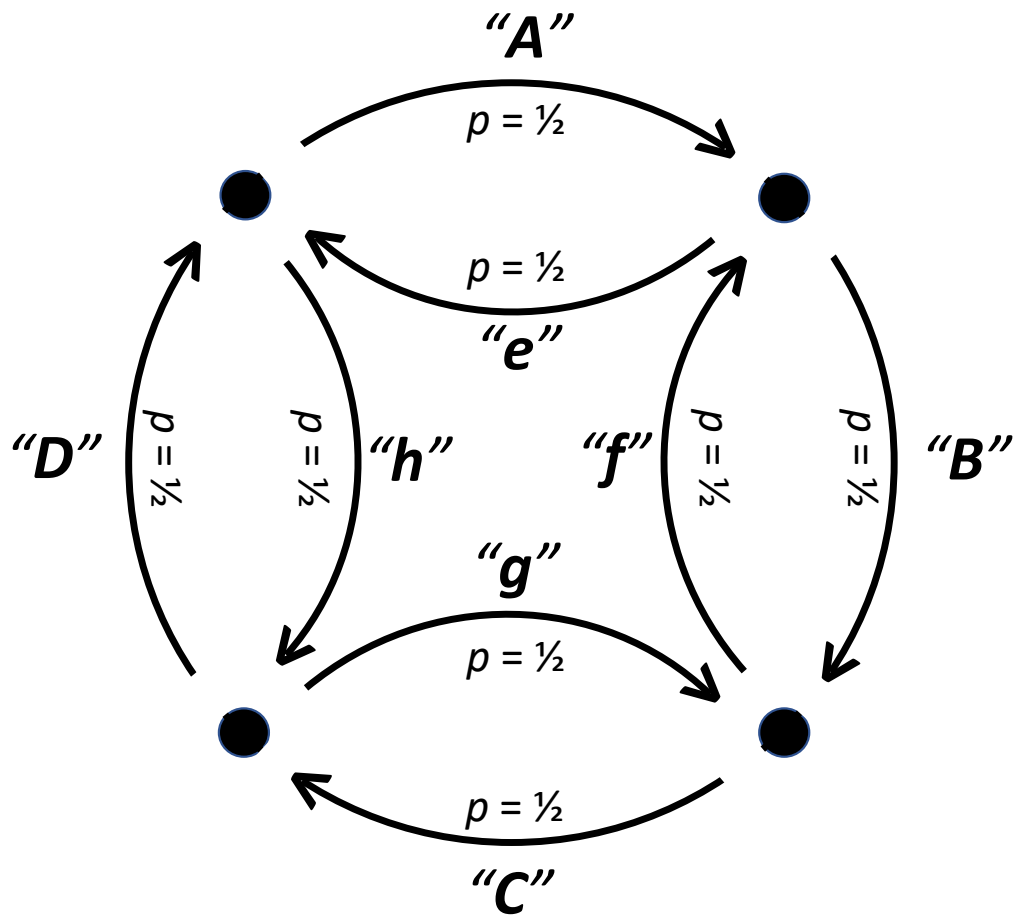
In State 1, this system emits letters (B or C) with probability 0.5 that generate a transition to State 2. Similarly, with probability 0.5, State 2 emits letters (A or E) generating a transition back to State 1. Thus the state occupancies are equiprobable, $P_1 = P_2 = 0.5$, and it is easy to see that the overall entropy of this two-state Markov process is:

$$H = (0.5)(1.5) + (0.5)(2.25) = 1\tfrac{7}{8} = 1.875 \text{ bits.}$$

Entropy of a Markov process versus stationary distributions

Suppose a one-state Markov process emits any of eight letters, all having equal probabilities. Clearly its entropy is $H = -\sum_{1}^{8} \frac{1}{8} \log_2 \left(\frac{1}{8}\right) = 3$ bits.

But now consider the following four-state Markov process:



It generates the same stationary (long term) probability distribution over the eight letters indicated:

letter	A	B	C	D	e	f	g	h
probability	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{8}$

But as the overall entropy of a multi-state Markov process is the average of the entropies of all states, weighted by their occupancy probabilities, (which are all the same in this case), and as the entropy of each state is obviously just 1 bit, we must conclude that the overall entropy of this four-state Markov process is therefore just 1 bit.

What reduced the entropy by a factor of three, compared to the one-state process, given that their output stationary distributions are the same?

Answer: The four-state Markov process has severe sequence constraints.

For example, an A can only be followed by a B or an e. An e can only be followed by an A or an h. These sequence constraints greatly reduce the randomness (or the uncertainty about the next letter).

Historical comment:

Such "leakage of entropy" played a significant role in World War II, helping the British and Polish code-breaking efforts at Bletchley Park to obtain military intelligence, code-named Ultra.

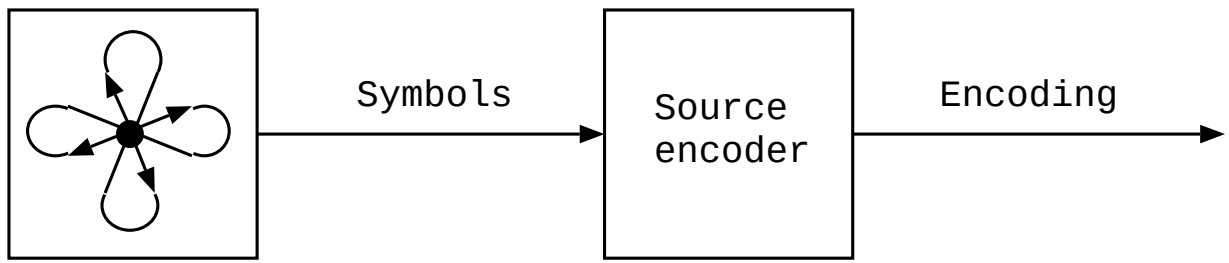
German *Enigma machines* had hard-wired cipher sequence constraints, reducing code entropy. They were substitution ciphers, with rotor wheels for specifying a new key every day; but a given letter in a plaintext source message could never be enciphered as itself. For code-breakers, this flaw greatly facilitated rapid elimination of candidate solutions.

Additionally, often source messages began predictably with "To" (*"An"*) or included the same salutation *"Heil Hitler!"*, undermining the strategy of rotating the permutation wheels daily to new key settings at midnight.

Historians agree that Bletchley code-breakers' work (e.g. Alan Turing's) on Ultra shortened the war by two years, saving about 14 million lives; – profoundly assisted by such entropy losses in German military ciphertext.

Fixed-length codes

We now consider various schemes for encoding symbols into codewords, as binary digits, with focus on the average codeword length per symbol when symbol probabilities are taken into account. We are interested in data compression, and the efficient use of channel capacity. We will also study the complexity of symbol decoding, and (later) the potential these schemes offer for error correction.



First consider encoding the set of N symbols $\{s_i\}$ having a probability distribution with entropy H, as a fixed-length (R) block of binary digits.

To ensure that the symbols can be decoded, we need a block of length $R = \log_2(N)$ if N is a power of 2, or otherwise $R = \lfloor \log_2(N) \rfloor + 1$ where $\lfloor X \rfloor$ is the largest integer less than X.

The code rate then is R bits per symbol, and as we noted earlier that entropy has an upper bound $H \leq \log_2(N)$, it follows that $H \leq R$.

The efficiency η of the coding is given by: $\eta = \dfrac{H}{R}$.

Note that fixed-length codes are inefficient for a number of reasons:

- If N is not a power of two, then much of the "address space" of the codewords is wasted. For example, if we have $N = 16$ symbols then 4 binary digits generate exactly the required number of codewords; but if $N = 17$ symbols then we need 5 bits, generating an address space of 32 codewords, of which 15 are wasted.
- If the N symbols occur with non-uniform probabilities, then even if N is a power of 2 the fixed-length code is still inefficient because we have $H < R$.
- In general, both of these sources of inefficiency for fixed-length codes will exist.

Variable-length codes, and those with the prefix property

In general symbols are not equiprobable, and we would hope to achieve some more compressed form of encoding by using variable-length codes, just as telegraphers used Morse code with short encodings for the more common letters and longer encodings for the less frequently used letters.

Consider a four-symbol alphabet and three possible variable-length codes:

x	$p(x)$	Code 1	Code 2	Code 3
A	1/2	1	0	0
B	1/4	00	10	01
C	1/8	01	110	011
D	1/8	10	111	111

The entropy of this alphabet is $H = (\frac{1}{2})(1) + (\frac{1}{4})(2) + (\frac{1}{8})(3) + (\frac{1}{8})(3)$ $= 1.75$ bits. Note too that the average codeword length in bits/symbol (weighting each codeword length by its symbol's probability) for Code 2 and for Code 3 is also $R = 1.75$ bits; indeed the arithmetic is identical to that for entropy H. But for Code 1, the average codeword length is instead $R = (\frac{1}{2})(1) + (\frac{1}{4})(2) + (\frac{1}{8})(2) + (\frac{1}{8})(2) = 1.5$ bits/symbol.

Now let us examine some properties of each code in more detail.

x	$p(x)$	Code 1	Code 2	Code 3
A	1/2	1	0	0
B	1/4	00	10	01
C	1/8	01	110	011
D	1/8	10	111	111

Code 1 is not uniquely decodable: a bit sequence such as 1001 could be decoded either as ABA, or as DC (unless punctuation is added).

Code 2 *is* uniquely decodable, and instantaneously: once we have the bits for an encoded symbol we can decode immediately, without backtracking or waiting for more. This is called the prefix property: no codeword is the prefix of a longer codeword. For example, the bit string 0110111100110 is instantaneously decodable as ACDBAC.

Code 3 lacks the prefix property and it is not an instantaneous code: the codeword for B is also the start of the codeword for C; and the codeword for A cannot be decoded until more bits are received to exclude B or C.

Clearly, self-punctuating Code 2 is the most desirable of the three codes.

Shannon's Source-Coding Theorem

A remarkably far-reaching result, which was illustrated already in Code 2, is that: *it is possible to compress a stream of data whose entropy is H into a code whose rate R approaches H in the limit, but it is impossible to achieve a code rate $R < H$ without loss of information.*

Thus the Source-Coding Theorem establishes limits to data compression, and it provides operational meaning to entropy.

The usual statement of the theorem is that for a discrete source with entropy H, for any $\epsilon > 0$, it is possible to encode the symbols into a uniquely decodable code at an average rate R such that

$$R = H + \epsilon$$

as an asymptotic limit ($\epsilon \to 0$ as the number of symbols gets large).

Proof is given in Shannon and Weaver (1949). This is sometimes called the Noiseless Coding Theorem as it does not consider noise processes, such as bit corruption in a communication channel. Note the assumption that the source statistics are known, so H can be calculated.

Huffman codes

An optimal prefix code (having the shortest possible average codeword length in bits/symbol) for any given probability distribution of symbols can be constructed using an algorithm discovered by Huffman. This is a constructive illustration of Shannon's source-coding theorem.

The idea is to assign the bits in a reverse sequence corresponding to increasing symbol probability, so that the more probable symbols are encoded with shorter codewords. Thus we start with the two least frequent symbols and assign the "least significant bit" to them. More probable symbols will lack "less significant bits" in their codewords, which are therefore shorter, given the prefix property.

A binary tree is constructed in reverse, which specifies a hierarchical partitioning of the alphabet using a priority queue based (inversely) on probability.

Constructing a Huffman tree (example on next slide)

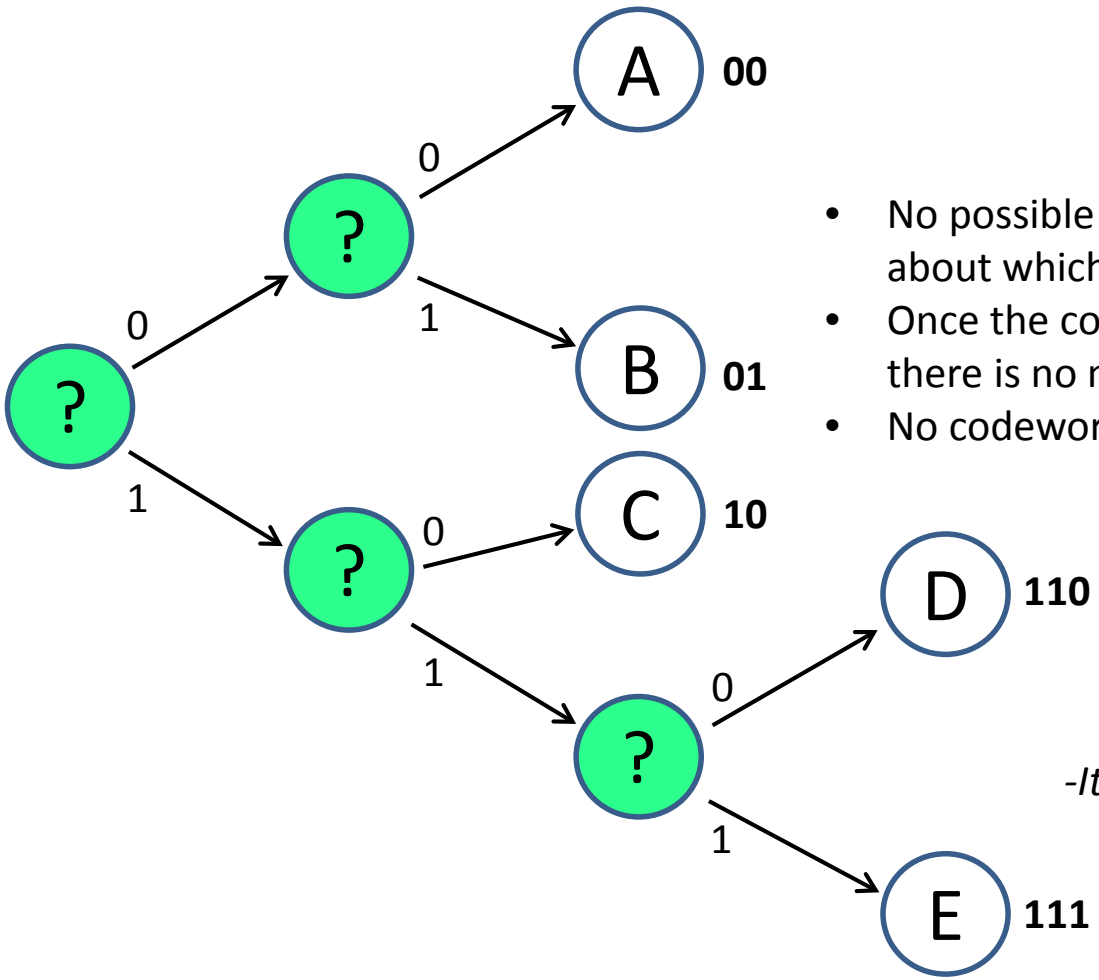
1. Find the two symbols having the lowest probabilities and assign a bit to distinguish them. This defines a branch in the binary tree.
2. Combine those two into a virtual "symbol" node whose probability is the sum of their two probabilities.
3. From this new shorter list of symbol nodes, repeat Step 1.
4. Repeat Step 2.
5. Continue this process until there is just one symbol node. That is the root node of the Huffman tree.

In the event that the symbol probabilities are powers of $1/2$, then a Huffman code achieves perfect efficiency ($R = H$). This is because each extra bit in a codeword removes half of the remaining uncertainty about the symbol, given that the bits so far have not specified it.

Note that there is not a unique Huffman code for any symbol alphabet. (The codewords having any given length could always be interchanged.) But a Huffman code is as efficient (compressive) as possible.

Example of a uniquely decodable, instantaneous, prefix code over 5 letters {A,B,C,D,E}

p(A)=1/4
p(B)=1/4
p(C)=1/4
p(D)=1/8
p(E)=1/8



- No possible received string of bits is ambiguous about which symbols were encoded.
- Once the codeword for any symbol is received, there is no need to wait for more bits to resolve it.
- No codeword is a prefix of another codeword.

How efficient is this code?
-It achieves optimal Shannon efficiency:

Note the entropy of this alphabet is:
$3 \cdot (1/4) \cdot 2 + 2 \cdot (1/8) \cdot 3 =$ **2.25 bits**

Note the average codeword length is also:
$3 \cdot (1/4) \cdot 2 + 2 \cdot (1/8) \cdot 3 =$ **2.25 bits/codeword**

Kraft-McMillan inequality

Any instantaneous code (one with the prefix property) must satisfy the following condition on the codeword lengths: if the N codewords have lengths $c_1 \leq c_2 \leq \cdots \leq c_N$, then

$$\sum_{i=1}^{N} \frac{1}{2^{c_i}} \leq 1\,.$$

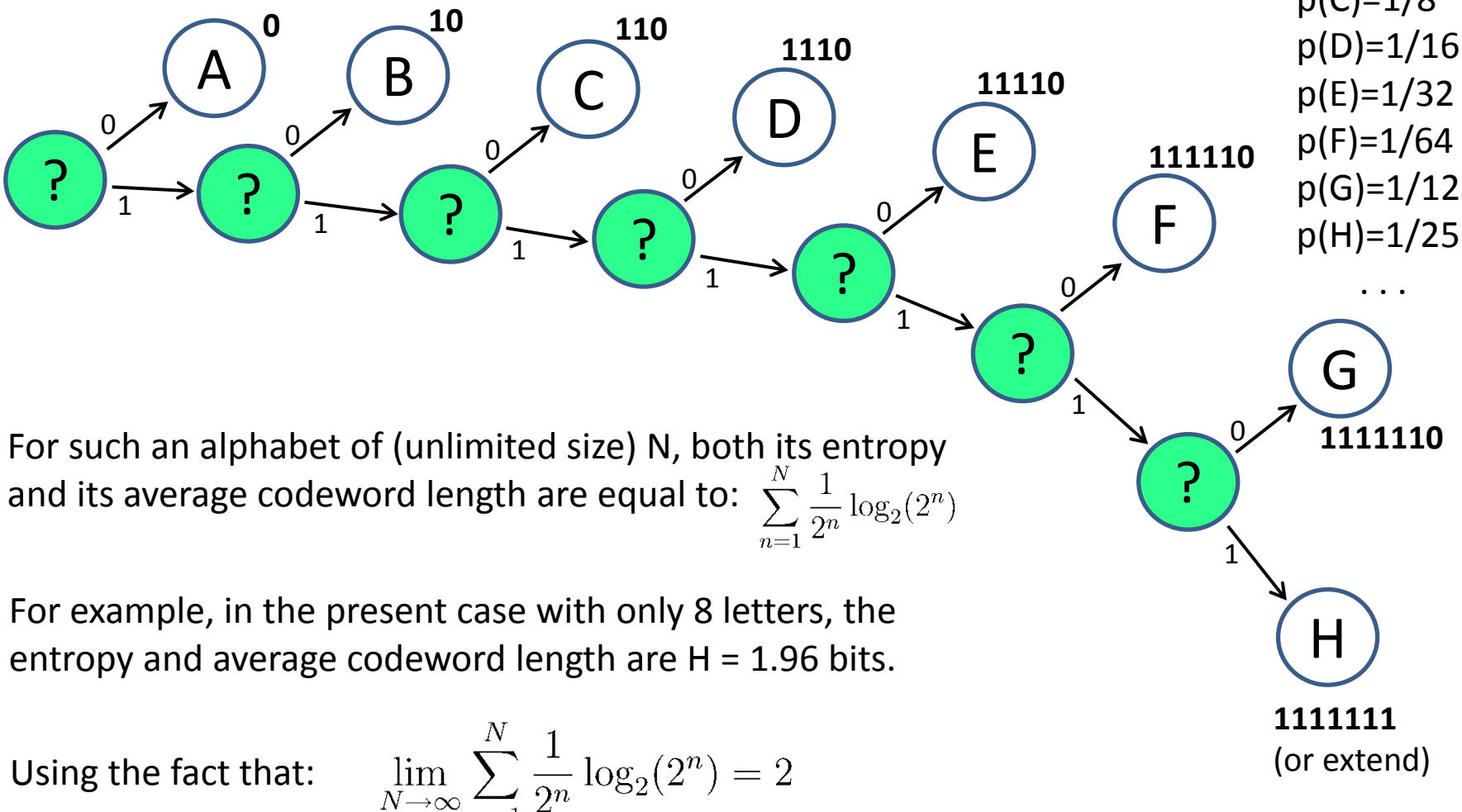
Although this is a necessary condition, it is not a sufficient condition for a code to be an instantaneous code. For example, in our earlier study of three codes, we saw that Code 3 was not an instantaneous code but its codeword lengths were the same as those of Code 2, and both of them do satisfy the Kraft-McMillan inequality. (Note that the summation above equals 1.0 for both Codes 2 and 3, but it equals 1.25 for Code 1.)

Finally, we note an amazing consequence of the material covered in this section: It is possible to encode an alphabet of INFINITE length, provided its symbols have a particular probability distribution, with a prefix code whose average codeword length is no more than 2 bits per codeword!

Efficiency of prefix codes

Example of an alphabet of unlimited size, with a special probability distribution, that can be uniquely encoded with average codeword length **< <u>2 bits/codeword</u>** !

p(A)=1/2
p(B)=1/4
p(C)=1/8
p(D)=1/16
p(E)=1/32
p(F)=1/64
p(G)=1/128
p(H)=1/256
. . .



For such an alphabet of (unlimited size) N, both its entropy and its average codeword length are equal to: $\sum_{n=1}^{N} \frac{1}{2^n} \log_2(2^n)$

For example, in the present case with only 8 letters, the entropy and average codeword length are H = 1.96 bits.

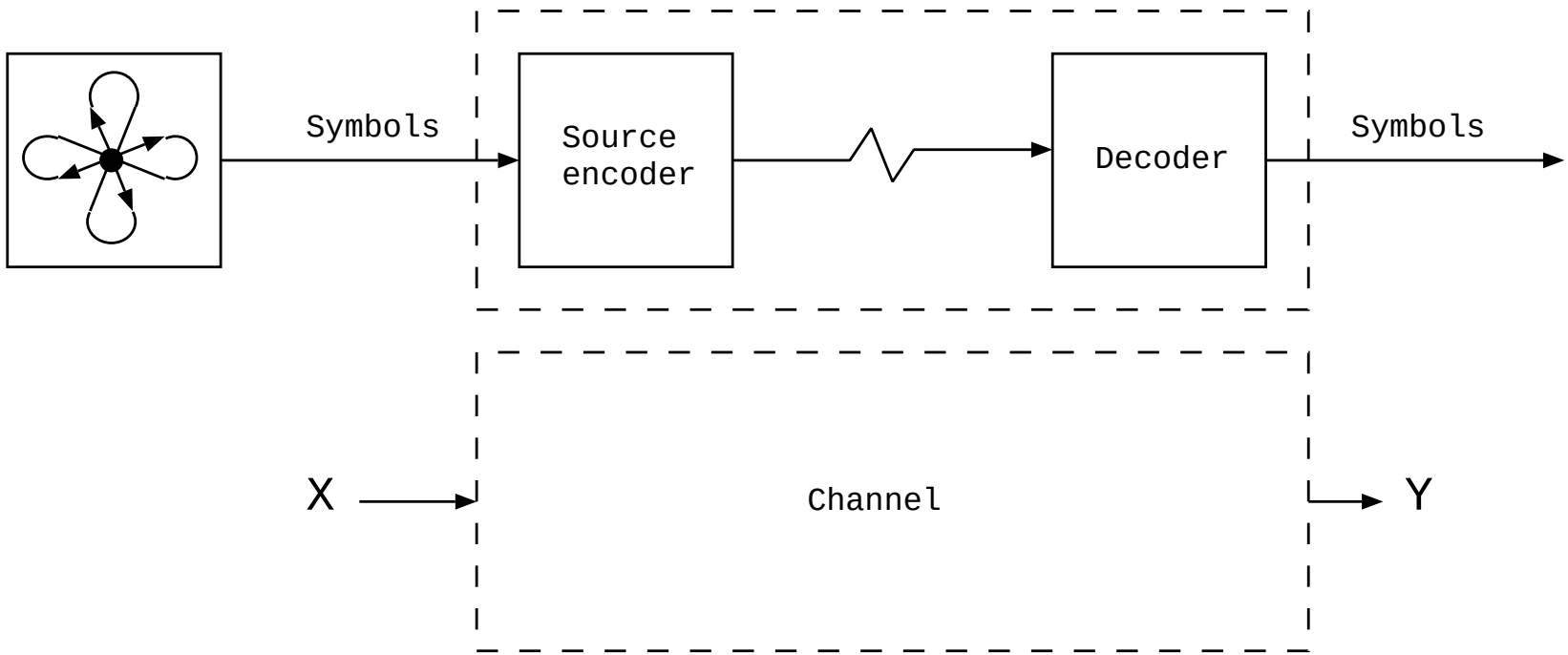
Using the fact that: $$\lim_{N\to\infty} \sum_{n=1}^{N} \frac{1}{2^n} \log_2(2^n) = 2$$

we see that even if the size of this alphabet grows indefinitely, it can still be uniquely encoded with an average codeword length just below 2 bits/codeword.

4. Discrete channel capacity, noise, and error correction

- *Channel matrix. Mutual information between input and output.*
- *Binary symmetric channel with error probability. Channel coding.*
- *Capacity of a noisy discrete channel. Error-correcting codes.*

We have considered discrete symbol sources, and encodings for them, and their code rates and compression limits. Now we consider channels through which such encodings pass, relating the input random variable X to the (perhaps randomly corrupted) output symbol, random variable Y.



Channel matrix

We shall apply all the tools and metrics introduced in the first part of this course. An input alphabet is random variable $X = \{x_1, \ldots, x_J\}$, and the output symbol is drawn from random variable $Y = \{y_1, \ldots, y_K\}$.

Note that J and K need not be the same. For example, for binary input $X = \{0, 1\}$ we could have an output alphabet $Y = \{0, 1, \perp\}$ where $\perp$ means the decoder has detected some error.

A discrete memoryless channel can then be represented as a set of transition probabilities $p(y_k|x_j)$: that if symbol x_j is injected into the channel, symbol y_k is emitted. These conditional probabilities form the channel matrix:

$$\begin{pmatrix} p(y_1|x_1) & p(y_2|x_1) & \ldots & p(y_K|x_1) \\ p(y_1|x_2) & p(y_2|x_2) & \ldots & p(y_K|x_2) \\ \vdots & \vdots & \ddots & \vdots \\ p(y_1|x_J) & p(y_2|x_J) & \ldots & p(y_K|x_J) \end{pmatrix}$$

For every input symbol we will get something out, so $\sum_{k=1}^{K} p(y_k|x_j) = 1$.

Average probability of symbol error, or correct reception

Using the channel matrix as well as some known probability distribution $\{p(x_j), j = 1, 2, \ldots, J\}$ for a memoryless symbol source X, we can now apply the product and sum rules of probability to compute certain useful quantities. The joint probability distribution for the random variables X (input) and Y (output symbols) is: $p(x_j, y_k) = p(y_k|x_j)p(x_j)$, and the marginal probability distribution for output symbol y_k appearing is:

$$p(y_k) = \sum_{j=1}^{J} p(x_j, y_k) = \sum_{j=1}^{J} p(y_k|x_j)p(x_j)$$

Finally we can define the average probability of symbol error P_e as the sum of all elements in the channel matrix in which a different symbol was emitted than the one injected, which we will signify as $k \neq j$, weighted by the probability distribution $p(x_j)$ for the input symbols:

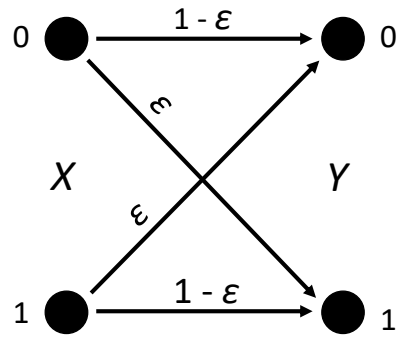
$$P_e = \sum_{j=1}^{J} \sum_{(k=1, k\neq j)}^{K} p(y_k|x_j)p(x_j)$$

and so the average probability of correction reception is: $1 - P_e$.

Binary symmetric channel

A binary symmetric channel has two input and two output symbols (denoted $\{0, 1\}$ for simplicity), and a common probability ϵ of incorrect decoding of the input at the output. Thus its channel matrix is:

$$\begin{pmatrix} 1-\epsilon & \epsilon \\ \epsilon & 1-\epsilon \end{pmatrix}$$ which we can graph as:



Using the concepts and tools introduced at the beginning, we now wish to characterise the channel in terms of the mutual information between the input and output, and the conditional entropy $H(X|Y)$: how much uncertainty remains about the input X, given receipt of the output Y.

Let us assume that the two symbols of the input source $\{0, 1\}$ have probabilities $\{1/2, 1/2\}$ and so the source entropy is $H(X) = 1$ bit.

Note that the two output symbols also retain probabilities $\{1/2, 1/2\}$, independent of the error probability ϵ, and so we also have $H(Y) = 1$ bit.

The channel's conditional entropy $H(X|Y) = -\sum_{x,y} p(x,y) \log p(x|y)$

$$= -\frac{1}{2}(1-\epsilon)\log(1-\epsilon) - \frac{1}{2}\epsilon\log(\epsilon) - \frac{1}{2}\epsilon\log(\epsilon) - \frac{1}{2}(1-\epsilon)\log(1-\epsilon)$$
$$= -\epsilon\log(\epsilon) - (1-\epsilon)\log(1-\epsilon)\,.$$

Finally, the mutual information $I(X;Y)$ between the input and output is:

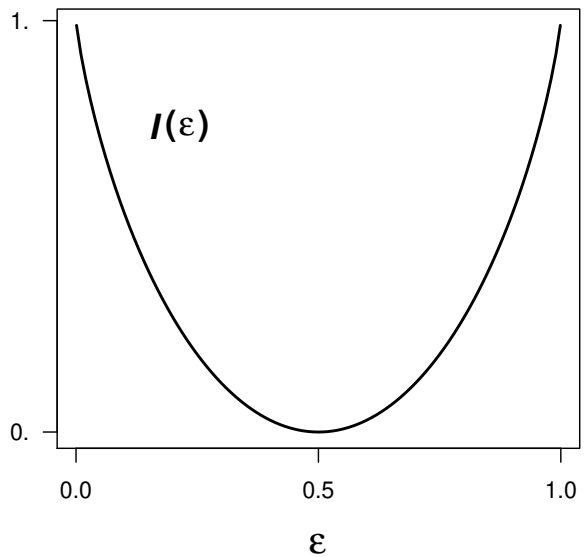
$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= 1 + \epsilon\log(\epsilon) + (1-\epsilon)\log(1-\epsilon)\,. \end{aligned}$$

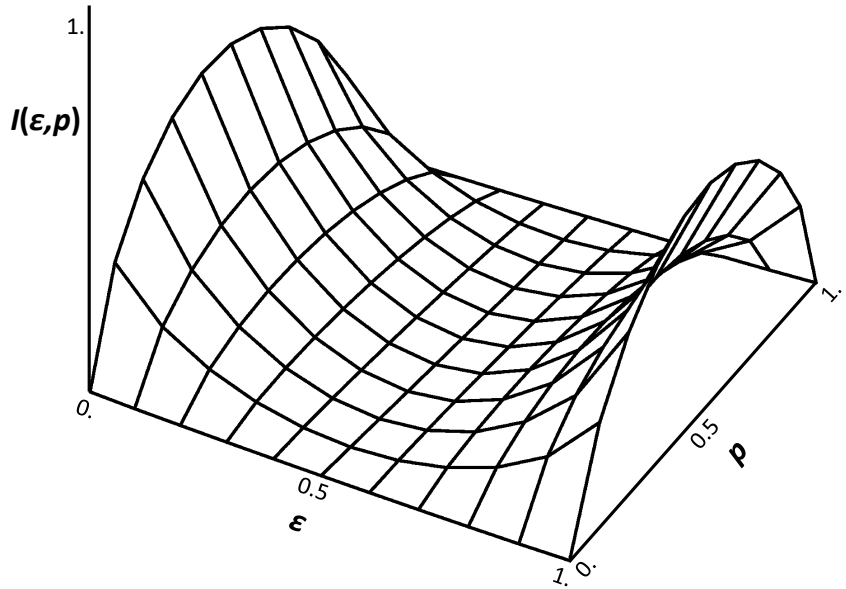
The channel capacity, denoted C, is defined as the maximum of its mutual information $I(X;Y)$ over all possible input source distributions.

$$C = \max_{\{p(x_j)\}} I(X;Y)$$

In this calculation we are assuming a "binary symmetric channel," meaning that flips are equally likely for both input bits. Note also that having equiprobable $\{1/2, 1/2\}$ binary source symbols maximises $I(X;Y)$, which would be smaller had the input symbols not been equiprobable.

Capacity of the binary symmetric channel





The plot on the left shows $I(X;Y) = 1 + \epsilon \log(\epsilon) + (1-\epsilon)\log(1-\epsilon)$ as a function of the channel error (or transition) probability ϵ, for the binary symmetric channel. Clearly $I(X;Y)$ is optimal in a channel with $\epsilon = 0$ or $\epsilon = 1$, meaning that a bit is never flipped or it is always flipped, leaving no uncertainty, and then the channel capacity is 1 bit per transmitted bit.

The surface on the right extends that same plot in a second dimension representing the variation of the two input probabilities. Over all the possible distributions $\{p(x_j)\}$ for the binary input, we see that channel capacity is always maximised for the equiprobable case $\{1/2, 1/2\}$, at a value that depends on the error probability ϵ (again optimal if 0 or 1).

Schemes for acquiring immunity to channel noise

Adding various kinds of redundancy to messages can provide immunity to channel noise, including some remarkable cases of error-correcting codes. We begin with simple repetition codes, and with the observation that much redundancy already exists in natural language. The two sentences

1. "Bring reinforcements, we're going to advance."
2. "It's easy to recognise speech."

remain intelligible after the random corruption of 1 in 10 characters:

1. "Brizg reinforce ents, we're goint to advance."
2. "It's easy mo recognis speech."

But this intrinsic redundancy does not overcome the lack of orthogonality between messages. In audio terms, insufficient "distance" exists between the original two sentences above and the following spoken sentences:

1. "Bring three and fourpence, we're going to a dance."
2. "It's easy to wreck a nice peach."

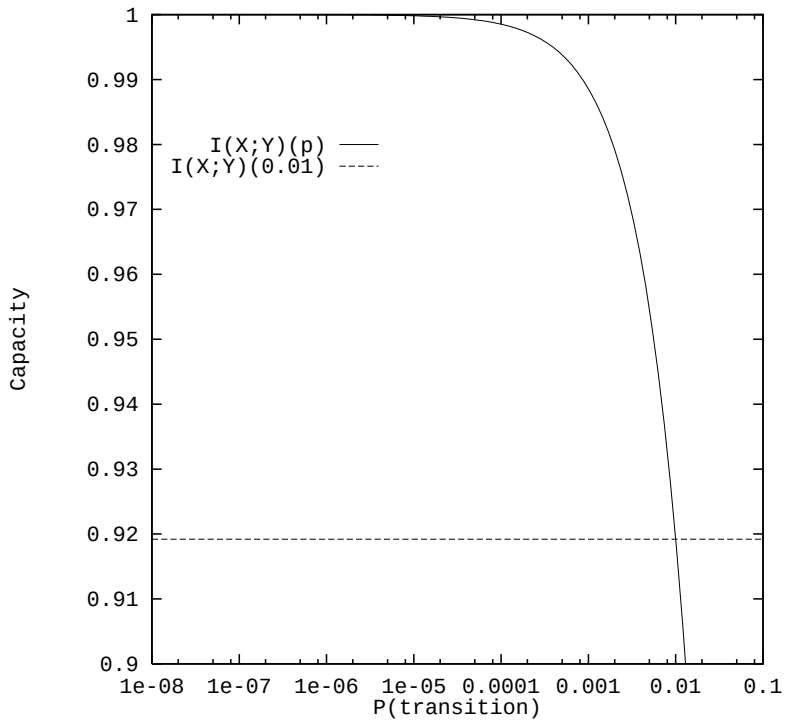
Repetition codes

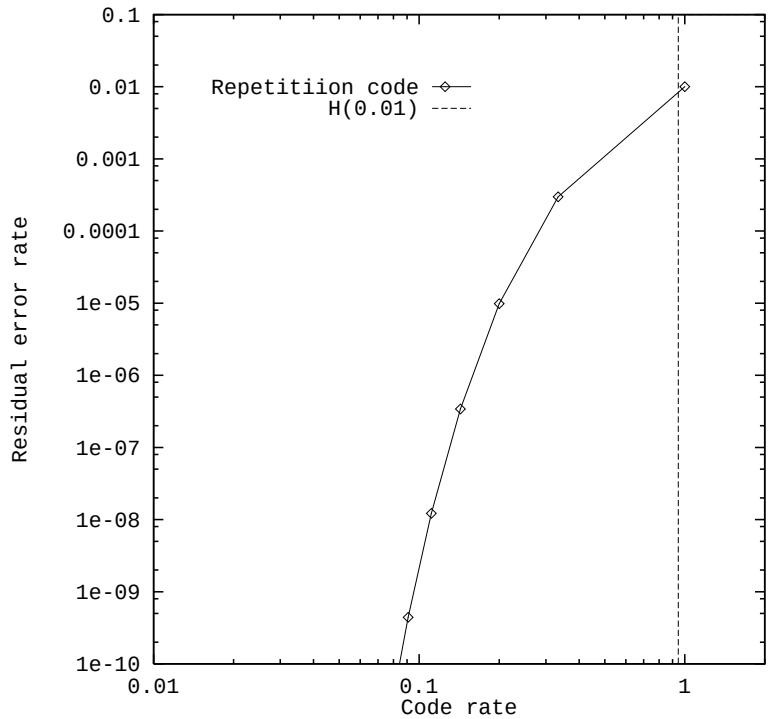
A simple approach to overcoming channel noise might be just to repeat each message several times. Obviously the effective transmission rate is then diluted by a factor of N if there are N transmissions. We can analyze how much good this does in the case of the binary symmetric channel, with transition (error) probability ϵ, which we saw for equiprobable inputs has a channel capacity $C = 1 + \epsilon \log(\epsilon) + (1 - \epsilon) \log(1 - \epsilon)$.

If we transmit every symbol an odd number $N = 2m + 1$ times and then perform majority voting, an error still persists if $m + 1$ or more bits are received in error. The probability P_e of this happening can be calculated with a binomial series. The summation is done over all modes of failed majority voting; the first factor is a combinatorial term (how many ways to choose i failed bits out of the $2m + 1$ bits), and the other factor is the probability that i of the bits were flipped while the remaining $2m + 1 - i$ bits were correctly transmitted:

$$P_e = \sum_{i=m+1}^{2m+1} \binom{2m+1}{i} \epsilon^i (1 - \epsilon)^{2m+1-i}$$

Let us suppose the transition probability is $\epsilon = 0.01$, in which case the channel capacity is $C = 1 + \epsilon \log(\epsilon) + (1 - \epsilon) \log(1 - \epsilon) = 0.9192$ bit per bit. Capacity is plotted as a function of ϵ in the left panel below.





The right panel plots the residual error probability P_e as a function of the code rate (channel capacity diluted by the number of transmissions).

It shows, for example, that if every transmission is repeated 7 times, giving a code rate of 0.13, then for this channel with transition probability $\epsilon = 0.01$ there remains an error probability of about 1 in a million.

Channel Coding Theorem: error-correcting codes

We arrive at Shannon's second theorem, the channel coding theorem:

> *For a channel of capacity C and a symbol source of entropy H, provided that $H \leq C$, there exists a coding scheme such that the source is reliably transmitted through the channel with a residual error rate lower than any arbitrarily small ϵ.*

Shannon's proof of this theorem is an existence proof rather than a means to construct such codes in the general case. In particular, the choice of a good code is dictated by the characteristics of the channel noise.

It is remarkable that it is possible to transmit data reliably through noisy channels, without using repetition. Methods of error-correcting encoding are pervasive not only in communications, but also within storage media. For example, a bubble or a scratch in CD or DVD media may obliterate many thousands of bits, but with no loss of data.

Today there is a vast literature around this subject. We will examine just one simple but efficient error-correcting code, with $H = C$.

A systematic (7/4) Hamming Code

Suppose symbols are encoded into a channel in blocks of seven bits. For example, it might be a block code for up to 128 ASCII characters. But the channel may randomly corrupt 1 bit in each block of 7 (or none).

Thus for each block of 7 bits $b_1 b_2 b_3 b_4 b_5 b_6 b_7$ encoding a particular input symbol x_j among the 128 in the alphabet, any one of 8 possible output symbols may emerge, all equiprobable (with probability 1/8), having the following bit patterns:

$$
\begin{array}{c}
b_1 b_2 b_3 b_4 b_5 b_6 b_7 \\
\bar{b}_1 b_2 b_3 b_4 b_5 b_6 b_7 \\
b_1 \bar{b}_2 b_3 b_4 b_5 b_6 b_7 \\
b_1 b_2 \bar{b}_3 b_4 b_5 b_6 b_7 \\
b_1 b_2 b_3 \bar{b}_4 b_5 b_6 b_7 \\
b_1 b_2 b_3 b_4 \bar{b}_5 b_6 b_7 \\
b_1 b_2 b_3 b_4 b_5 \bar{b}_6 b_7 \\
b_1 b_2 b_3 b_4 b_5 b_6 \bar{b}_7
\end{array}
$$

where an overbar $\bar{b}_i$ signifies that bit b_i has been flipped.

Let us calculate the information capacity of this channel per symbol, $C_S = \max_{\{p(x_j)\}} I(X;Y) = \max_{\{p(x_j)\}} (H(Y) - H(Y|X))$, allocated per bit C_b by dividing by 7. For convenience we observe that $H(Y) = 7$ because there are $N = 128 = 2^7$ equiprobable possible symbols, and also we observe that $p(y_k|x_j)$ is always $1/8$ because any one of 8 equiprobable possible output symbols y_k will emerge for any given input symbol x_j.

$$C_S = \max_{\{p(x_j)\}} (H(Y) - H(Y|X)) \text{ bits per symbol}$$

$$C_b = \frac{1}{7}\left(7 - \sum_j \sum_k p(y_k|x_j) \log\left(\frac{1}{p(y_k|x_j)}\right) p(x_j)\right) \text{ bits per bit}$$

$$= \frac{1}{7}\left(7 + \sum_j 8(\frac{1}{8}\log\frac{1}{8})\frac{1}{N}\right)$$

$$= \frac{1}{7}\left(7 + N(\frac{8}{8}\log\frac{1}{8})\frac{1}{N}\right)$$

$$= \frac{4}{7} \text{ bits per bit}$$

Thus the information capacity of this channel is $\frac{4}{7}$ bit per bit encoded.

Syndromes

Can we develop an error-correcting code that reliably transmits data through this channel, if our source entropy $H \leq C = \frac{4}{7}$ bit per bit?

We construct new 7-bit codewords, each of which contains just 4 bits of symbol-encoding data, plus another 3 bits computed for error correction.

In our new codewords, bits b_3, b_5, b_6, and b_7 are symbol-encoding data. But new bits b_4, b_2, and b_1 are computed from those 4 bits, as follows:

$$b_4 = b_5 \oplus b_6 \oplus b_7$$
$$b_2 = b_3 \oplus b_6 \oplus b_7$$
$$b_1 = b_3 \oplus b_5 \oplus b_7$$

Upon reception of these new 7-bit codewords (or corruptions of them), 3 further bits called syndromes s_4, s_2, and s_1 are then computed:

$$s_4 = b_4 \oplus b_5 \oplus b_6 \oplus b_7$$
$$s_2 = b_2 \oplus b_3 \oplus b_6 \oplus b_7$$
$$s_1 = b_1 \oplus b_3 \oplus b_5 \oplus b_7$$

If the syndromes computed upon reception are all 0, there was no error. Otherwise, the bit position $b_{s_4 s_2 s_1}$ is the bit in error.

Thus we can reliably transmit data through this noisy channel, whose capacity is $C = \frac{4}{7}$ bit per bit encoded, by embedding 3 error-correcting bits with every 4 useful data bits. This dilutes our source entropy to $H = \frac{4}{7}$ bit per bit of data, consistent with the requirement of Shannon's Channel Coding Theorem that $H \leq C$.

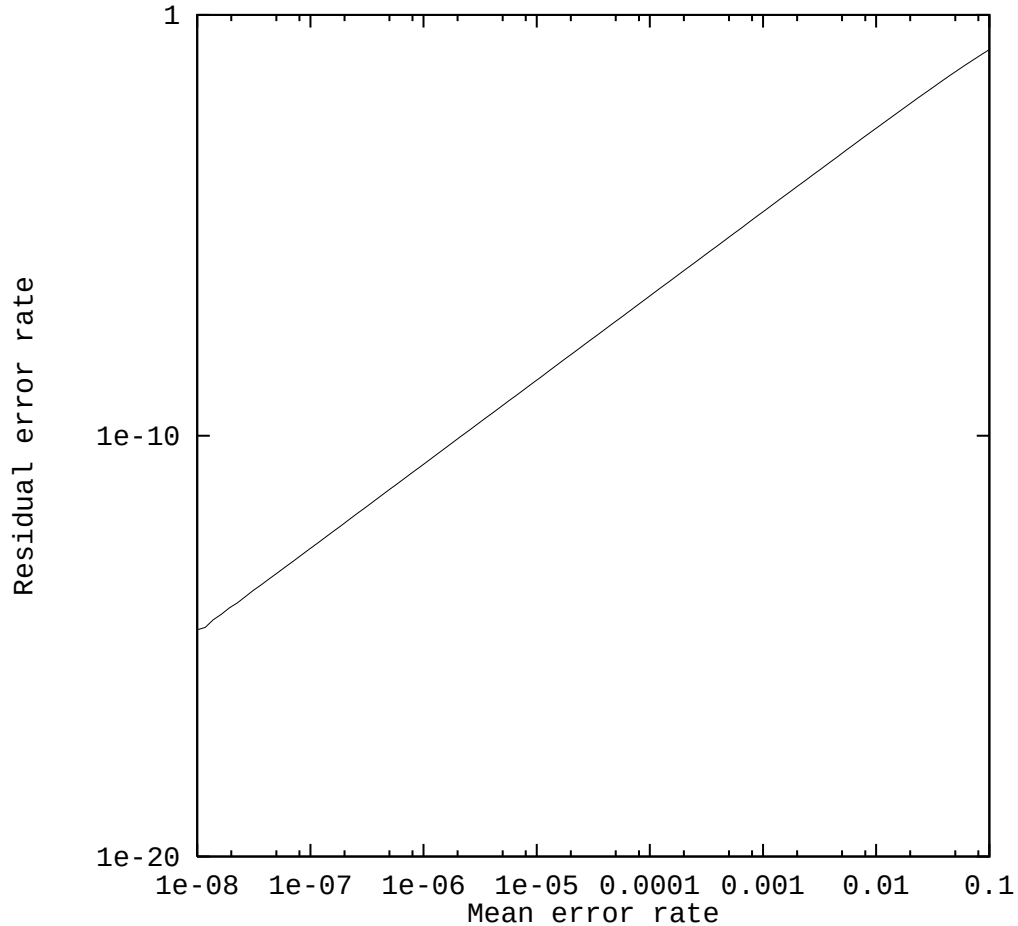
Hamming codes are called perfect because they use m bits to correct $2^m - 1$ error patterns (in this case 3 bits to correct 7 error patterns), and transmit $2^m - 1 - m$ (in this case 4) useful bits.

Finally we observe that in the more general case in which more than 1 bit might be corrupted in each block of 7 bits, then this scheme too will fail. The residual error probability P_e is again given by the remainder of a binomial series:

$$P_e = \sum_{i=2}^{7} \binom{7}{i} \epsilon^i (1-\epsilon)^{7-i}$$

as $i = 2$ or more bits in each block of 7 are corrupted, and $7 - i$ are not.

The plot shows how P_e depends on unconstrained bit error probability ϵ.



Many more robust block codes exist; for example the Golay code embeds 11 error-correcting bits into codeword blocks of 23 bits, enabling it to correct up to 3 bit errors in the block, and transmit 12 useful data bits.

5. Information represented in vector spaces by projections

Often it is useful to represent data in vector spaces, in subspaces, or as linear combinations of basis functions. New aspects of data emerge.

Example 1: u is a vector of data or samples, and $\{e_1, e_2, e_3, ..., e_n\}$ are the basis vectors of some orthonormal system. The projection of u into that space yields coefficients a_i computed as the inner product $a_i = \langle u, e_i \rangle$ of vector u with each basis vector e_i. The a_i represent u in terms of the e_i:

$$u = \sum_{i=1}^{n} a_i e_i \equiv \sum_{i=1}^{n} \langle u, e_i \rangle e_i$$

Example 2: a continuous function or signal $f(x)$ can be represented as a linear combination of continuous basis functions $\Psi_i(x)$ such that

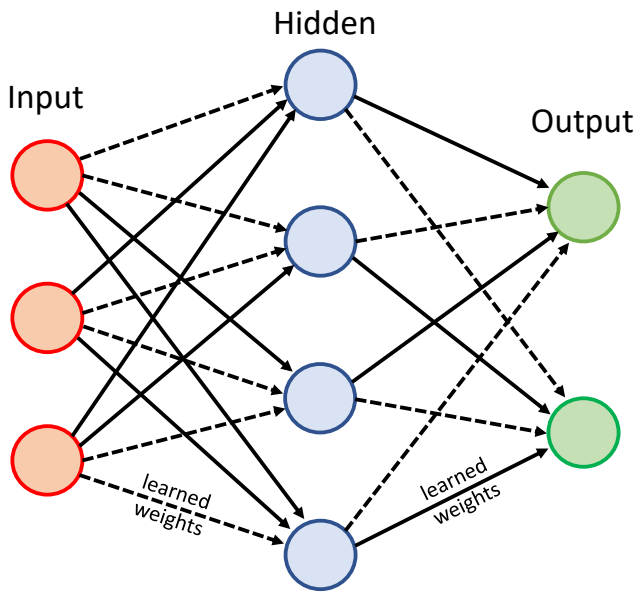
$$f(x) = \sum_{i=1}^{n} a_i \Psi_i(x)$$

where the coefficients a_i that represent $f(x)$ in terms of the $\Psi_i(x)$ are computed by an inner product integral:

$$a_i = \langle f, \Psi_i \rangle \equiv \int f(x) \Psi_i(x) dx$$

Vector spaces, linear combinations, span, bases, subspaces, linear independence, projection, inner products, norms

Linear projections and transforms are ubiquitous in the Data Sciences. For example, a core "building block" of Convolutional Neural Networks (CNNs) which lie at the heart of machine learning and AI today, is a propagation function in the form of an inner product between a vector of inputs (perhaps output from a preceding layer), and a vector of learned weights (acquired and tuned in a training process).



We will review some key concepts about vector spaces and operations within them in quite general form, before specific approaches and cases.

Definition (Linear combination, span, subspace)

If V is a vector space and $\{v_1, v_2, \ldots, v_n\} \in V$ are vectors in V then $u \in V$ is a linear combination of $v_1, v_2, \ldots, v_n$ if there exist scalars $a_1, a_2, \ldots, a_n$ such that $u = a_1 v_1 + a_2 v_2 + \cdots + a_n v_n$.

We also define the span of a set of vectors as all such linear combinations:

$$\text{span}\{v_1, v_2, \ldots, v_n\} = \{u \in V \,:\, u \text{ is a linear combination of } v_1, v_2, \ldots, v_n\}\,.$$

The span of a set of vectors is "everything that can be represented" by linear combinations of them.

A subset $W \subset V$ of the vectors in V constitute a linear subspace of V.

Finding the representation of a function or of data in a linear subspace is to project it onto only that subset of vectors. This may amount to finding an approximation, or to extracting (say) just the low-frequency structure of the data or signal.

Projecting onto a subspace is sometimes called dimensionality reduction.

Definition (Linear dependence and independence)

For vector space V, vectors $v_1, v_2, \ldots, v_n \in V$ are linearly independent if, for scalars $a_1, a_2, \ldots, a_n$, whenever

$$a_1 v_1 + a_2 v_2 + \cdots + a_n v_n = \vec{0}$$

then $a_1 = a_2 = \cdots = a_n = 0$

The vectors $v_1, v_2, \ldots, v_n$ are linearly dependent otherwise.

Linear independence of the vectors in V means that none of them can be represented by any linear combination of others. They are non-redundant: no combination of some of them can "do the work" of another.

Definition (Basis)

A finite set of vectors $v_1, v_2, \ldots, v_n \in V$ is a basis for vector space V if $v_1, v_2, \ldots, v_n$ are linearly independent and $V = \text{span}\{v_1, v_2, \ldots, v_n\}$. The number n is called the dimension of V, written $n = \dim(V)$.

A geometric interpretation and example: any point in the familiar 3 dim Euclidean space $\mathbb{R}^3$ around us can be reached by a linear combination of 3 linearly independent vectors, such as the canonical "(x, y, z) axes." But this would not be possible if the 3 vectors were co-planar; then they would not be linearly independent because any one of them could be represented by a linear combination of the other two, and they would span a space whose dimension was only 2. Note that linear independence of vectors neither requires nor implies orthogonality of the vectors.

A result from linear algebra is that while there are infinitely many choices of basis vectors, any two bases will always consist of the same **number** of element vectors. Thus, the dimension of a linear space is well-defined.

Inner products and inner product spaces

Suppose that V is either a real or complex vector space.

Definition (Inner product)

The inner product of two vectors $u, v \in V$, written in bracket notation $\langle u, v \rangle$, is a scalar value (which may be real or complex) satisfying:

1. For each $v \in V$, $\langle v, v \rangle$ is a non-negative real number, so $\langle v, v \rangle \geq 0$
2. For each $v \in V$, $\langle v, v \rangle = 0$ if and only if $v = \vec{0}$
3. For all $u, v, w \in V$ and for scalars a, b, we have: $\langle au + bv, w \rangle = a\langle u, w \rangle + b\langle v, w \rangle$
4. For all $u, v \in V$ then $\langle u, v \rangle = \overline{\langle v, u \rangle}$.

Here, $\overline{\langle v, u \rangle}$ denotes the complex conjugate of the complex number $\langle v, u \rangle$. Note that for a real vector space, the complex conjugate is redundant. In that case the fourth condition above just says that $\langle u, v \rangle = \langle v, u \rangle$. But inner product order matters for complex vectors.

A vector space together with an inner product is an inner product space.

Useful properties of the inner product

Before looking at some examples of inner products there are several consequences of the definition of an inner product that are useful in calculations.

1. For all $v \in V$ and for scalar a, then $\langle av, av \rangle = |a|^2 \langle v, v \rangle$
2. For all $v \in V$, $\langle \vec{0}, v \rangle = 0$
3. For all $v \in V$ and finite sequences of vectors $u_1, u_2, \ldots, u_n \in V$ and for scalars $a_1, a_2, \ldots, a_n$ then

$$\left\langle \sum_{i=1}^{n} a_i u_i, v \right\rangle = \sum_{i=1}^{n} a_i \langle u_i, v \rangle$$

$$\left\langle v, \sum_{i=1}^{n} a_i u_i \right\rangle = \sum_{i=1}^{n} \overline{a_i} \langle v, u_i \rangle$$

Again we see that in complex vector spaces, the order matters for inner products: note $\overline{a_i}$, the complex conjugate of a_i in the second equation.

Inner product: examples

Example (Euclidean space, $\mathbb{R}^n$)

$V = \mathbb{R}^n$ with the usual operations of vector addition, and multiplication by a real-valued scalar, is a vector space over the scalars $\mathbb{R}$. Given two vectors $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ in $\mathbb{R}^n$ we can define an inner product by

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i \, .$$

Often this inner product is called the dot product and is written $x \cdot y$

Example (space of complex vectors, $V = \mathbb{C}^n$)

Similarly, for complex vector space $V = \mathbb{C}^n$, we can define an inner product by

$$\langle x, y \rangle = x \cdot y = \sum_{i=1}^{n} x_i \overline{y_i}$$

These inner products are projections of vectors onto each other.

Example (Space of continuous functions on an interval)

$V = C[a, b]$, the space of continuous functions $f : [a, b] \to \mathbb{C}$ with the standard operations of the sum of two functions, and multiplication by a scalar, is a vector space over complex scalars $\mathbb{C}$, and we can define an inner product for $f, g \in C[a, b]$ by

$$\langle f, g \rangle = \int_a^b f(x)\overline{g(x)}dx\,.$$

Note that now the "vectors" have become continuous functions instead.

This generalisation can be regarded as the limit in which the number of vector elements becomes infinite, having the density of the reals. The discrete summation over products of corresponding vector elements in our earlier formulation of inner product then becomes, in this limit, a continuous integral of the product of two functions over an interval.

Definition (Norm)

Let V be a real or complex vector space. A norm on V is a function from V to $\mathbb{R}_+$, written $||v||$ and closely related to an inner product, that has the following properties:

1. For all $v \in V$, $||v|| \geq 0$
2. $||v|| = 0$ if and only if $v = \vec{0}$
3. For each $v \in V$ and for some scalar a, $||av|| = |a|\,||v||$
4. For all $u, v \in V$, $||u + v|| \leq ||u|| + ||v||$ (the triangle inequality).

A norm can be thought of as the length of a vector or as a generalisation of the notion of the distance between two vectors $u, v \in V$: the number $||u - v||$ is the distance between u and v.

Example (Euclidean: natural norm for an inner product space)

For a vector space $V = \mathbb{R}^n$ or $\mathbb{C}^n$ and for $x = (x_1, x_2, \ldots, x_n) \in V$, define

$$||x|| = +\sqrt{\langle x, x \rangle} = +\sqrt{\sum_{i=1}^{n} |x_i|^2}\,.$$

Application in feature detection and pattern recognition

- Inner products can be taken to seek evidence of particular features, or patterns, in a vector of data.
- By projecting a vector of input data u (which might be, for example, an array of pixels) onto a stored or a learned vector v representing a feature or a pattern (such as a face), their inner product $\langle u, v \rangle$ is a measure of their degree of agreement.
- If implemented as a convolution, so the input pixel array is a shifting subset of the pixels in a larger image, the location where the largest inner product $\langle u, v \rangle$ is found reveals the position of a best matching pattern (e.g. a certain face) within the larger image.
- The norm $||u - v||$ of the distance between the input vector u and the learned pattern vector v is a measure of how different they are.

Clearly, metrics like inner product and norm can be extremely useful in feature detection, pattern matching, or classification.

Orthogonal and orthonormal systems

Let V be an inner product space and choose the natural Euclidean norm.

Definition (Orthogonality)

We say that $u, v \in V$ are orthogonal (written $u \perp v$) if $\langle u, v \rangle = 0$.

Definition (Orthogonal system)

A finite or infinite sequence of vectors $\{u_i\}$ in V is an orthogonal system if

1. $u_i \neq \vec{0}$ for all such vectors u_i
2. $u_i \perp u_j$ for all $i \neq j$.

Definition (Orthonormal system)

An orthogonal system is called an orthonormal system if, in addition, $||u_i|| = 1$ for all such vectors u_i.

A vector $u \in V$ with unit norm, $||u|| = 1$, is called a unit vector.

We use the special notation e_i for such unit vectors u_i comprising an orthonormal system.

Theorem

Suppose that $\{e_1, e_2, \ldots, e_n\}$ *is an orthonormal system in the inner product space* V*. If* $u = \sum_{i=1}^{n} a_i e_i$ *then* $a_i = \langle u, e_i \rangle$.

(Another way to say this is that in an orthonormal system,
the expansion coefficients are the same as the projection coefficients.)

Proof.

$$\begin{aligned}\langle u, e_i \rangle &= \langle a_1 e_1 + a_2 e_2 + \cdots + a_n e_n, e_i \rangle \\ &= a_1 \langle e_1, e_i \rangle + a_2 \langle e_2, e_i \rangle + \cdots + a_n \langle e_n, e_i \rangle \\ &= a_i \, .\end{aligned}$$

□

Hence, if $\{e_1, e_2, \ldots, e_n\}$ is an orthonormal system, then for all $u \in \operatorname{span}\{e_1, e_2, \ldots, e_n\}$ we have

$$u = \sum_{i=1}^{n} a_i e_i = \sum_{i=1}^{n} \langle u, e_i \rangle e_i \, .$$

Infinite orthonormal systems

We now consider the situation of an inner product space, V, with $\dim(V) = \infty$ and consider orthonormal systems $\{e_1, e_2, \ldots\}$ consisting of infinitely many vectors.

Definition (Convergence in norm)

Let $\{u_1, u_2, \ldots\}$ be an infinite sequence of vectors in the normed vector space V, and let $\{a_1, a_2, \ldots\}$ be some sequence of scalars. We say that the series $\sum_{n=1}^{\infty} a_n u_n$ converges in norm to $w \in V$ if

$$\lim_{m\to\infty} ||w - \sum_{n=1}^{m} a_n u_n|| = 0\,.$$

This means that the (infinite dimensional) vector w *would* be exactly represented by a linear combination of the vectors $\{u_i\}$ in the space V, in the limit that we could use *all* of them. This property of an infinite orthonormal system in an inner product space is called closure.

Remarks on orthonormal systems that are "closed"

- If the system is closed it may still be that the required number m of terms in the above linear combination for a "good" approximation is too great for practical purposes.
- Seeking alternative closed systems of orthonormal vectors may produce "better" approximations in the sense of requiring fewer terms for a given accuracy. The best system for representing a particular dataset will depend on the dataset. (Example: faces.)
- There exists a numerical method for constructing an orthonormal system $\{e_1, e_2, \ldots\}$ such that any given set of vectors $\{u_1, u_2, \ldots\}$ (which are often a set of multivariate data) can be represented within it with the best possible accuracy using any specified finite **number** of terms. Optimising the approximation under truncation requires deriving the orthogonal system $\{e_1, e_2, \ldots\}$ **from** the data set $\{u_1, u_2, \ldots\}$. This is called the Karhunen-Loève transform or alternatively the Hotelling transform, or Dimensionality Reduction, or Principal Components Analysis, and it is used in statistics and in exploratory data analysis, but it is outside the scope of this course.

6. Fourier representations of information

The decomposition of functions (signals, data, patterns, ...) into superpositions of elementary sinusoidal functions underlies much of science and engineering. It allows many problems to be solved.

One reason is **Physics:** many physical phenomena[1] such as wave propagation (e.g. sound, water, radio waves) are governed by linear differential operators whose eigenfunctions (unchanged by propagation) are the complex exponentials: $e^{i\omega x} = \cos(\omega x) + i \sin(\omega x)$

Another reason is **Engineering:** the most powerful analytical tools are those of *linear systems analysis*, which allow the behaviour of a linear system in response to *any* input to be predicted by its response to just *certain* inputs, namely those eigenfunctions, the complex exponentials.

A further reason is **Computational Mathematics:** when phenomena, patterns, data or signals are represented in Fourier terms, very powerful manipulations become possible. For example, extracting underlying forces or vibrational modes; the atomic structure revealed by a spectrum; the identity of a pattern under transformations; or the trends and cycles in economic data, asset prices, or medical vital signs.

[1] A tuned mechanical resonator (Tacoma Narrows Bridge): `http://www.youtube.com/watch?v=j-zczJXSxnw`

So who was Fourier and what was his insight?

Jean Baptiste Joseph Fourier (1768 – 1830)

(Quick sketch of a Frenchman whose fortunes were sinusoidal)

Orphaned at 8. Attended military school hoping to join the artillery but was refused and sent to a Benedictine school to prepare for Seminary.

The French Revolution interfered. Fourier promoted it, but he was arrested in 1794 because he had then defended victims of the Terror. Fortunately, Robespierre was executed first, and so Fourier was spared.

In 1795 his support for the Revolution was rewarded by a chair at the École Polytechnique. Soon he was arrested again, this time accused of having supported Robespierre. He escaped the guillotine twice more.

Napoleon selected Fourier for his Egyptian campaign and later elevated him to a barony. Fourier was elected to the Académie des Sciences but Louis XVII overturned this because of his connection to Napoleon.

He proposed his famous sine series in a paper on the theory of heat, which was rejected at first by Lagrange, his own doctoral advisor. He proposed the "greenhouse effect." Believing that keeping one's body wrapped in blankets to preserve heat was beneficial, in 1830 Fourier died after tripping in this condition and falling down his stairs. His name is inscribed on the Eiffel Tower.

Fourier series

We now consider representation of information in piecewise continuous functions f, for convenience just over the interval $[-\pi, \pi]$, by projecting them into this vector space $\{e_1, e_2, \ldots\}$:

$$\left\{ \frac{1}{\sqrt{2}}, \sin(x), \cos(x), \sin(2x), \cos(2x), \sin(3x), \cos(3x), \ldots \right\}$$

It is easily shown that this is a closed infinite orthonormal system, because for each of the $\{e_1, e_2, \ldots\}$ functions above, we have:

$$\langle e_i, e_j \rangle = 0, \quad i \neq j$$

and

$$||e_i|| = +\sqrt{\langle e_i, e_i \rangle} = 1, \quad \forall i$$

From the properties of closed orthonormal systems we know that we can represent any piecewise continuous function f by a linear combination

$$\sum_{n=1}^{\infty} \langle f, e_n \rangle e_n \, .$$

Fourier coefficients

Thus in this orthonormal system, the linear combination

$$\sum_{n=1}^{\infty} \langle f, e_n \rangle e_n$$

becomes the familiar Fourier series for a function f, namely

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nx) + b_n \sin(nx)]$$

where

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(nx)\, dx, \quad n = 0, 1, 2, \ldots$$
$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(nx)\, dx, \quad n = 1, 2, 3, \ldots .$$

Note how the constant term is written $a_0/2$ where $a_0 = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) dx$.

Periodic functions

Our Fourier series

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nx) + b_n \sin(nx)]$$

defines a function, say $g(x)$, that is 2π-periodic in the sense that

$$g(x + 2\pi) = g(x), \qquad \text{for all } x \in \mathbb{R}\,.$$

Hence, it is convenient to extend f to a 2π-periodic function defined on $\mathbb{R}$ instead of being restricted to $[-\pi, \pi]$.

This finesse will prove important later, when we discuss the Discrete Fourier Transform and the Fast Fourier Transform algorithm for datasets that are not actually periodic. In effect, such datasets of whatever length are regarded as just one "period" within endlessly repeating copies of themselves. To define the continuous Fourier transform of an aperiodic continuous function, we will regard its period as being infinite, and the increment of frequencies (index n above) will become infinitesimal.

Even and odd functions

A particularly useful simplification occurs when the function f is either an even function, that is, for all x,

$$f(-x) = f(x)$$

or an odd function, that is, for all x,

$$f(-x) = -f(x)\,.$$

The following properties can be easily verified.

1. If f, g are even then fg is even
2. If f, g are odd then fg is even
3. If f is even and g is odd then fg is odd
4. If g is odd then for any $h > 0$, we have $\int_{-h}^{h} g(x)dx = 0$
5. If g is even then for any $h > 0$, we have $\int_{-h}^{h} g(x)dx = 2\int_{0}^{h} g(x)dx$.

Even functions and cosine series

Recall that the Fourier coefficients are given by

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(nx)\,dx, \quad n = 0, 1, 2, \ldots$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(nx)\,dx, \quad n = 1, 2, 3, \ldots$$

so if f is even then they become

$$a_n = \frac{2}{\pi}\int_{0}^{\pi} f(x)\cos(nx)\,dx, \quad n = 0, 1, 2, \ldots$$

$$b_n = 0, \quad n = 1, 2, 3, \ldots .$$

Odd functions and sine series

Similarly, the Fourier coefficients

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(nx)\,dx, \quad n = 0, 1, 2, \ldots$$
$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(nx)\,dx, \quad n = 1, 2, 3, \ldots,$$

for the case where f is an odd function become

$$a_n = 0, \quad n = 0, 1, 2, \ldots$$
$$b_n = \frac{2}{\pi}\int_{0}^{\pi} f(x)\sin(nx)\,dx, \quad n = 1, 2, 3, \ldots.$$

Thus, the Fourier series for even functions require only cosine terms. The Fourier series for odd functions require only sine terms. In both cases, the integrals for obtaining their coefficients involve only half the real line.

Fourier series: example 1: sawtooth function

Consider $f(x) = x$ for $x \in [-\pi, \pi]$, so f is clearly odd and thus we need to calculate a sine series with coefficients, b_n, $n = 1, 2, \ldots$ given by

$$\begin{aligned} b_n &= \frac{2}{\pi}\int_0^\pi x\sin(nx)\,dx = \frac{2}{\pi}\left\{\left[-x\frac{\cos(nx)}{n}\right]_0^\pi + \int_0^\pi \frac{\cos(nx)}{n}dx\right\} \\ &= \frac{2}{\pi}\left\{-\pi\frac{(-1)^n}{n} + \left[\frac{\sin(nx)}{n^2}\right]_0^\pi\right\} \\ &= \frac{2}{\pi}\left\{-\pi\frac{(-1)^n}{n} + 0\right\} = \frac{2(-1)^{n+1}}{n}\,. \end{aligned}$$

Hence the Fourier series of $f(x) = x$ on $x \in [-\pi, \pi]$ is

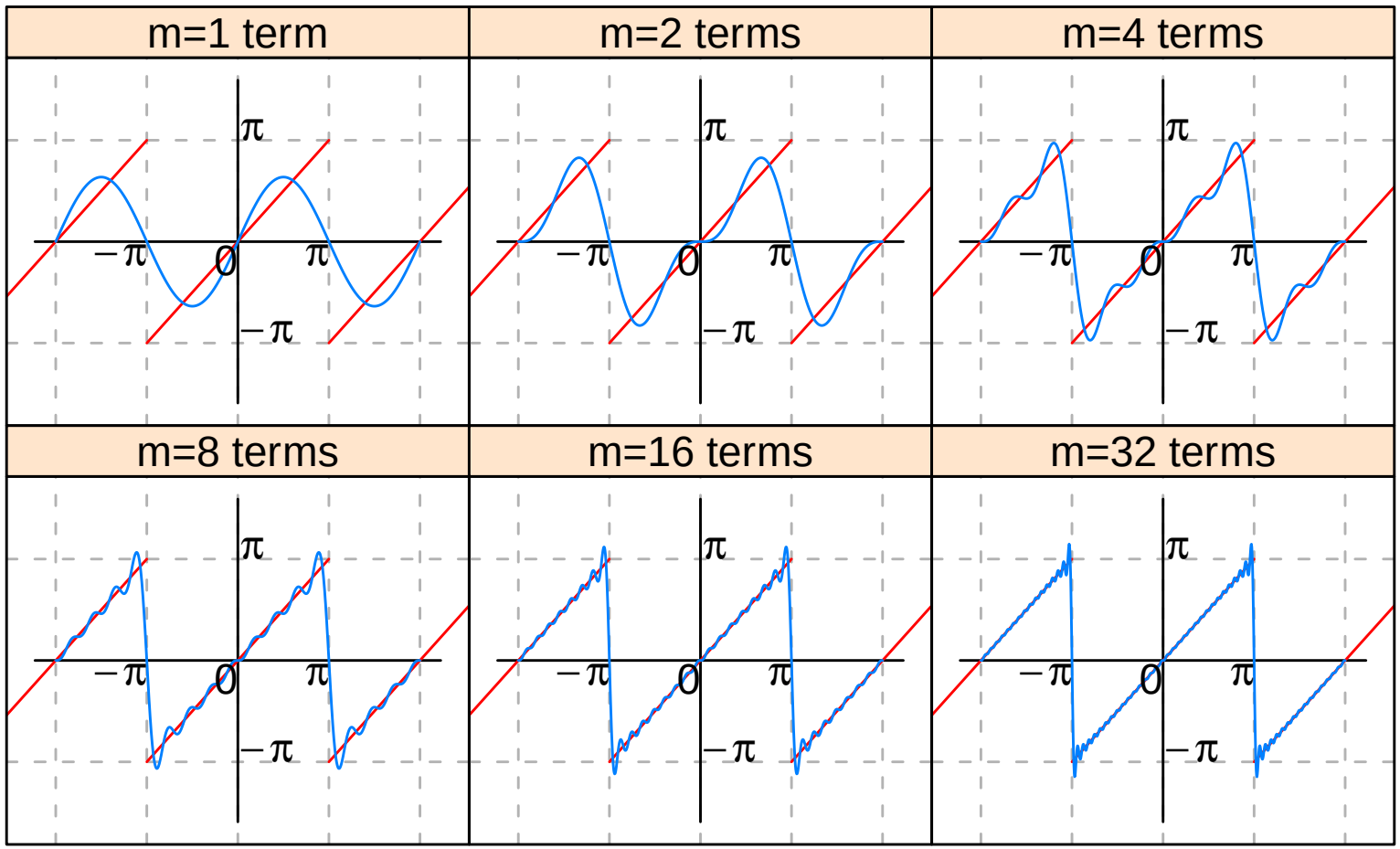
$$\sum_{n=1}^{\infty} \frac{2(-1)^{n+1}}{n}\sin(nx)\,.$$

Observe that the series does *not* agree with $f(x)$ at $x = \pm\pi$, where derivatives are infinite — a matter known as the Gibbs phenomenon.

(example 1, con't)

Let us examine plots of the partial sums to m terms

$$\sum_{n=1}^{m} \frac{2(-1)^{n+1}}{n} \sin(nx)\,.$$



Fourier series: example 2: trianglewave function

Now suppose $f(x) = |x|$ for $x \in [-\pi, \pi]$ which is clearly an even function so we need to construct a cosine series with coefficients

$$a_0 = \frac{2}{\pi}\int_0^{\pi} x dx = \frac{2}{\pi}\frac{\pi^2}{2} = \pi$$

and for $n = 1, 2, \ldots$

$$a_n = \frac{2}{\pi}\int_0^{\pi} x\cos(nx)\,dx = \frac{2}{\pi}\left\{\left[\frac{x\sin(nx)}{n}\right]_0^{\pi} - \int_0^{\pi}\frac{\sin(nx)}{n}dx\right\}$$
$$= \frac{2}{\pi}\left\{\left[\frac{\cos(nx)}{n^2}\right]_0^{\pi}\right\} = \frac{2}{\pi}\left\{\frac{(-1)^n - 1}{n^2}\right\} = \begin{cases} -\frac{4}{\pi n^2} & n \text{ is odd} \\ 0 & n \text{ is even} \end{cases}.$$

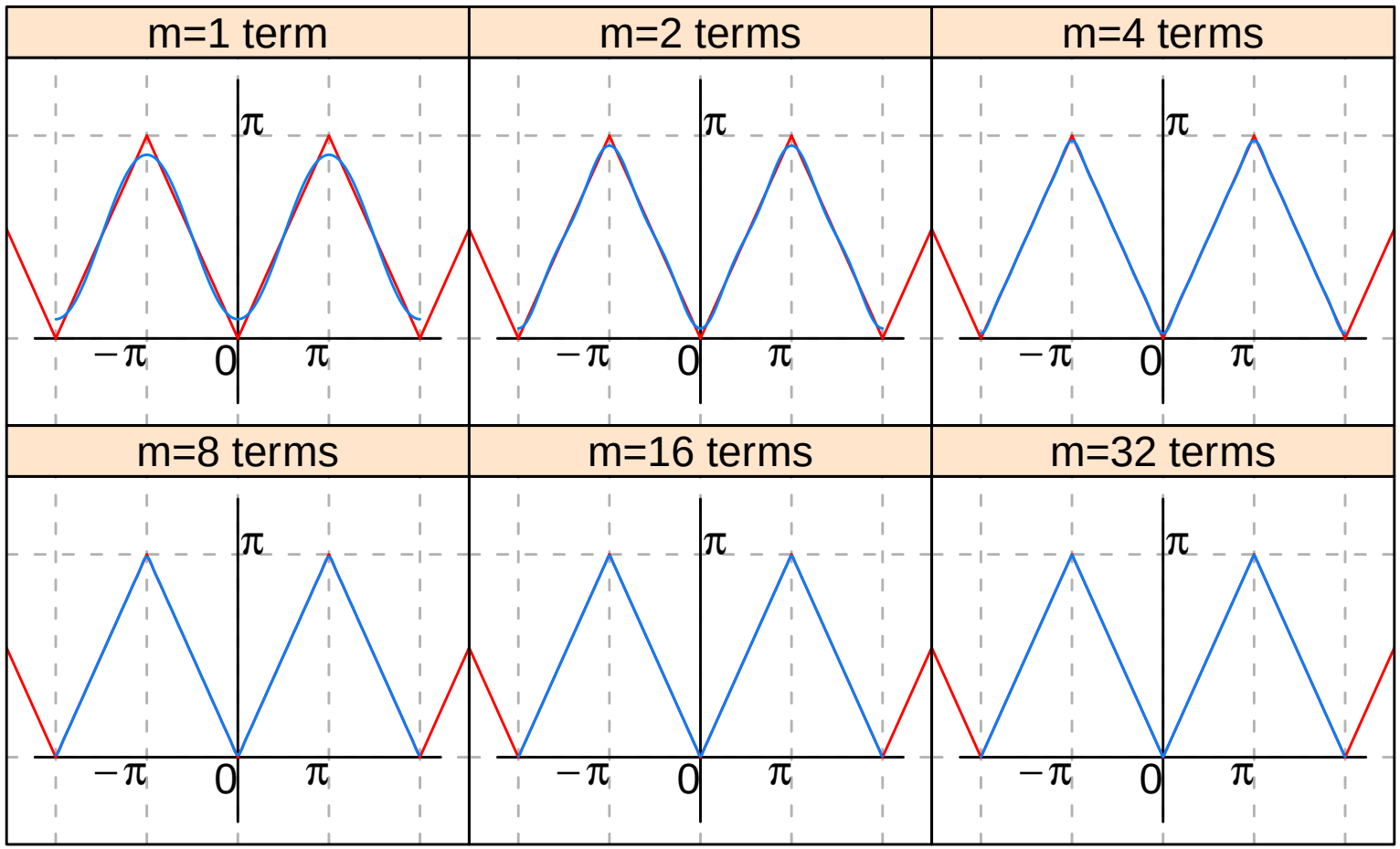
Hence, the Fourier series of $f(x) = |x|$ on $x \in [-\pi, \pi]$ is

$$\frac{\pi}{2} - \sum_{k=1}^{\infty}\frac{4}{\pi(2k-1)^2}\cos\left((2k-1)x\right).$$

(example 2, con't)

Let us examine plots of the partial sums to m terms

$$\frac{\pi}{2} - \sum_{k=1}^{m} \frac{4}{\pi(2k-1)^2} \cos\left((2k-1)x\right) .$$



Animation showing rates of convergence: `https://www.youtube.com/watch?v=Lu2nnvYORec`

Complex Fourier series

We have used real-valued functions $\sin(nx)$ and $\cos(nx)$ as our orthonormal system, but we can also use complex-valued functions. In this case, the inner product is

$$\langle f, g \rangle = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x)\overline{g(x)}dx\,.$$

A suitable orthonormal system which captures the earlier (sine, cosine) Fourier series approach is the collection of functions

$$\left\{1, e^{ix}, e^{-ix}, e^{i2x}, e^{-i2x}, \ldots\right\}\,.$$

Then we have a representation called the complex Fourier series of f, given by

$$\sum_{n=-\infty}^{\infty} c_n e^{inx}$$

where

$$c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x)e^{-inx}dx, \quad n = 0, \pm 1, \pm 2, \ldots\,.$$

(Complex Fourier series, con't)

Euler's formula, $e^{ix} = \cos(x) + i\sin(x)$, gives for $n = 1, 2, \ldots$ that

$$e^{inx} = \cos(nx) + i\sin(nx)$$
$$e^{-inx} = \cos(nx) - i\sin(nx)$$

and $e^{i0x} = 1$. Using these relations it can be shown that for $n = 1, 2, \ldots$

$$c_n = \frac{a_n - ib_n}{2}, \qquad c_{-n} = \frac{a_n + ib_n}{2}.$$

Hence,

$$a_n = c_n + c_{-n}, \qquad b_n = i(c_n - c_{-n})$$

and

$$c_0 = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-i0x}dx = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)dx = \frac{a_0}{2}.$$

Poetry: the fundamental constants $\{0, 1, i, e, \pi\}$ combine as $e^{i\pi} + 1 = 0$.

Joke recorded on a mathematician's answering machine: *"I am sorry but you have dialed an imaginary number. Please rotate your phone by 90 degrees and try again."*

7. Spectral properties of continuous-time channels

- *Continuous-time signals as bandlimited carriers of information.*
- *Signals represented as superpositions of complex exponentials.*
- *Eigenfunctions of channels modelled as linear time-invariant systems.*
- *Continuous-time channels as spectral filters with added noise.*

Both signals and the channels that transmit them are ultimately physical systems, with spectral and information properties determined by physics. Understanding these physical properties leads to important insights.

Information channels are typically an assigned spectral band of some medium, within which a carrier signal is modulated in certain of its parameters (sine frequency, amplitude, or phase) to encode information. The resulting complex $f(t)$ has a certain bandwidth W in Hertz, which is related to its information capacity C in bits/sec. By Fourier analysis we may regard $f(t)$ as a superposition of complex exponentials:

$$f(t) = \sum_n c_n e^{(i\omega_n t)}$$

We can understand the process of sending signals through channels in Fourier terms, because channels are (ideally) linear time-invariant systems whose eigenfunctions are in fact complex exponentials:

$$e^{(i\omega_n t)} \longrightarrow \boxed{h(t)} \longrightarrow \alpha\ e^{(i\omega_n t)}$$

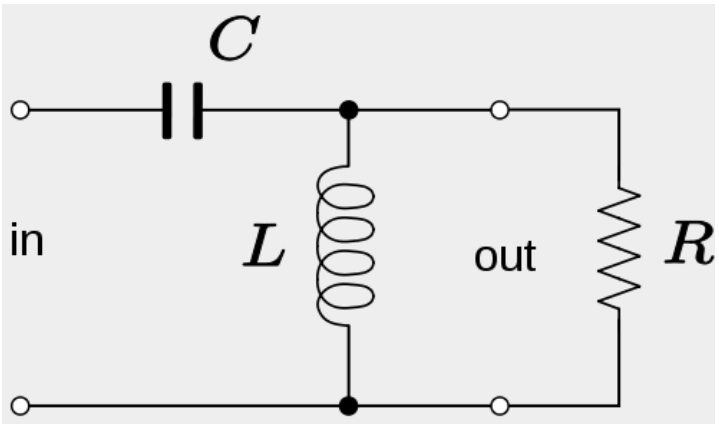
Linear time-invariant systems (*e.g.* a coaxial cable, or the air through which spoken sounds pass, or the electromagnetic spectrum in space) obey the properties of superposition and proportionality, and can always be described by some linear operator $h(t)$. Examples of $h(t)$ may include: a derivative, or combination of them making a differential operator; or a convolution. The point is that a complex exponential is never changed in its form by being acted on by a linear operator; it is only multiplied by a complex constant, α. This is the eigenfunction property.

The consequence is that if we know the amplitude-and-phase changing values α_n for the relevant frequencies ω_n in a particular channel, then we simply incorporate them into the earlier Fourier series expansion of $f(t)$ in order to understand how $f(t)$ is spectrally affected by transmission through that channel:

$$f(t) = \sum_n \alpha_n c_n e^{(i\omega_n t)}$$

Frequency-selective linear circuits: filters

Signals (e.g. audio signals expressed as a time-varying voltage) can be regarded as combinations of frequencies, whose amplitudes and phases are changed when passing through circuits that act as frequency filters.



Simple linear analogue circuit elements have a *complex impedance*, Z, which expresses their frequency-dependent behaviour and reveals what sorts of filters they will make when combined in various configurations.

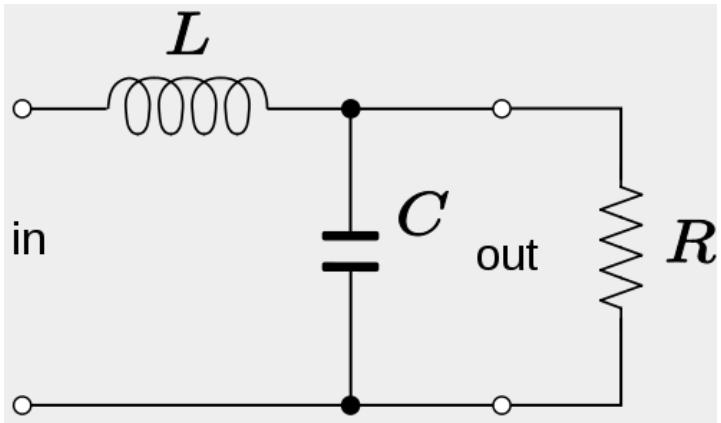
Resistors (R in ohms) just have a constant impedance: $Z = R$; but...

Capacitors (C in farads) have low impedance at high frequencies ω, and high impedance at low frequencies: $Z(\omega) = \frac{1}{i\omega C}$

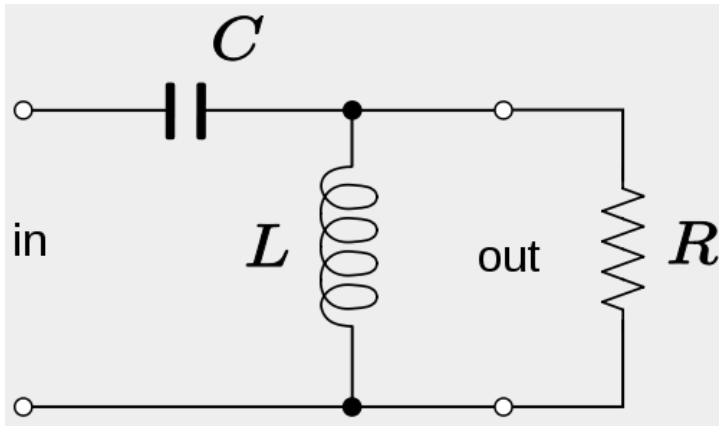
Inductors (L in henrys) have high impedance at high frequencies ω, and low impedance at low frequencies: $Z(\omega) = i\omega L$

(Frequency-selective simple filters, con't)

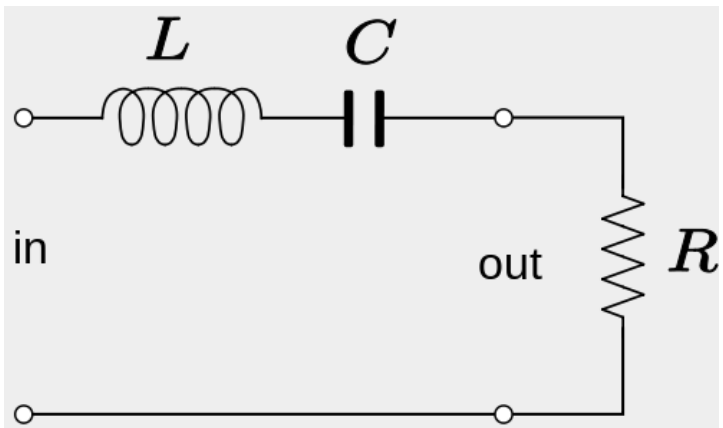
The equations relating voltage to current flow through circuit elements with impedance Z (of which Ohm's Law is a simple example) allow systems to be designed with specific Fourier (frequency-dependent) properties, including filters, resonators, and tuners. Today these may be implemented digitally.



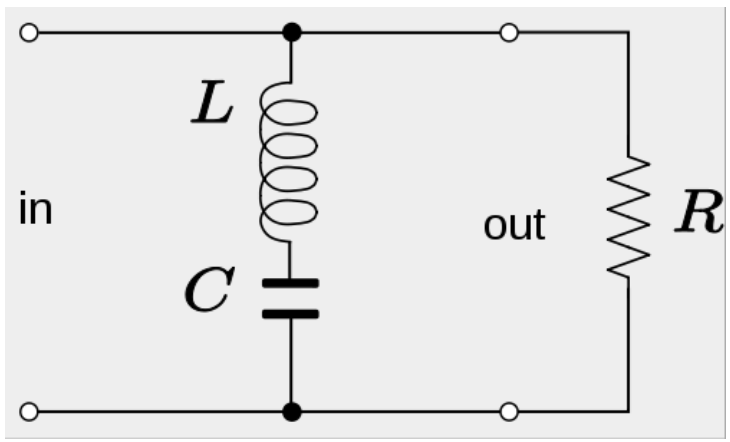
Low-pass filter: higher frequencies are attenuated.



High-pass filter: lower frequencies are rejected.



Band-pass filter: only middle frequencies pass.

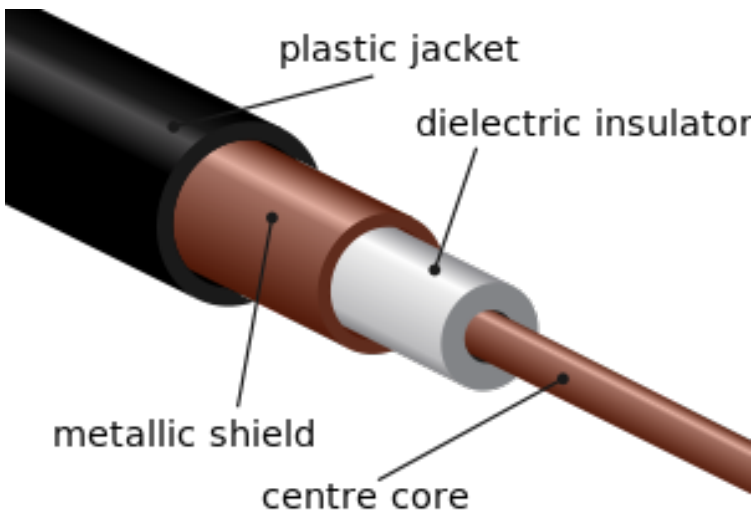


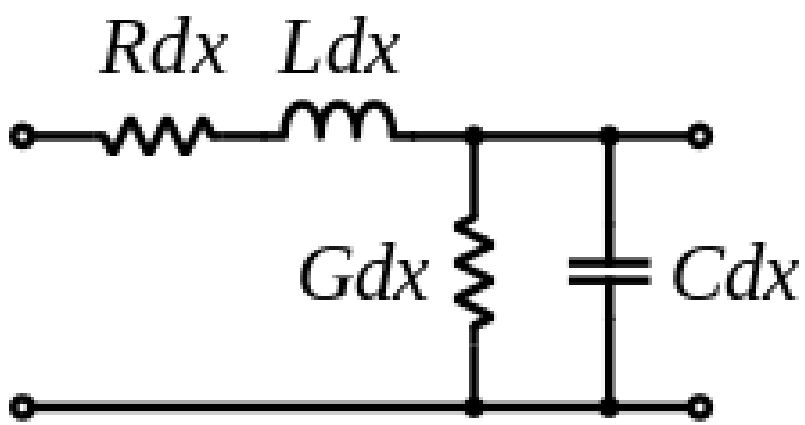
Band-reject filter: middle frequencies attenuate.

Bandwidth limitation of communication channels

A coaxial cable is one example of a communication channel. Its physical properties (distributed per unit length, hence the differential dx notation) include:

- a non-zero series resistance R;
- a non-zero series inductance L;
- a non-zero shunt conductance G (or non-infinite resistance) through the insulator separating signal from ground;
- and a non-zero shunt capacitance C between signal and ground.





This "equivalent circuit" for a physical communication channel makes it a low-pass filter: its ability to transmit signals is restricted to some finite bandwidth W, in Hertz. Frequency components higher than about $W \approx 1/(RC)$ are significantly attenuated by the signal pathway between conductor and shield (signal and ground). Although the attenuation is first-order and therefore gradual (6 dB per octave in terms of amplitude), still we must take into account this limited bandwidth W in Hertz.

How do you expect the band limitation W will influence the information carrying capacity of a channel?

Another physical limitation of communication channels is that they add some wideband noise to the signal during transmission. This is generally thermal noise, or shot noise, but it may come from various sources and generally has a broad spectral density across the channel's bandwidth W. If the noise spectrum is quite uniform, it is called white noise, in analogy with the spectral composition of white light. *Other colours are available.*

How do you expect the added noise, of a certain power, will influence the information carrying capacity of a channel?

8. Continuous information; Noisy Channel Coding Theorem

- *Extensions of discrete entropies and measures to continuous variables.*
- *Gaussian channels; signal-to-noise ratio; power spectral density.*
- *Relative significance of bandwidth and noise limits on channel capacity.*

We turn now to the encoding and transmission of information that is continuously variable in time or space, such as sound, or optical images, rather than discrete symbol sets. Using continuous probability densities, many of our metrics generalise from the discrete case in a natural way.

If the value X that a continuous signal may take (such as voltage, or sound pressure $x(t)$ as a function of time) has some probability density $p(x)$ with $\int_{-\infty}^{+\infty} p(x)dx = 1$, then we define its differential entropy as:

$$h(X) = \int_{-\infty}^{+\infty} p(x) \log_2 \left(\frac{1}{p(x)} \right) dx \, .$$

(We use h for entropy of a continuous variable; H for discrete variables.)

Let $p(x, y)$ be the joint probability distribution of two continuous random variables X and Y. The marginal distribution of either one is obtained by integrating $p(x, y)$ over the other variable, just like the Sum Rule:

$$p(x) = \int_{-\infty}^{+\infty} p(x, y) dy$$

$$p(y) = \int_{-\infty}^{+\infty} p(x, y) dx \, .$$

The joint entropy $h(X, Y)$ of continuous random variables X and Y is

$$h(X, Y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} p(x, y) \log_2 \left(\frac{1}{p(x, y)} \right) dxdy$$

and their conditional entropies $h(X|Y)$ and $h(Y|X)$ are

$$h(X|Y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} p(x, y) \log_2 \left(\frac{p(y)}{p(x, y)} \right) dxdy$$

$$h(Y|X) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} p(x, y) \log_2 \left(\frac{p(x)}{p(x, y)} \right) dxdy \, .$$

We also have the property that $h(X, Y) \leq h(X) + h(Y)$, with the upper bound reached in the case that X and Y are independent.

Finally, the mutual information $i(X; Y)$ between two continuous random variables X and Y is

$$i(X; Y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} p(x, y) \log_2 \left(\frac{p(x|y)}{p(x)} \right) dxdy = h(Y) - h(Y|X)$$

and as before, the capacity C of a continuous communication channel is determined by maximising this mutual information over all possible input distributions $p(x)$ for X, the signal entering the channel.

Thus we need to think about what class of continuous signals have "maximum entropy" for a given amplitude range of excursions, which corresponds essentially to the power level, or variance, of a signal.

We know that entropy is maximised with "equiprobable symbols," which means in the continuous case that if signal value x is limited to some range v, then the probability density for x is uniformly $p(x) = 1/v$.

Gaussian noise maximises entropy for any given variance

For any continuous random variable X, the greater its variance (which corresponds to the power or volume level of a sound signal), the greater its differential entropy $h(X)$. But for any *given* power level or variance σ^2 of the signal's excursions around its mean value μ, it can be proven that the distribution $p(x)$ of those excursions which generates *maximum* $h(X)$ is the Gaussian distribution:

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

in which case the differential entropy is maximised at

$$h(X) = \frac{1}{2} \log_2(2\pi e\sigma^2) \,.$$

Such a signal sounds like "hiss" and it is called white noise, because its power spectrum is flat, just like the spectrum of white light.

The entropy-maximising property of Gaussian noise is important because we compute channel capacity as the mutual information $i(X;Y)$ of the channel when maximised over all possible input distributions $p(x)$.

Additive white Gaussian noise (AWGN) channel

We consider channels into which Gaussian noise (denoted N) is injected, independently of the signal, and added to it. The channel itself has a limited spectral bandwidth W in Hertz, and so both the signal and the added noise are strictly lowpass: they have no frequency components higher than W within the channel.

Over this bandwidth W, the white noise has power spectral density N_0, which gives it a total noise power $N_0 W$ and variance $\sigma^2 = N_0 W$.

Because the noise N is independent of the input signal X, we have $h(Y|X) = h(N)$; and because the noise is Gaussian with $\sigma^2 = N_0 W$,

$$h(Y|X) = h(N) = \frac{1}{2}\log_2(2\pi e\sigma^2) = \frac{1}{2}\log_2(2\pi e N_0 W)\,.$$

The input signal X itself has variance or power P over the spectrum W. As variances add, the channel output $Y = X + N$ has variance $P + N_0 W$.

Now we can ask the question:
What is the capacity C of such a channel, given P, N_0, and W?

Noisy Channel Coding Theorem

Maximising the mutual information of the channel over all possible input distributions requires this calculation to assume that the input X itself has a Gaussian $p(x)$. The mutual information $i(X;Y)$ between the input to the AWGN channel and the output signal Y transmitted is then:

$$\begin{aligned} i(X;Y) &= h(Y) - h(Y|X) \\ &= h(Y) - h(N) \\ &= \frac{1}{2}\log_2(2\pi e(P + N_0 W)) - \frac{1}{2}\log_2(2\pi e N_0 W) \\ &= \frac{1}{2}\log_2 \frac{2\pi e(P + N_0 W)}{2\pi e N_0 W} \\ &= \frac{1}{2}\log_2 \left(1 + \frac{P}{N_0 W}\right) \end{aligned}$$

which gives us the channel capacity C for the noisy signals it emits:

$$C = \frac{1}{2}\log_2 \left(1 + \frac{P}{N_0 W}\right)$$

(Noisy Channel Coding Theorem, con't)

We will see from the sampling theorem that a strictly bandlimited signal whose lowpass bandwidth in Hertz is W (like the output of this channel) is completely specified by sampling it at a rate $2W$ samples per second. Thus we can convert the channel capacity C into bits per second for that sampling rate by multiplying the last expression for C bits by $2W$:

$$C = W \log_2 \left(1 + \frac{P}{N_0 W}\right) \text{ bits/sec}$$

Note that the term inside the logarithm is $1+$ the signal-to-noise ratio, often abbreviated SNR. Because of the logarithm, SNR is often reported in decibels, denoted dB: $10 \times \log_{10}(\text{SNR})$ if SNR is a ratio of power, or $20 \times \log_{10}(\text{SNR})$ if SNR is a ratio of amplitudes. Thus, for example, an amplitude signal-to-noise ratio of 100:1 corresponds to an SNR of 40 dB.

The critical insight about this theorem is that continuous-time channel capacity is dictated primarily by the signal-to-noise ratio!

(Noisy Channel Coding Theorem, con't)

Thus we have arrived at Shannon's third theorem (also called the Shannon-Hartley Theorem), the Noisy Channel Coding Theorem:

> *The capacity of a continuous-time channel, bandlimited to W Hertz, perturbed by additive white Gaussian noise of power spectral density N_0 and bandwidth W, using average transmitted power P, is:*

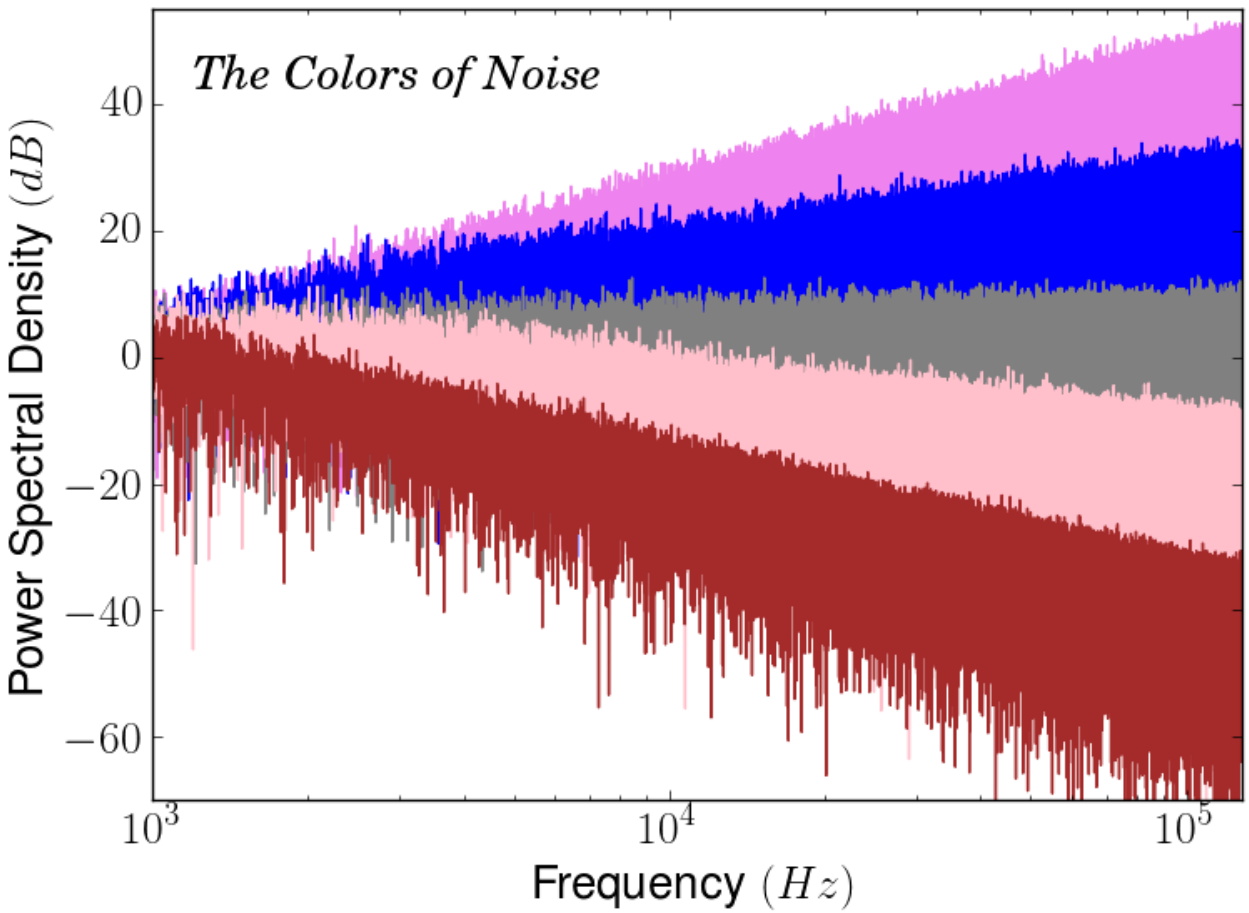
$$C = W \log_2 \left(1 + \frac{P}{N_0 W}\right) \text{ bits/sec}$$

It is noteworthy that increasing the bandwidth W in Hertz yields a monotonic, but asymptotically limited, improvement in capacity because inside the logarithm its effect on the total noise power $N_0 W$ is reciprocal. Using the series: $\log_e(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots$, so $\log_e(1+x) \approx x$ for $x << 1$, we see that the limit as $W \to \infty$ is: $C \to \frac{P}{N_0} \log_2 e$.

But improving the channel's signal-to-noise ratio SNR improves the channel capacity without limit.

Effect of "coloured" noise on channel capacity

The preceding analysis assumed a constant noise power spectral density N_0 over the bandwidth W. But often the noise spectrum is non-uniform. For example, pink noise (also called $1/f$ or flicker noise) has a power spectral density inversely proportional to the frequency. Other named noise "colours" include brown, red, blue, violet, and grey noise.



(Effect of "coloured" noise on channel capacity, con't)

The main named noise "colours" have a power spectral density that varies as a power function of frequency ω, with exponents -2 or -1 for more low frequency noise, $+1$ or $+2$ for more high frequency noise, or 0 for a flat noise spectrum (white noise). In log-log coordinates (previous slide), such power function plots become straight lines whose slope is the exponent.

As SNR thus varies across the frequency band W, one would expect the information capacity also to be non-uniformly distributed. Taking an infinitesimal approach, the channel capacity C in any small portion $\Delta\omega$ around frequency ω where the signal-to-noise ratio follows SNR(ω), is:

$$C_{(\Delta\omega)} = \Delta\omega \log_2(1 + \text{SNR}(\omega)) \quad \text{bits/sec.}$$

Integrating over all of these small $\Delta\omega$ bands in some available range from ω_1 to ω_2, the information capacity of this variable-SNR channel is thus:

$$C = \int_{\omega 1}^{\omega 2} \log_2(1 + \text{SNR}(\omega)) d\omega \quad \text{bits/sec.}$$

9. Encodings based on Fourier transform properties

- We have seen how functions f on the interval $[-\pi, \pi]$ (repeating periodically outside that interval) can be represented using closed orthonormal systems such as the Fourier series

$$\sum_{n=-\infty}^{\infty} c_n e^{inx}$$

where

$$c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-inx} dx \quad n = 0, \pm 1, \pm 2, \ldots .$$

Note that all the Fourier components used in the series have integer frequencies n; or more precisely, the frequencies used are all integer multiples of the repetition frequency, which here had period 2π.

The domain $[-\pi, \pi]$ can be swapped for a general interval $[a, b]$ and the function can be regarded as L-periodic and defined for all $\mathbb{R}$, where $L = (b - a) < \infty$ is the length of the interval.

- We shall now consider the situation where $f(x)$ is a non-periodic ("aperiodic") function. We do this by allowing $L = (b - a) \to \infty$.
- The interval between the frequency components therefore becomes infinitesimal, and they acquire the density of the reals rather than the integers. The Fourier series then becomes a Fourier transform.
- We require that f be piecewise continuous and absolutely integrable:

$$\int_{-\infty}^{\infty} |f(x)| dx < \infty$$

- We can then define the Fourier transform $F(\omega)$ of $f(x)$ with the following properties:
  1. $F(\omega)$ is defined for all $\omega \in \mathbb{R}$
  2. $F(\omega)$ is a continuous function
  3. $\lim_{\omega \to \pm\infty} F(\omega) = 0$

We shall use the notation $F(\omega)$ or $\mathcal{F}_{[f]}(\omega)$ as convenient, and refer to it as "the representation of $f(x)$ in the frequency (or Fourier) domain."

Fourier transform

Definition (Fourier transform)

For $f : \mathbb{R} \to \mathbb{C}$ define the Fourier transform of f to be the function $F : \mathbb{R} \to \mathbb{C}$ given by

$$F(\omega) = \mathcal{F}_{[f]}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx$$

Note two key changes from the Fourier series, now that the function $f(x)$ is no longer restricted to a finite interval or periodically repeating:

1. the bounds of integration are now $[-\infty, \infty]$ instead of $[-\pi, \pi]$.
2. the frequency parameter inside the complex exponential previously took only integer values n, but now it must take all real values ω.

Note also the local-to-global property that $f(x)$ at any point x affects the value of $F(\omega)$ at *every* point ω; and also the global-to-local property that $F(\omega)$ at each point ω is affected by the behaviour of $f(x)$ at *all* points x. This is somewhat reminiscent of holographic data storage.

Example

For $a > 0$, let $f(x) = e^{-a|x|}$. Then the Fourier transform of $f(x)$ is

$$\begin{aligned}
F(\omega) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-a|x|} e^{-i\omega x} dx \\
&= \frac{1}{2\pi} \left\{ \int_{0}^{\infty} e^{-ax} e^{-i\omega x} dx + \int_{-\infty}^{0} e^{ax} e^{-i\omega x} dx \right\} \\
&= \frac{1}{2\pi} \left\{ - \left[\frac{e^{-(a+i\omega)x}}{a+i\omega} \right]_{0}^{\infty} + \left[\frac{e^{(a-i\omega)x}}{a-i\omega} \right]_{-\infty}^{0} \right\} \\
&= \frac{1}{2\pi} \left\{ \frac{1}{a+i\omega} + \frac{1}{a-i\omega} \right\} \\
&= \frac{a}{\pi(a^2+\omega^2)} .
\end{aligned}$$

Observe that $f(x)$ is real and even, and so is its Fourier transform $F(\omega)$. This is part of a more general property relating symmetry and conjugacy.

Properties

Several properties of the Fourier transform are very helpful in calculations.

First, note that by the linearity of integrals we have that for $a, b \in \mathbb{C}$ and for functions f and g, then

$$\mathcal{F}_{[af+bg]}(\omega) = a\mathcal{F}_{[f]}(\omega) + b\mathcal{F}_{[g]}(\omega)$$

Secondly, if f is real-valued then

$$F(-\omega) = \overline{F(\omega)}\,.$$

This property is called Hermitian symmetry: the Fourier transform of a real-valued function has even symmetry in its real part and odd symmetry in its imaginary part. An obvious consequence is that when calculating the Fourier transform of a real-valued function, we need only consider positive values of ω since $F(\omega)$ determines $F(-\omega)$ by conjugacy.

Even and odd real-valued functions

Theorem

If $f(x)$ is an even real-valued function then its Fourier transform $F(\omega)$ is even and purely real-valued. If $f(x)$ is an odd real-valued function then its Fourier transform $F(\omega)$ is odd and purely imaginary.

Proof.

Suppose that f is even and real-valued. Then

$$\begin{aligned} F(\omega) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} f(x)e^{-i\omega x}dx \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} f(x)\left[\cos(\omega x) - i\sin(\omega x)\right]dx \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} f(x)\cos(\omega x)dx\,. \end{aligned}$$

Hence, F is real-valued and even (the imaginary part has vanished, and both $f(x)$ and $\cos(\omega x)$ are themselves even functions, which ensures $F(\omega)$ is an even function of ω). The second part follows similarly. □

Frequency shifting: multiplying $f(x)$ by e^{icx}

Theorem

For $c \in \mathbb{R}$ then

$$\mathcal{F}_{[e^{icx} f(x)]}(\omega) = \mathcal{F}_{[f]}(\omega - c)\,.$$

Proof.

$$\begin{aligned}
\mathcal{F}_{[e^{icx} f(x)]}(\omega) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{icx} f(x) e^{-i\omega x} dx \\
&= \frac{1}{2\pi} \int_{-\infty}^{\infty} f(x) e^{-i(\omega - c)x} dx \\
&= \mathcal{F}_{[f]}(\omega - c)\,.
\end{aligned}$$

□

There is a symmetry (sometimes called a "duality): a shift in $f(x)$ by b causes $\mathcal{F}_{[f]}(\omega)$ to be *multiplied* by $e^{i\omega b}$, whereas multiplying $f(x)$ by e^{icx} causes $\mathcal{F}_{[f]}(\omega)$ to be *shifted* by c.

Modulation property

Theorem

For $c \in \mathbb{R}$ then

$$\mathcal{F}_{[f(x)\cos(cx)]}(\omega) = \frac{\mathcal{F}_{[f]}(\omega - c) + \mathcal{F}_{[f]}(\omega + c)}{2}$$

$$\mathcal{F}_{[f(x)\sin(cx)]}(\omega) = \frac{\mathcal{F}_{[f]}(\omega - c) - \mathcal{F}_{[f]}(\omega + c)}{2i}\,.$$

Proof.

We have that $\cos(cx) = \frac{e^{icx}+e^{-icx}}{2}$ and $\sin(cx) = \frac{e^{icx}-e^{-icx}}{2i}$, so

$$\begin{aligned}
\mathcal{F}_{[f(x)\cos(cx)]}(\omega) &= \mathcal{F}_{\left[f(x)\frac{e^{icx}+e^{-icx}}{2}\right]}(\omega) \\
&= \frac{1}{2}\mathcal{F}_{[f(x)e^{icx}]}(\omega) + \frac{1}{2}\mathcal{F}_{[f(x)e^{-icx}]}(\omega) \\
&= \frac{\mathcal{F}_{[f]}(\omega - c) + \mathcal{F}_{[f]}(\omega + c)}{2}\,.
\end{aligned}$$

Similarly, for $\mathcal{F}_{[f(x)\sin(cx)]}(\omega)$. □

Derivatives

There are further useful properties relating to the Fourier transform of derivatives that we shall state here but omit further proofs.

Theorem

If function f has derivative f' then

$$\mathcal{F}_{[f']}(\omega) = i\omega \mathcal{F}_{[f]}(\omega)\,.$$

It follows by concatenation that for n^{th}-order derivatives $f^{(n)}$

$$\mathcal{F}_{[f^{(n)}]}(\omega) = (i\omega)^n \mathcal{F}_{[f]}(\omega)\,.$$

In Fourier terms, taking a derivative (of order n) is thus a kind of filtering operation: the Fourier transform of the original function is just multiplied by $(i\omega)^n$, which emphasizes the higher frequencies while discarding the lower frequencies. Taking derivatives amounts to high-pass filtering.

The notion of derivative can thus be generalized to non-integer order, $n \in \mathbb{R}$ instead of just $n \in \mathbb{N}$. In fields like fluid mechanics, it is sometimes useful to have the 0.5^{th} or 1.5^{th} derivative of a function, $f^{(0.5)}$ or $f^{(1.5)}$.

Application of the derivative property

In a remarkable way, the derivative property converts calculus problems (such as solving differential equations) into much easier algebra problems. Consider for example a 2^{nd}-order differential equation such as

$$af''(x) + bf'(x) + cf(x) = g(x)$$

where $g \neq 0$ is some known function or numerically sampled behaviour whose Fourier transform $G(\omega)$ is known or can be computed. Solving this common class of differential equation requires finding function(s) $f(x)$ for which the equation is satisfied. How can this be done?

By taking Fourier transforms of both sides of the differential equation and applying the derivative property, we immediately get a simple algebraic equation in terms of $G(\omega) = \mathcal{F}_{[g]}(\omega)$ and $F(\omega) = \mathcal{F}_{[f]}(\omega)$:

$$[a(i\omega)^2 + bi\omega + c]F(\omega) = G(\omega)$$

Now we can express the Fourier transform of our desired solution $f(x)$

$$F(\omega) = \frac{G(\omega)}{-a\omega^2 + bi\omega + c}$$

and wish that we could "invert" $F(\omega)$ to express $f(x)$!

Inverse Fourier transform

There is an inverse operation for recovering a function f given its Fourier transform $F(\omega) = \mathcal{F}_{[f]}(\omega)$, which takes the form

$$f(x) = \int_{-\infty}^{\infty} \mathcal{F}_{[f]}(\omega) e^{i\omega x} d\omega \,,$$

which you will recognize as the property of an orthonormal system in the space of continuous functions, using the complex exponentials $e^{i\omega x}$ as its basis elements.

Note that in the inverse Fourier transform we use $e^{+i\omega x}$, whereas in the forward Fourier transform we used $e^{-i\omega x}$.

Observe again the local-to-global property that $F(\omega)$ at any point ω affects the value of $f(x)$ at *every* point x; and also the global-to-local property that $f(x)$ at each point x is affected by the behaviour of $F(\omega)$ at *all* points ω.

Convolution

An important operation combining two functions to create a third one, with many applications (especially in signal and image processing, and pattern recognition), is convolution, defined as follows.

Definition (Convolution)

If f and g are two functions $\mathbb{R} \to \mathbb{C}$ then the convolution operation, denoted by an asterisk $f * g$, creating a third function, is given by

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x - y)g(y)dy$$

In words: the two functions are multiplied together under all possible amounts of relative shift, after one has been flipped $f(y) \to f(-y)$. The integral of their product creates a new function of x, the relative shift.

The convolution operation is commutative: $f * g = g * f$.

A nice animation demonstrating convolution may be found at:
`http://mathworld.wolfram.com/Convolution.html`

Fourier transforms and convolutions

The importance of Fourier transform techniques for signal processing rests, in part, on the fact that all "filtering" operations are convolutions, and even taking derivatives amounts really to a filtering or convolution operation. The following result shows that all such operations can be implemented merely by *multiplication* of functions in the Fourier domain, which is much simpler and faster.

Theorem (Convolution theorem)

For functions f and g having convolution $f * g$, then

$$\mathcal{F}_{[f*g]}(\omega) = 2\pi \mathcal{F}_{[f]}(\omega) \cdot \mathcal{F}_{[g]}(\omega)\,.$$

The convolution integral, whose definition explicitly required integrating the product of two functions for all possible relative shifts between them, to generate a new function in the variable of the amount of shift, is now seen to correspond to the much simpler operation of multiplying together both of their Fourier transforms. Then to obtain the convolution result, one need only take the inverse Fourier transform of this product.

Proof

We have that

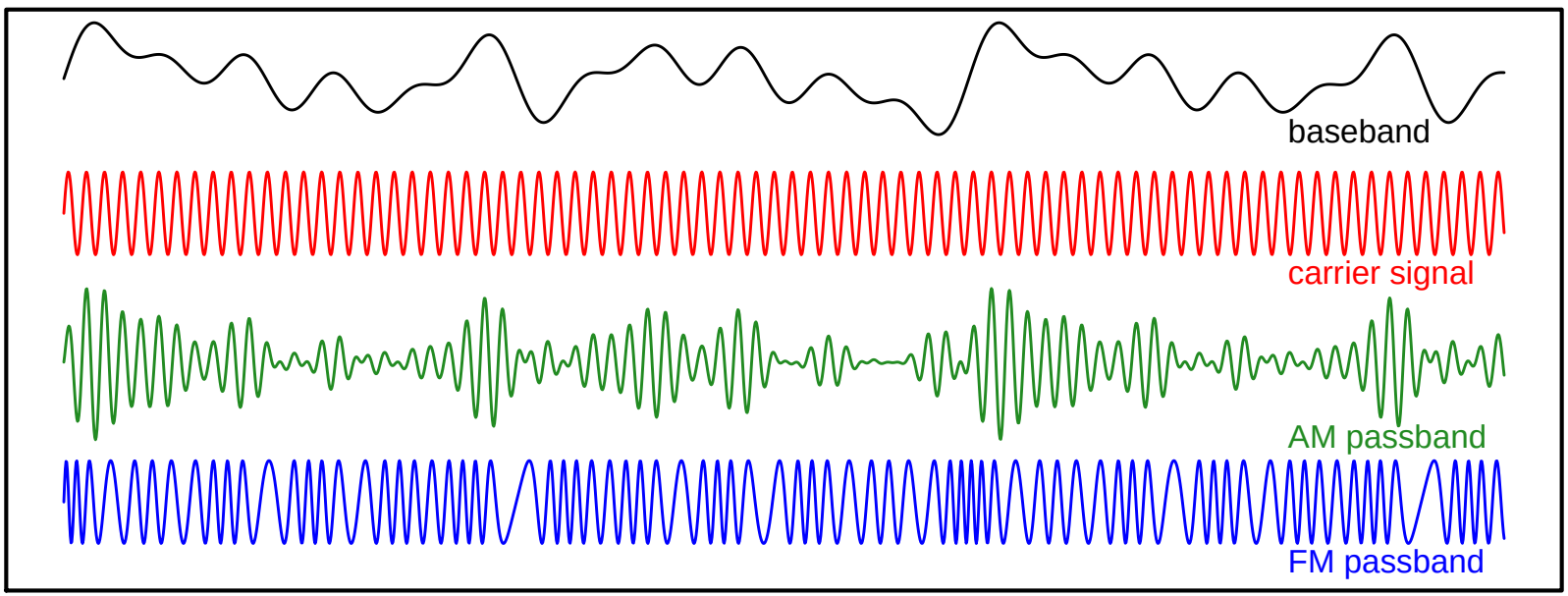
$$\begin{aligned}
\mathcal{F}_{[f*g]}(\omega) &= \frac{1}{2\pi}\int_{-\infty}^{\infty}(f*g)(x)e^{-i\omega x}dx \\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty}\left(\int_{-\infty}^{\infty}f(x-y)g(y)dy\right)e^{-i\omega x}dx \\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(x-y)e^{-i\omega(x-y)}g(y)e^{-i\omega y}dxdy \\
&= \int_{-\infty}^{\infty}\left(\frac{1}{2\pi}\int_{-\infty}^{\infty}f(x-y)e^{-i\omega(x-y)}dx\right)g(y)e^{-i\omega y}dy \\
&= \mathcal{F}_{[f]}(\omega)\int_{-\infty}^{\infty}g(y)e^{-i\omega y}dy \\
&= 2\pi\mathcal{F}_{[f]}(\omega)\cdot\mathcal{F}_{[g]}(\omega)\,.
\end{aligned}$$

□

Unsurprisingly, this duality applies also in the reverse direction: convolving two functions in the Fourier domain is equivalent to simply multiplying together their inverse Fourier transforms.

Communications schemes exploiting Fourier theorems

Encoding and transmission of information in communication channels involves many schemes for the modulation of some parameters of a carrier signal, which is typically just a sinewave of some high frequency. Some familiar examples include amplitude modulation (AM), frequency modulation (FM), and phase modulation (PM, or phase-shift keying). Upon reception, the modulated carrier (*passband*) must be demodulated to recover the encoded information (the *baseband*). One reason for such schemes is that many different channels can share a common medium, such as a band of the electromagnetic spectrum. Selecting the carrier frequency is how one "tunes" into a given channel or mobile signal.



Practical application of the modulation theorem

Conveying a message signal inside another signal that can be transmitted involves manipulations in the Fourier domain. Here we consider the basis of amplitude modulation, either SSB or DSB (single or double sideband).

Let $f(t)$ be the baseband message we wish to transmit; it might be the audio signal of somebody speaking. Let $F(\omega)$ be its Fourier transform:

$$F(\omega) = \mathcal{F}_{[f(t)]} = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt$$

But what happens to $F(\omega)$ if we first multiply (modulate) $f(t)$ by a complex exponential carrier signal e^{ict} of frequency c? We saw that:

$$\begin{aligned} \mathcal{F}_{[e^{ict} f(t)]} &= \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{ict} f(t) e^{-i\omega t} dt \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} f(t) e^{-i(\omega - c)t} dt \\ &= F(\omega - c)\,. \end{aligned}$$

Recovering the baseband signal after modulation

We have seen that modulating a signal using a complex exponential of frequency c simply shifts its Fourier spectrum up by that frequency c, so we end up with $F(\omega - c)$ instead of $F(\omega)$. This enables the baseband signal $f(t)$ to be encoded into its own nominated slice of a shared broad communication spectrum, for transmission.

As c ranges into MHz or GHz, upon reception of this passband we must recover the original audio baseband signal $f(t)$ by shifting the spectrum back down again by the same amount, c. Clearly such demodulation can be achieved simply by multiplying the received signal by e^{-ict}, which is what a tuner does (*i.e.* the channel selector "dial" corresponds to c):

$$\left[e^{ict} f(t)\right] e^{-ict} = f(t)\,.$$

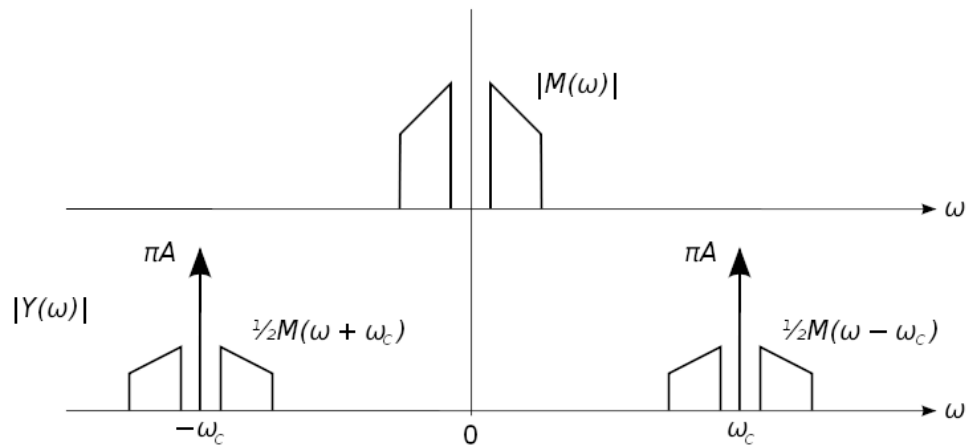
But note both these operations of multiplying $f(t)$ by complex-valued functions e^{ict} and e^{-ict} are complicated: two modulating sinewaves are actually required, $\cos(ct)$ and $\sin(ct)$, in precise quadrature phase, and two circuits for multiplying them are needed. Both parts of the resulting complex-valued signal must be transmitted and detected.

Double sideband amplitude modulation

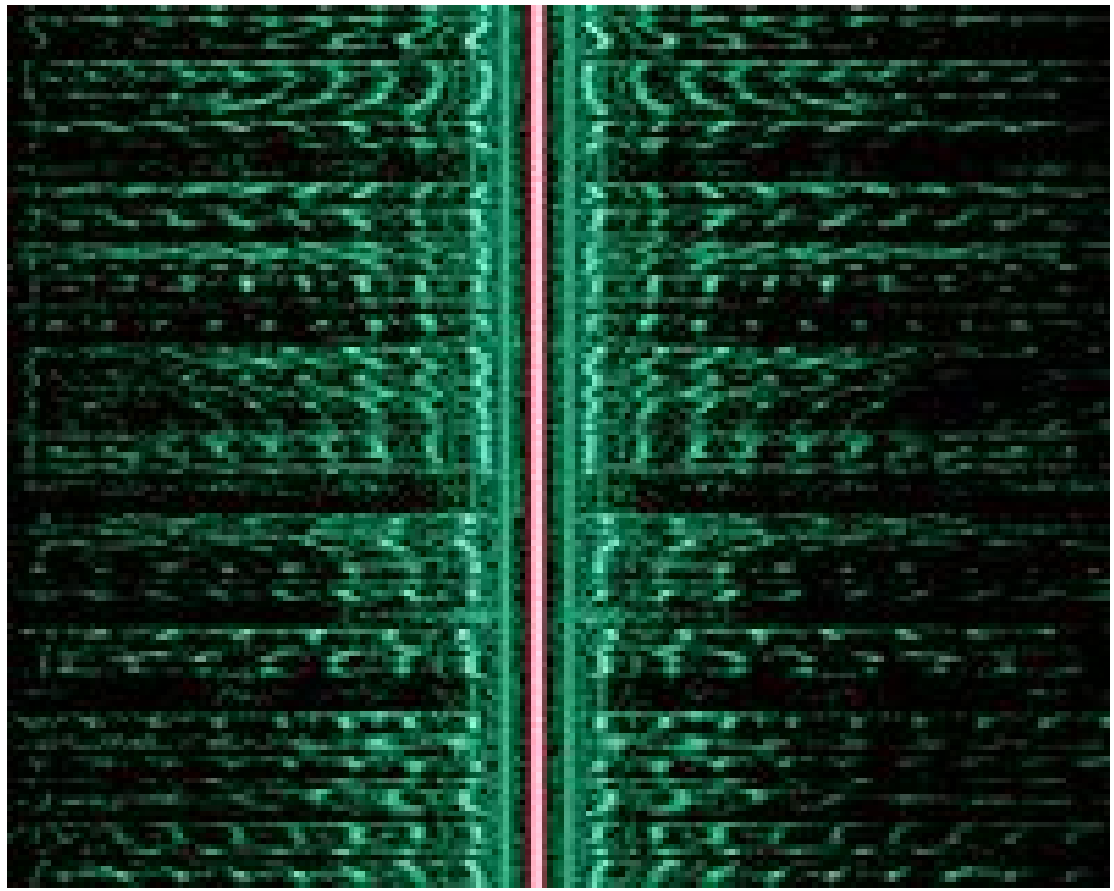
To simplify the process and circuitry, but doubling the bandwidth needed, an alternative is just to multiply $f(t)$ by one (real-valued) cosine wave. Since $\cos(ct) = \dfrac{e^{ict} + e^{-ict}}{2}$, by applying the Modulation theorem twice (adding the results) we see that if $F(\omega)$ is the Fourier transform of the original baseband $f(t)$, then after multiplying $f(t)$ by $\cos(ct)$, the new spectrum as we noted earlier becomes:

$$\mathcal{F}_{[\cos(ct)f(t)]} = \frac{F(\omega - c) + F(\omega + c)}{2}.$$

Thus we end up with a passband whose spectrum consists of two copies of the baseband, shifted by both positive, and negative, frequencies c. One pair of the duplicated lobes can be suppressed (SSB modulation).



An actual passband broadcast using amplitude modulation



Spectrogram (frequency spectrum versus time) of an AM broadcast. Here, time is the vertical axis; frequency is the horizontal axis. The pink band of energy is the carrier itself (which can be suppressed). Sidebands are displayed in green, on either side of the carrier frequency. Speech formants are visible.

Frequency Division Multiple Access (FDMA)

Radio waves propagate well through the atmosphere in a frequency range extending into GigaHertz, with specific bands allocated by governments for commercial broadcasting, mobile broadband, etc. Narrow bands of spectrum are auctioned for *tens of billions* of pounds, *e.g.* in the UHF band around 1 GigaHertz (0.3 to 3.0 GHz) for mobile phone operators.

A human audio signal $f(t)$ occupies only about 1 KHz, so its spectrum $F(\omega)$ has a bandwidth that is tiny relative to the carrier frequency; thus a great many different mobile channels can be allocated by assignments of frequencies c. Frequency Division Multiple Access (FDMA) refers to one such regulated access protocol. (Other access protocols are available.)

Many alternative coding schemes exist, regulated by government agencies and by standards. Phase modulation (PM) illustrates another method for transmitting information. One natural application is in colour television, since perceived colour has a cyclical topology (the "colour wheel"), just like the phase ϕ of a carrier signal. PM resembles FM, because frequency ω is the time-derivative of phase: $\omega = \dfrac{d\phi}{dt}$. The detection of modulated functions as such can be decoded into symbols, as readily as colour.

10. Quantised degrees-of-freedom in a continuous signal

- *Nyquist sampling theorem; strict bandlimiting; aliasing and its prevention.*
- *Logan's theorem and the richness of zero-crossings in one-octave signals.*
- *The information diagram: how many independent quanta can it contain?*

Several independent results all illustrate the (perhaps surprising) fact that strictly bandlimiting a continuous function or signal causes it to have a finite, countable number of degrees-of-freedom. All the data contained in the continuous function can be regarded as quantised, into countable *quanta* of information, rather than having the density of real numbers.

▷ Nyquist's sampling theorem asserts that if a signal is strictly bandlimited to some highest frequency W, then simply sampling its values at a rate $2W$ specifies it completely everywhere, even *between* the sample points. Thus over some time interval T, it is fully determined by $2WT$ numbers.

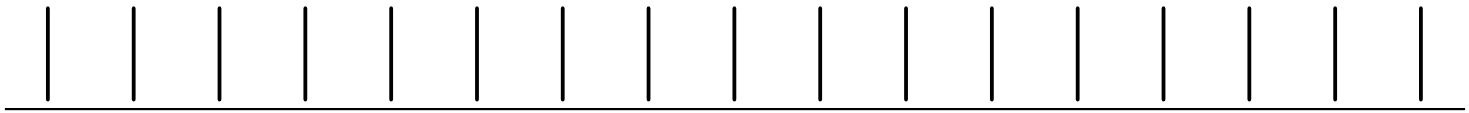
▷ Logan's theorem asserts that if a function is bandlimited to one octave, then merely listing its zero-crossings fully determines it.

▷ Gabor's information diagram has a quantal structure, with a minimal area (bandwidth $\times$ time) dictated by an Uncertainty Principle and occupied by Gaussian-attenuated complex exponentials: Gabor wavelets.

Nyquist's sampling theorem: ideal sampling function

We define a sampling function $\mathrm{comb}(t) = \delta_X(t)$ as an endless sequence of regularly spaced tines, separated by some sampling interval X:

$\delta_X(t)$:



$\ldots$

Each "tine" is actually a Dirac δ-function, which can be regarded as the limit of a Gaussian whose width shrinks to 0. Multiplying a signal with one $\delta(t)$ samples its value at $t = 0$. The portrayed sequence of tines spaced by X is a sum of shifted δ-functions, making a sampling comb:

$$\delta_X(t) = \sum_n \delta(t - nX)\,.$$

The sampling function $\delta_X(t)$ is self-Fourier: its Fourier transform $\Delta_X(\omega)$ is *also* a $\mathrm{comb}(\omega)$ function, but with reciprocal interval $1/X$ in frequency:

$$\mathcal{FT}\{\delta_X(t)\} = \Delta_X(\omega) = \frac{1}{X}\sum_m \delta(\omega X - 2\pi m)\,.$$

$\Delta_X(\omega)$:

$\ldots$

Properties of Dirac δ-functions

Conceptually,

$$\delta(t) = \begin{cases} \text{“}\infty\text{”} & t = 0 \\ 0 & t \neq 0 \end{cases}$$

and $\delta(t)$ contains unit area:

$$\int_{-\infty}^{\infty} \delta(t)dt = 1\,.$$

Multiplying any function $f(t)$ with a displaced δ-function $\delta(t-c)$ and integrating the product just picks out the value of $f(t)$ at $t = c$:

$$\int_{-\infty}^{\infty} f(t)\delta(t-c)dt = f(c)$$

which implies also that convolving any function with a δ-function simply reproduces the original function.

Having defined the “comb” sampling function as a sequence of displaced δ-functions with some sampling interval X, we can now understand the act of sampling a signal as just multiplying it with our comb(t) function.

Strict bandlimiting, and spectral consequence of sampling

We now consider only signals $f(t)$ that have been strictly bandlimited, with some upper bound W on frequency. Thus their Fourier transforms $F(\omega)$ are 0 for all frequencies ω larger in absolute value than W:

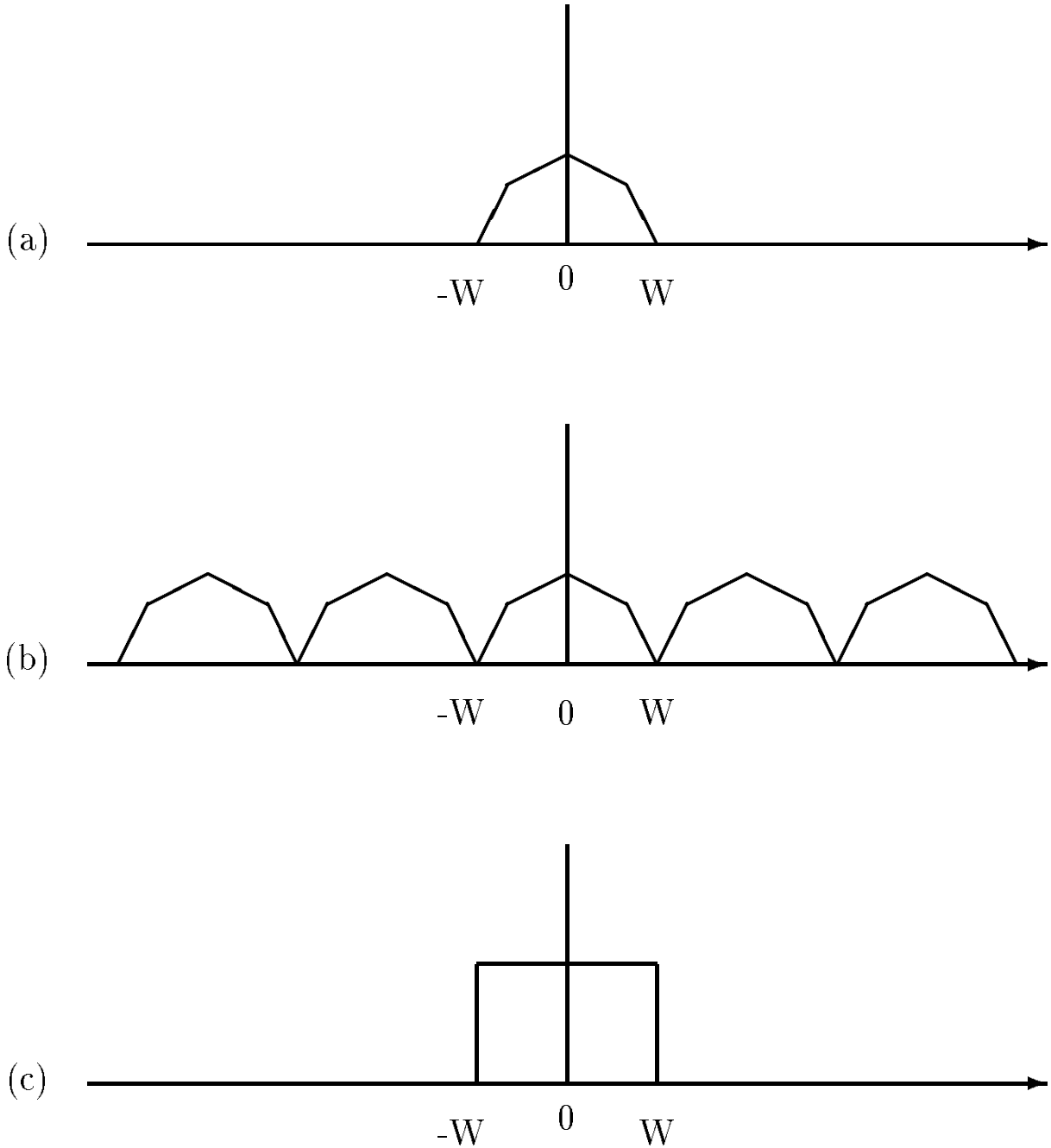
$$F(\omega) = 0 \quad \text{for} \quad |\,\omega\,| > W.$$

If we have a signal $f(t)$ for which this is not already true, we make it so, by passing the signal through a strict lowpass filter, set for W. Its Fourier transform $F(\omega)$ becomes truncated, similar to trace (a) on the next slide.

Now if we sample $f(t)$ by multiplying it with our comb function $\delta_X(t)$, and use a sampling rate of at least $2W$ so that the sampling interval $X \leq 1/(2W)$, then we know from the Convolution theorem that the resulting sequence of samples will have a Fourier transform that is the convolution of $F(\omega)$ with $\Delta_X(\omega)$, the Fourier transform of $\delta_X(t)$.

Since convolution with a single δ-function reproduces function $F(\omega)$, and since $\Delta_X(\omega)$ is a sum of many shifted δ-functions $\sum_m \delta(\omega X - 2\pi m)$, our $F(\omega)$ becomes reproduced at every "tine" of $\Delta_X(\omega)$, as seen in trace (b).

(Nyquist's sampling theorem, continued)



Recovering the signal, even between its sampled points!

It is clear from trace (b) that the original spectrum $F(\omega)$ has completely survived the sampling process; but it just has been joined by multiple additional copies of itself. As those did not overlap and superimpose, we need only eliminate them (by another strict lowpass filtering action) in order to recover perfectly the Fourier transform $F(\omega)$ of our original (strictly bandlimited) signal, and thereby, that signal $f(t)$ itself. Visually, on the previous slide, multiplying trace(b) with trace(c) recovers trace(a).

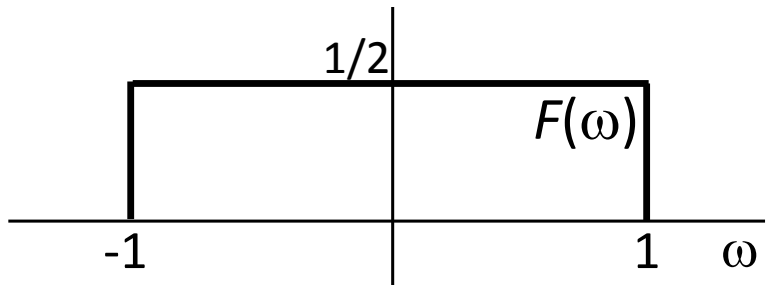
Why did those copied spectral lobes not overlap and superimpose?

Answer: we now see the critical importance of sampling the bandlimited signal $f(t)$ at a sampling rate, $2W$, at least twice as rapid as its highest frequency component W. Given the reciprocal spacing of the "tines" in $\delta_X(t)$ (namely $X \leq 1/(2W)$) and those in $\Delta_X(\omega)$ (namely $1/X \geq 2W$), it is clear that this Nyquist sampling rate $\geq 2W$ is the key constraint that ensures complete reconstructability of the signal $f(t)$ from just its discrete samples, even between the points where it was sampled.

But how can such complete reconstruction be achieved?

Ideal lowpass filtering: convolution with sinc(x) function

An ideal lowpass filter can be described as a zero-centred pulse function $F(\omega)$ in the frequency domain ω: it removes all frequencies higher than some cut-off frequency W. For simplicity we will initially set $W = \pm 1$, and give the pulse function unit area: $F(\omega) = 1/2$ for $\omega \in [-1, +1]$, and $F(\omega) = 0$ for $\mid \omega \mid > 1$:



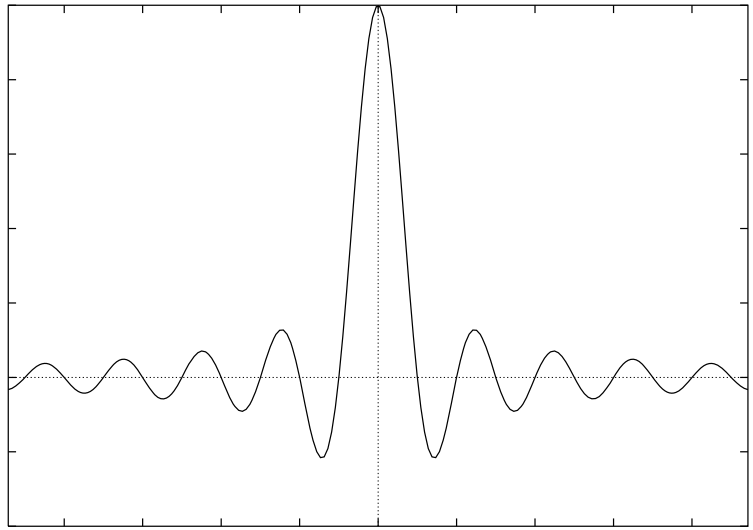
The inverse Fourier transform of this $F(\omega)$ lowpass filter is $f(x)$:

$$f(x) = \int_{-\infty}^{+\infty} F(\omega) e^{i\omega x} d\omega = \frac{1}{2} \int_{-1}^{+1} e^{i\omega x} d\omega$$

$$= \frac{1}{2} \left[\frac{e^{i\omega x}}{ix} \right]_{\omega=-1}^{\omega=+1} = \frac{e^{ix} - e^{-ix}}{2ix} = \frac{\sin(x)}{x} \ .$$

(At $x = 0$, $f(x) = \frac{1}{2} \int_{-1}^{+1} d\omega = 1$.) This wiggly function is called sinc.

The sinc(x) function for recovering a signal from samples



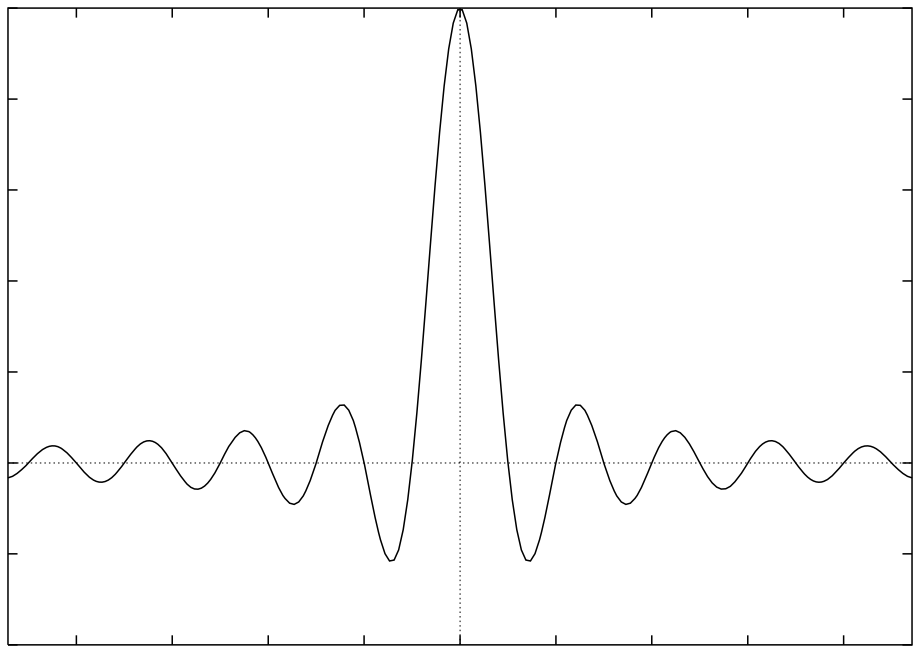
The zero-crossings are at $x = n\pi$ for all ($\pm$) integers n *except* $n = 0$.

In the more general case when we may wish to remove from a signal all frequencies higher than W in Hertz, so the lowpass filter has cut-off at $\pm W$, the corresponding time-domain sinc(t) function is parameterised by that frequency W, and it wiggles faster the higher W is:

$$\frac{\sin(2\pi Wt)}{2\pi Wt}$$

(Nyquist's sampling theorem, continued)

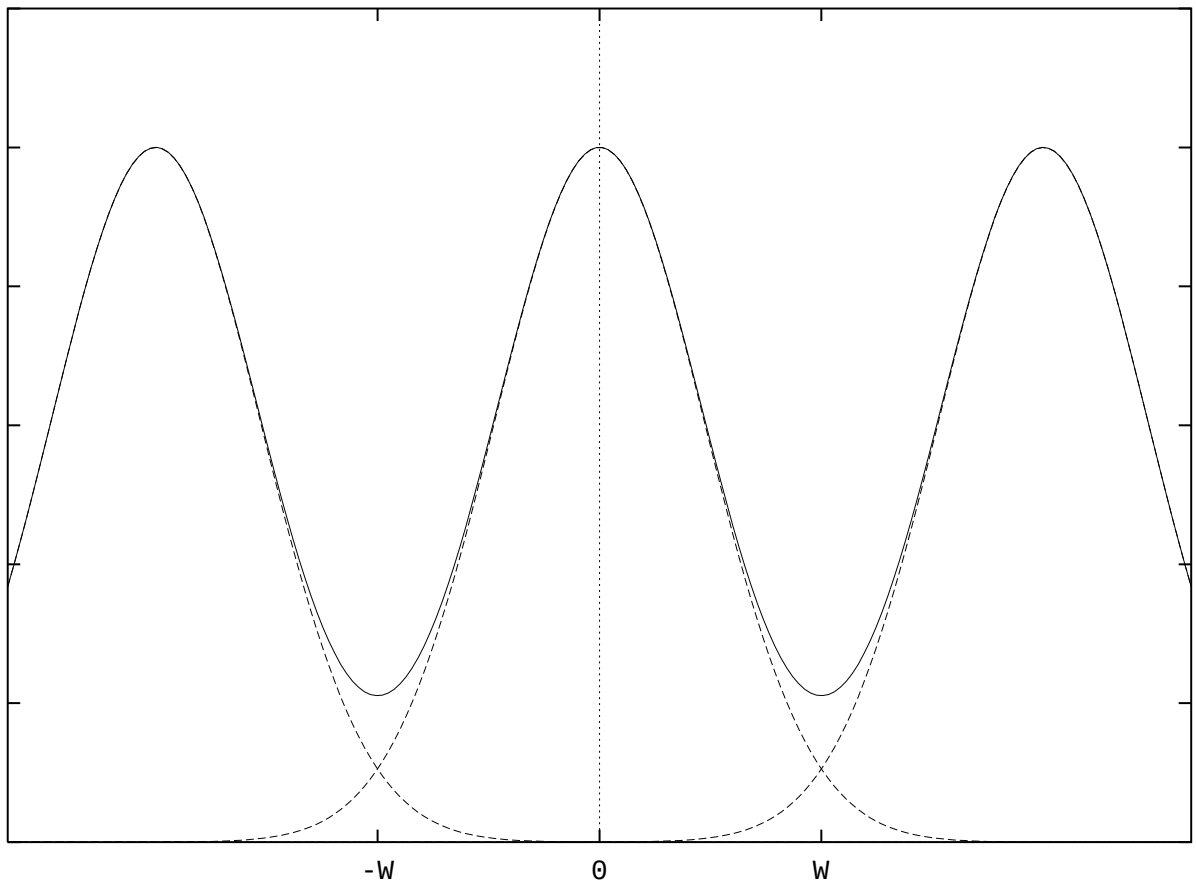
Again, we know from the Convolution theorem that strict lowpass filtering of the sequence $\delta_X(t)f(t)$ resulting from the sampling process equates to convolving this sample sequence by the inverse Fourier transform of that ideal lowpass filter. We saw that this is the sinc function: $\frac{\sin(2\pi Wt)}{2\pi Wt}$



Effectively, each sampled value get replaced by one of these, scaled in its (signed) amplitude by the sample value. These wiggly functions "fill in" all the space between the sample points, giving us back a continuous function $f(t)$; indeed exactly the one we started with before sampling it.

Aliasing

Failure to sample at a rate $\geq 2W$ causes aliasing: the added copies of the spectrum $F(\omega)$ overlap each other, because the tines in the frequency domain are now too close together. The extra lobes of $F(\omega)$ which they generate in the sampling process become superimposed, and now they can no longer be separated from each other by strict lowpass filtering.



(Nyquist's sampling theorem, con't): Effects of aliasing

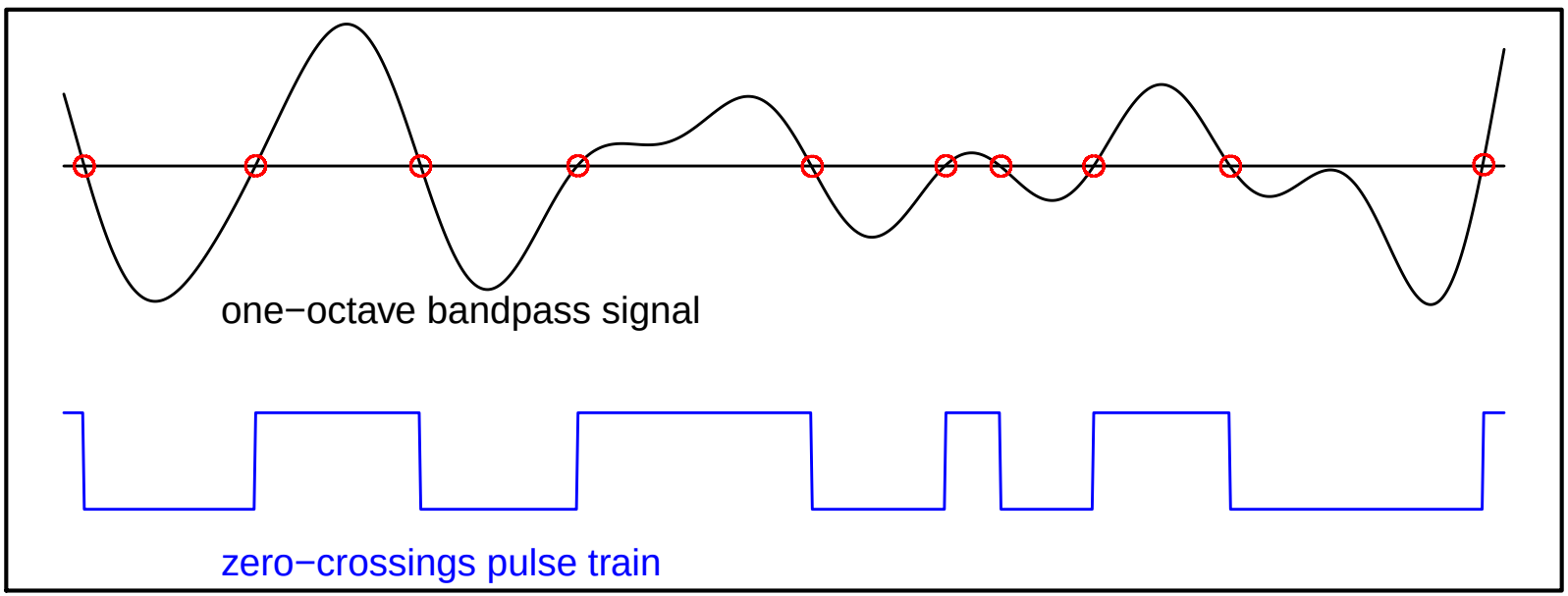
This is the reason why "old western" movies with stagecoaches dashing away often seem to have their cartwheels rotating backwards. The film frame rate was only about 16 Hz, and in the interval between frames, each spoke in a cartwheel can be interpreted as moving backwards by a small amount, rather than forwards by the actual amount. Subtracting a small phase is equivalent to adding a larger phase to the cartwheel angle.


In terms of the previous slide showing superimposed spectral lobes after sampling at too low a frequency like 16 Hz $< 2W$, effectively the higher frequencies get "folded back" into the lower frequencies, producing an apparent, slow, reverse motion of the cartwheels.

Logan's theorem: reconstruction from zero-crossings alone

We saw in Nyquist's sampling theorem that strictly bandlimiting a signal (in that case to W) means that all the information it contains can be represented by a countable number of packets, or quanta of information (namely $2WT$ quanta over some time interval T).

Another result in the same spirit is Logan's theorem, which says that if a signal is bandpass limited to one-octave, so that the highest frequency component W_H in it is no higher than twice W_L its lowest frequency component, $W_H \leq 2W_L$, then merely listing the zero-crossings of this signal suffice for exact reconstruction of the signal (up to a scale factor in amplitude), even in the continuous spaces between those zero-crossings.



Does Logan's theorem explain how cartoons work?

It has been suggested that Logan's theorem might explain how cartoons work: cartoon sketches are highly impoverished images, with mere edges being drawn, and no real resemblance to the data contents of a picture. Perhaps human vision imparts so much richness to edges because they are the zero-crossings in bandpass filtered images, and retinal encoding is usually modelled in such terms (albeit with > 1.3 octave bandwidth).

The question can be posed in more abstract form, too. "Attneave's cat" is even more impoverished than Margaret Thatcher's caricature, because it consists only of some vertices joined by straight lines (no curvature). Yet like the former PM's sketch, it is instantly recognised as intended.

Limitations to Logan's theorem

After Logan proved his remarkable theorem at Bell Labs in 1977, there was enthusiasm to exploit it for extreme compression of speech signals. The zero-crossings are where the MSB (most significant bit) changes, as the waveform is bandpass; and indeed a pulse train of "1-bit speech" is remarkably intelligible. But the distortion is audible, and objectionable.

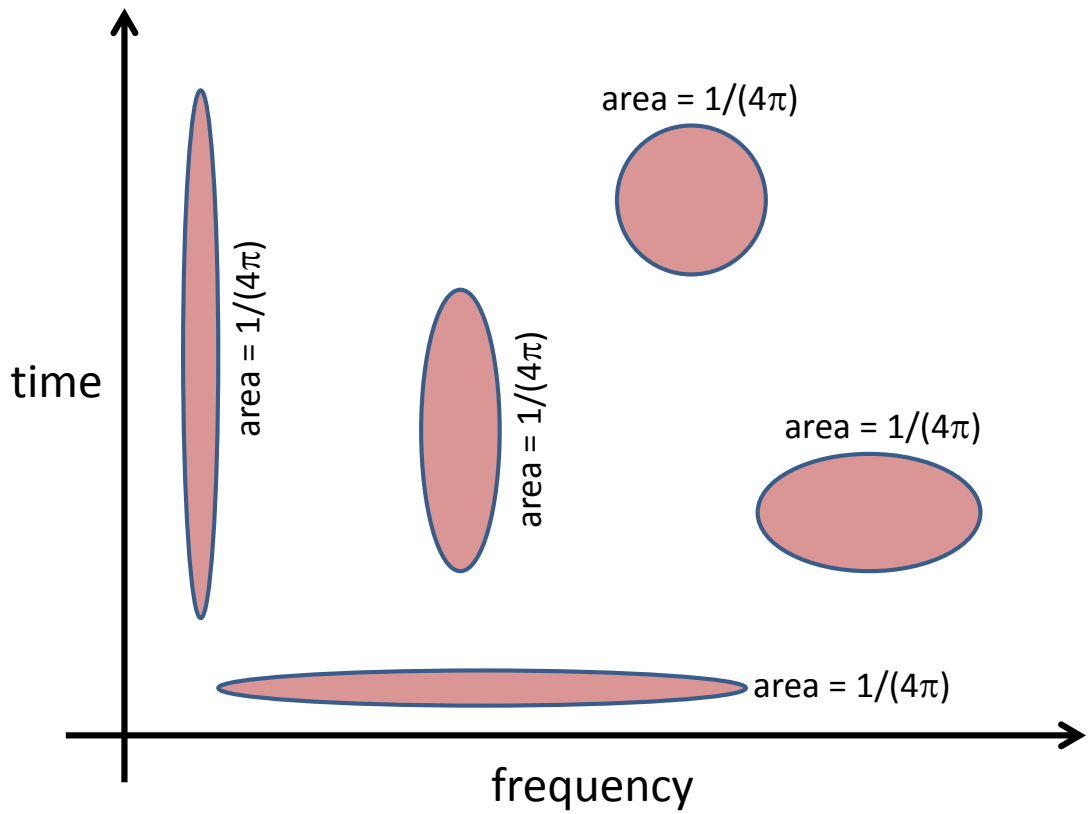
Clearly signals that are purely positive amplitude-modulations of a carrier $c(t)$, of the form $f(t) = [1 + a(t)]c(t)$ where $[1 + a(t)] > 0$, will encode nothing about $a(t)$ in the zero-crossings of $f(t)$: they will remain always just the zero-crossings of the carrier $c(t)$. Thus, "pure AM" signals are excluded from Logan's theorem by a condition that the signal must have no complex zeroes in common with its Hilbert transform.

Another limitation emerged in efforts to extend Logan's theorem to computer vision. The zero-crossings of bandpass filtered *images* are not discrete and countable, as for a 1D signal, but form continuous "snakes."

Finally, numerical algorithms for actually reconstructing 1D signals from their zero-crossings are unstable: tiny changes in the position of an extracted zero-crossing have divergent effects on the reconstruction.

Quantal structure of Gabor's "information diagram"

We will see that a fundamental uncertainty principle limits simultaneous localisability of a signal in time and frequency. The shorter its duration, the broader its bandwidth becomes. The narrower its bandwidth, the longer it must persist in time. Thus the information diagram whose axes are time and frequency has a profoundly quantal structure: no function can occupy an "area" smaller than some irreducible quantum, $1/(4\pi)$. The blobs in this diagram are called time-frequency atoms.



11. Gabor-Heisenberg-Weyl uncertainty principle; "logons"

We define the "effective width" (Δx) of a (complex-valued) function $f(x)$ in terms of its normalized variance, or normalized second-moment:

$$(\Delta x)^2 = \frac{\int_{-\infty}^{+\infty} f(x) f^*(x) (x-\mu)^2 dx}{\int_{-\infty}^{+\infty} f(x) f^*(x) dx}$$

where the purpose of the denominator is to normalise the amplitude or power in $f(x)$ so that we really just measure its width, and where μ is the mean value, or normalized first-moment, of the function $\|f(x)\|$:

$$\mu = \frac{\int_{-\infty}^{+\infty} x f(x) f^*(x) dx}{\int_{-\infty}^{+\infty} f(x) f^*(x) dx} \, .$$

Then, if we perform the same operations to measure the effective width $(\Delta\omega)$ of the Fourier transform $F(\omega) = \mathcal{FT}\{f(x)\}$ of the function $f(x)$:

(Uncertainty principle, continued)

$$(\Delta\omega)^2 = \frac{\int_{-\infty}^{+\infty} F(\omega)F^*(\omega)(\omega-\nu)^2 d\omega}{\int_{-\infty}^{+\infty} F(\omega)F^*(\omega)d\omega}$$

where ν is the mean value, or normalized first-moment, of $\|F(\omega)\|$:

$$\nu = \frac{\int_{-\infty}^{+\infty} \omega F(\omega)F^*(\omega)d\omega}{\int_{-\infty}^{+\infty} F(\omega)F^*(\omega)d\omega}$$

then it can be proven (by Schwartz Inequality arguments) that there exists a fundamental lower bound on the product of these two "spreads," *regardless* of the function $f(x)$:

$$\boxed{(\Delta x)(\Delta\omega) \geq \frac{1}{4\pi}}$$

This is the famous Gabor-Heisenberg-Weyl Uncertainty Principle.

(Uncertainty principle, continued)

Mathematically this is exactly equivalent to the uncertainty principle in quantum physics, where (x) would be interpreted as the position of an electron or other particle, and (ω) would be interpreted as its momentum or deBroglie wavelength^{-1} (because of "wave-particle duality").

$$\boxed{(\Delta x)(\Delta \omega) \geq \frac{1}{4\pi}}$$

We now see that this irreducible joint uncertainty is not just a law of nature, but it is a property of all functions and their Fourier transforms. It is thus another respect in which the information in continuous signals is quantised, since there is a limit to how sharply resolved they can be in the Information Diagram (*e.g.*, their joint resolution in time and frequency).

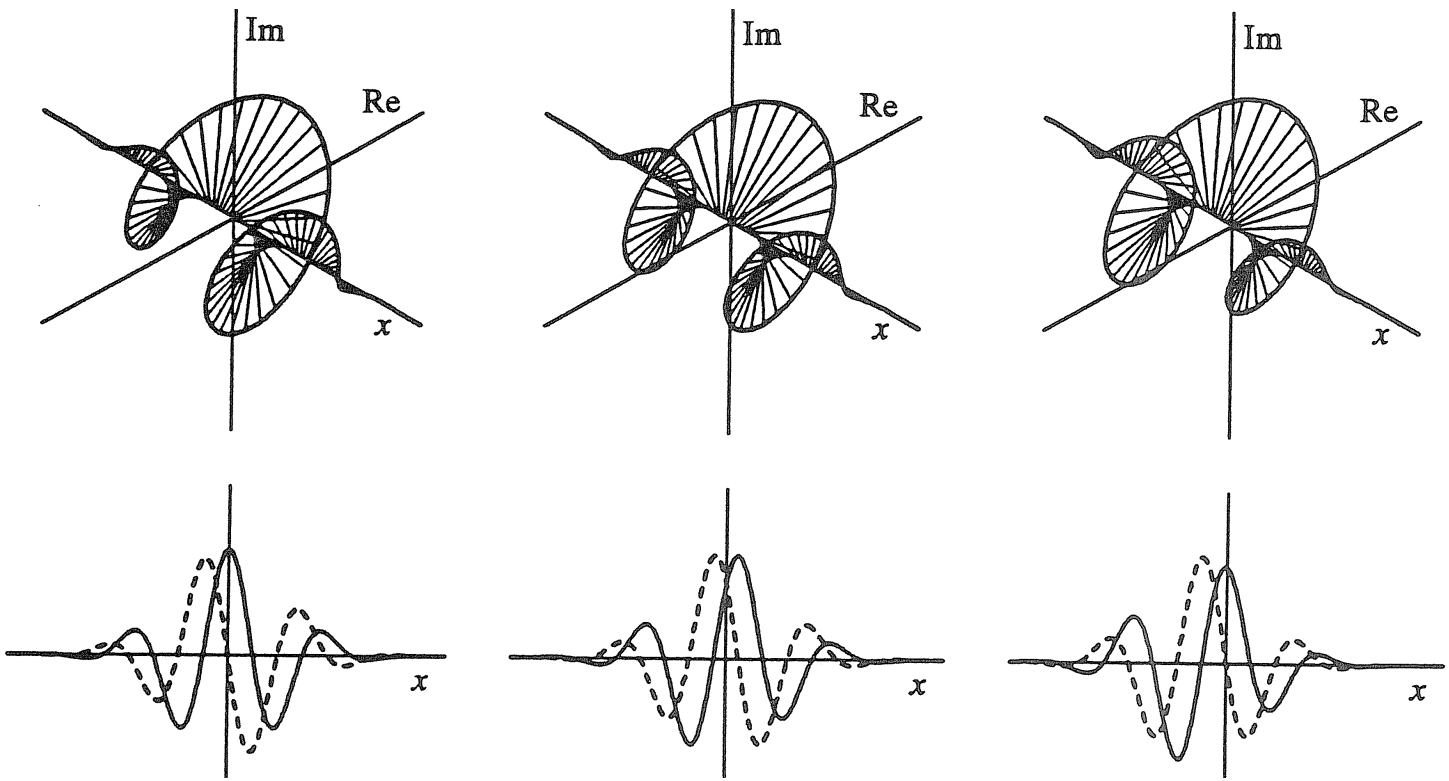
Dennis Gabor named such minimal areas "logons" from the Greek word for information, or order: lōgos. Their total number within a region of the Information Diagram specifies the maximum number of independent degrees-of-freedom available to a signal. Gabor went on to discover what family of functions actually achieve this minimal joint uncertainty.

Gabor wavelets, or "logons"

The unique family of functions that actually achieve the lower bound in joint uncertainty are the complex exponentials multiplied by Gaussians. Originally "logons", today these are usually called Gabor wavelets:

$$f(x) = e^{-(x-x_0)^2/\alpha^2} e^{i\omega_0(x-x_0)}$$

localised at "epoch" x_0, modulated with frequency ω_0, and having a size or spread constant α. As phasors they are helical functions of x, and the lower trace plots the real and imaginary parts for different epochs x_0.



(Gabor wavelets, continued)

Such wavelets are self-Fourier: their Fourier transforms $F(\omega)$ have the same functional form, but with the parameters just interchanged:

$$F(\omega) = e^{-(\omega-\omega_0)^2\alpha^2} e^{-ix_0(\omega-\omega_0)}$$

Note that for a wavelet whose epoch is $x_0 = 0$, its Fourier transform is just a Gaussian located at the modulation frequency ω_0, and whose size is $1/\alpha$, the reciprocal of the centred wavelet's size or spread constant.

Because such wavelets have the greatest possible simultaneous resolution in time and frequency, Gabor proposed using them as an expansion basis to represent signals. Unfortunately, because these wavelets are mutually non-orthogonal, it is difficult to compute the expansion coefficients.

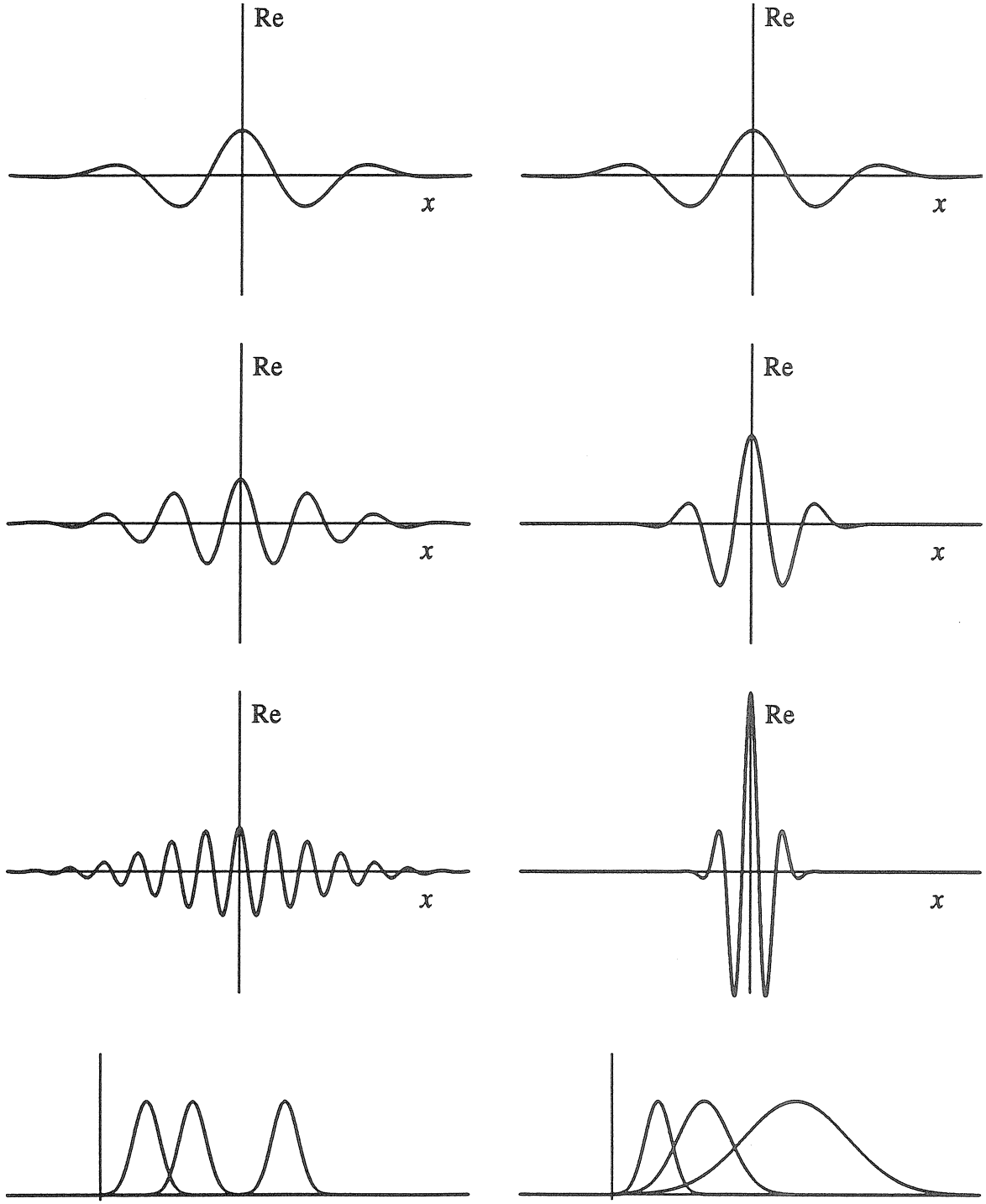
In the wavelets plotted (real parts only) in left column of the next slide, the Gaussian is always the same size α, while the frequency ω_0 increases. This reduces the wavelets' bandwidth in octave terms. It is more usual to use a self-similar scaling rule (illustrated in the right column), in which α and ω_0 are inversely proportional, so the wavelets are all dilated copies of a single mother wavelet.

(Gabor wavelets, con't)

Left column: wavelets sharing a constant Gaussian size α but with increasing ω_0.

Right column: self-similar wavelets with $\alpha^{-1} \sim \omega_0$.

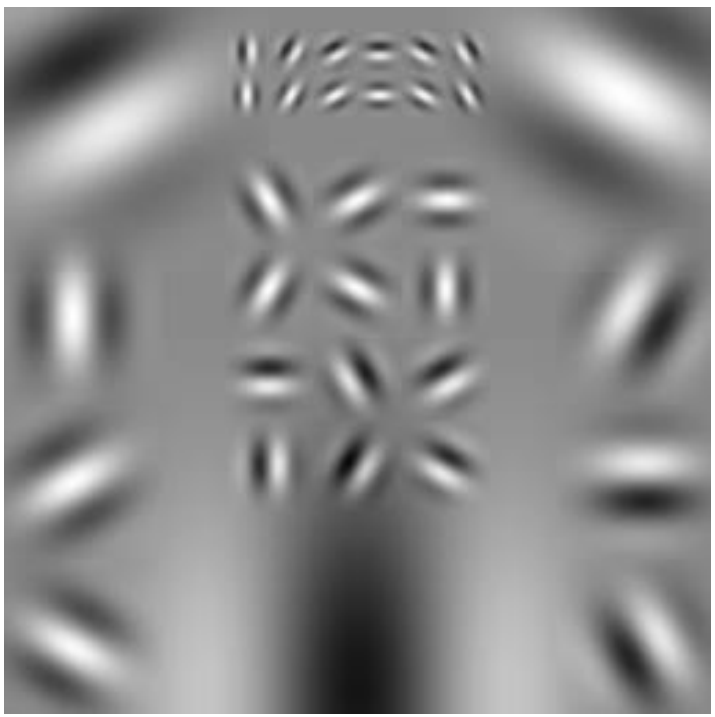
Bottom row: all of their Fourier transforms.



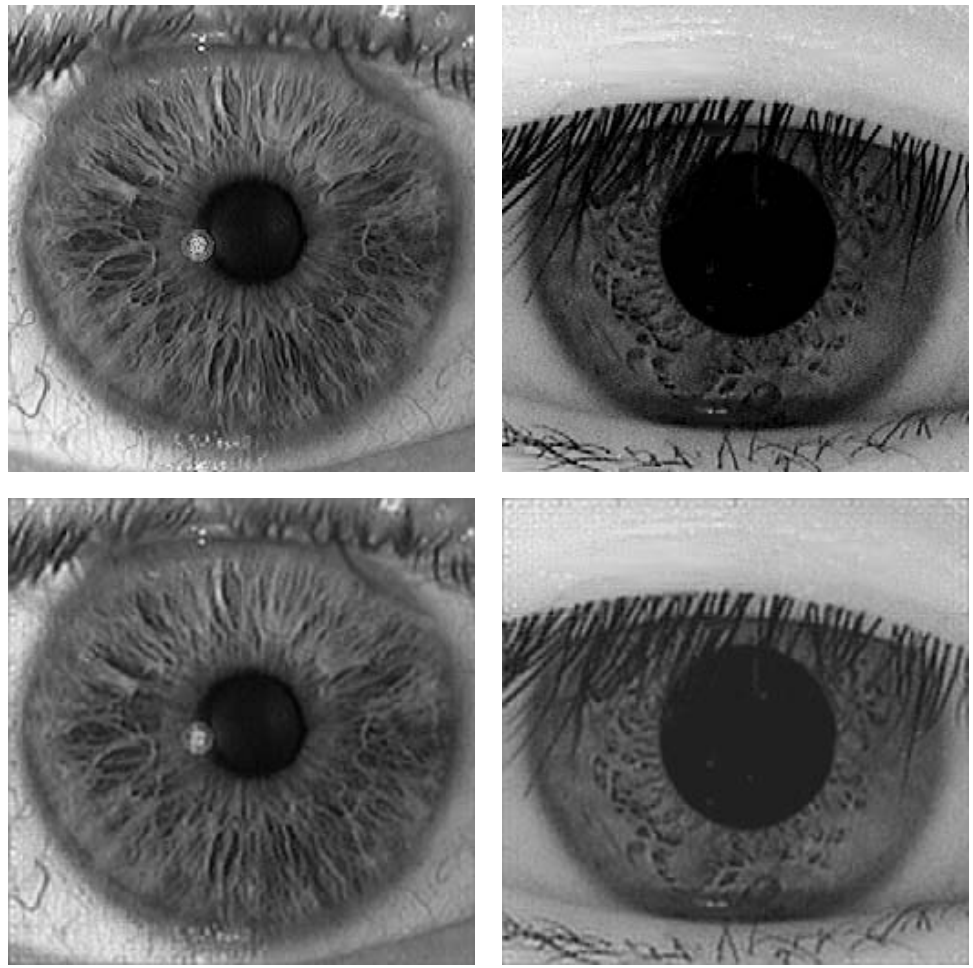
2D Gabor wavelets as encoders in computer vision

2D Gabor wavelets (defined as a complex exponential plane-wave times a 2D Gaussian windowing function) are extensively used in computer vision.

As multi-scale image encoders, and as pattern detectors, they form a complete basis that can extract image structure with a vocabulary of: location, scale, spatial frequency, orientation, and phase (or symmetry). This collage shows a 4-octave ensemble of such wavelets, differing in size (or spatial frequency) by factors of two, having five sizes, six orientations, and two quadrature phases (even/odd), over a lattice of spatial positions.

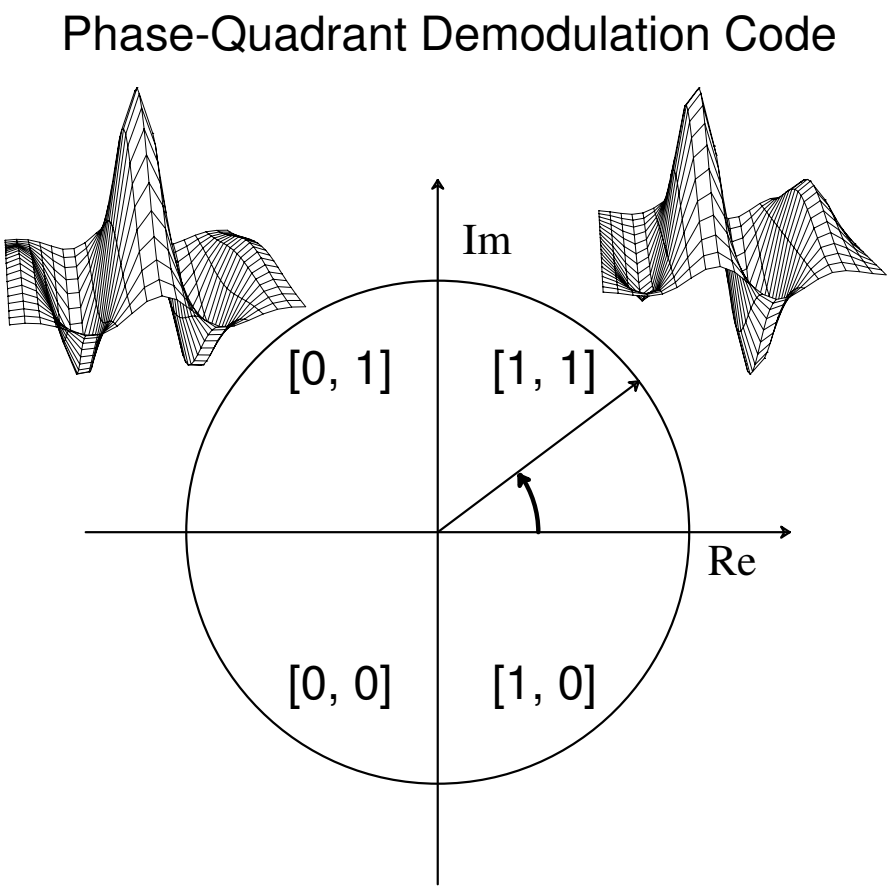


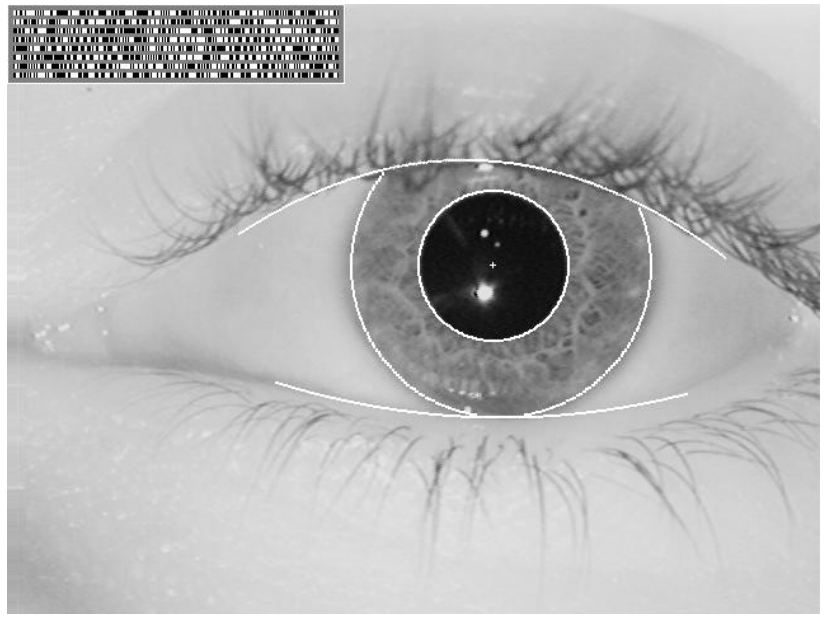
Complex natural patterns are very well represented in such terms. Phase information is especially useful to encode for pattern recognition *e.g.* iris. The upper panels show two iris images (acquired in near-infrared light); caucasian iris on the left, and oriental iris on the right.

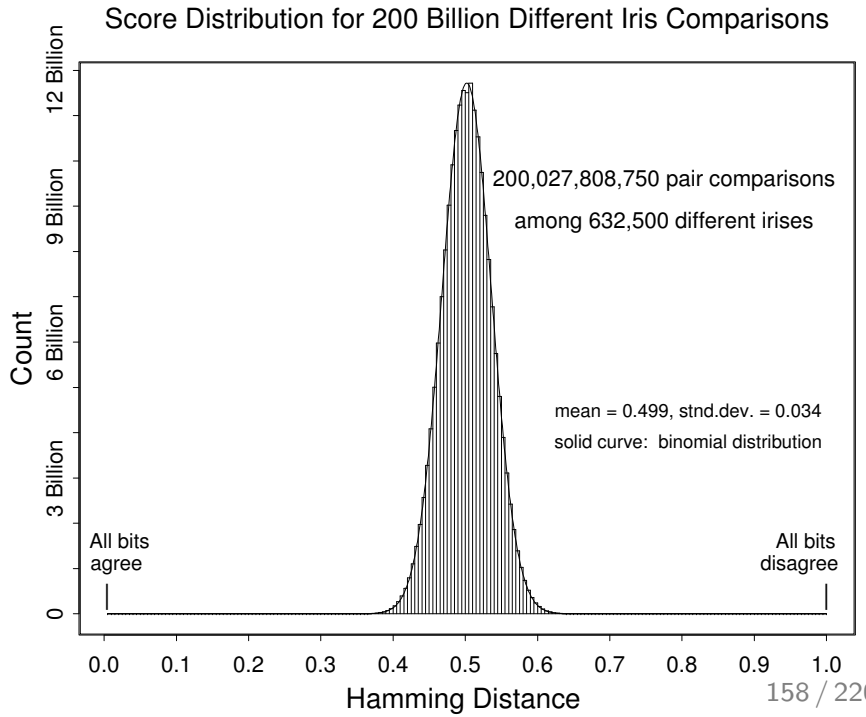


The lower panels show the images reconstructed just from combinations of the 2D Gabor wavelets spanning 4 octaves seen in the previous slide.

Gabor wavelets are the basis for Iris Recognition systems







$$h_{Re} = 1 \text{ if } \mathrm{Re} \int_\rho \int_\phi e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} I(\rho,\phi)\rho d\rho d\phi \geq 0$$

$$h_{Re} = 0 \text{ if } \mathrm{Re} \int_\rho \int_\phi e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} I(\rho,\phi)\rho d\rho d\phi < 0$$

$$h_{Im} = 1 \text{ if } \mathrm{Im} \int_\rho \int_\phi e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} I(\rho,\phi)\rho d\rho d\phi \geq 0$$

$$h_{Im} = 0 \text{ if } \mathrm{Im} \int_\rho \int_\phi e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} I(\rho,\phi)\rho d\rho d\phi < 0$$

Gabor wavelets are widely used for information encoding

At many airports worldwide, the **IRIS** system (Iris Recognition Immigration System) allows registered travellers to cross borders without having to present their passports, or make any other claim of identity. They just look at an iris camera, and (if they are already enrolled), the border barrier opens within seconds. Similar systems are in place for other applications. In a national ID system, the Gov't of India has enrolled the iris patterns of all its 1.25 Billion citizens as a means to access entitlements and benefits (the UIDAI slogan is "To give the poor an identity"), and to enhance social inclusion.

12. Data compression codes and protocols

- *Run-length encoding; predictive coding; dictionaries; vector quantisation.*
- *Image compression; Fast Fourier Transform algorithm; JPEG; wavelets.*

In general, a basic perspective from information theory is that where there is redundancy, there is opportunity for compression. The limit to compressibility is a representation in which no redundancy remains.

Run-length encoding (RLE) is an obvious example: if part of a data string contains (say) 1023 zeroes, one alternative would be to list them, in 1023 character bytes. An RLE code would instead summarise this substring more compactly as: "1023 $\times$ 0".

Predictive coding refers to a broad family of methods in which just the deviations from a prediction are encoded, rather than the actual sample values themselves. Today's temperature (or equity index value, etc) is usually a good prediction about tomorrow's, and so it can be more efficient to encode the data as a string of (often small) Δ variations.

Dictionary-based compression such as various lossless schemes for text, images, and audio originated by Lempel and Ziv, exploit the fact that strings of symbols have probabilities that vary much more than the probabilities of the individual symbols. Sparseness is exploitable.

Dictionary methods

Practical dictionary methods (LZW, gif, etc) first construct a dictionary by scanning through the data file and adding an entry for every new word (or byte) encountered. Because of repeated appearances, the dictionary is much smaller than the data file. Its words are sorted by frequency. For example, the word *"the"* is 7% of all written and spoken English.

An index is constructed, and then the data file is scanned a second time, replacing each word with a pointer to its position in the index.

Note that the most commonly occuring words will have the shortest indices, because the dictionary was sorted in order of descending word frequency. Decoding reads the words from the dictionary according to the index pointers. (Note that the dictionary itself is part of the payload.)

Adaptive variants of such dictionary methods monitor the compression level achieved and make an assessment about how effective the dictionary is being. A dictionary may be abandoned when it ceases to be a compact compendium for the source, and construction of a new one begun.

Vector quantisation

A lossy method that also uses dictionary look-up, for compressing strings (vectors) by finding good approximations to them, is vector quantisation. The key insight is that the space of possible combinations of samples, or symbols, is only very sparsely populated with those that actually occur.

For example, suppose we have a "coding budget" of only 15 bits that we wish to use to encode English words. If we encode them in terms of their component letters, then naïvely we might spend 5 bits on each letter, and find that we only get as far as all possible 3-letter strings, most of which are nonsense.

On the other hand, if we use the 15 bits to build pointers to entries in a codebook or dictionary of actual English words, then our address space spans $2^{15} = 32,768$ actual words. This is much more than most people use or even know. (The average US university graduate has a vocabulary of about 10,000 words.)

(Vector quantisation, continued)

This idea generalises to encoding (for example) image structure using a codebook of pixel combinations that can be the basis of making at least good approximations to the kinds of local image structure (edges, etc) that actually occur. For example, instead of encoding (4×4) pixel tiles as 16 pixels, consuming 16 bytes $= 128$ bits, a codebook with an address space of $2^{128} \approx 10^{39}$ possible image tiles could be "constructed." That is more than a *trillion-trillion-trillion* possible image tiles; ...surely enough...

Taking this codebook concept to an absurd length: across all of human history, about 10^{10} humans have lived, each for about 10^9 seconds, with visual experiences resolvable in time (when their eyes were open) at a rate of about 10 "frames"/second. Thus, our collective human history of distinguishable visual experiences could be encoded into a codebook with about 10^{20} different image frames as entries. Such an address space is spanned by pointers having a length of just 66 bits. Compare 66 bits with the length of a single SMS text message...

Obviously, VQ methods pose certain problems in terms of the codebook memory space requirements. Their operation also exemplifies the classic space-time complexity trade-off.

13. Discrete, and Fast Fourier Transform algorithms

Previously we considered functions defined on intervals or on all of $\mathbb{R}$. Now we move to discrete sequences $f[n]$ of values $f[0], f[1], \ldots, f[N-1]$.

Notation: whereas continuous functions were denoted $f(x)$ for $x \in \mathbb{R}$, discrete sequences of values at regular points are denoted with square brackets as $f[n]$ for $n \in \mathbb{Z}$ (the index values n have unit increments). Thus $f[n]$ is essentially a vector of data points, and similarly for $e_k[n]$, parameterised discrete complex exponentials forming a vector space.

We will see that discrete transforms amount to a matrix multiplication: a vector of N data points $f[n]$ is multiplied by an $(N \times N)$ matrix to yield a vector of N transform coefficients $F[k]$. Thus the computational cost is N^2 multiplications, generally complex, and N may be $\mathcal{O}(10^6)$ or more:

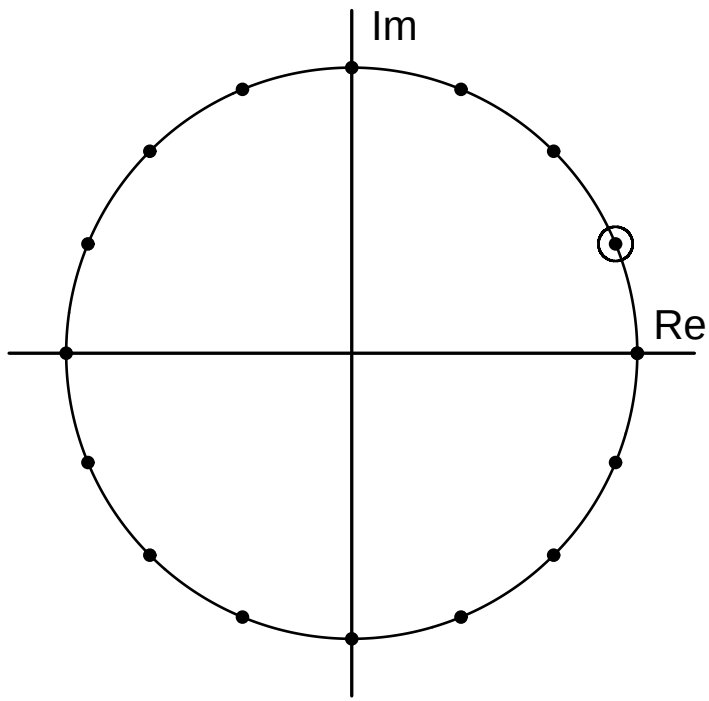
$$\begin{pmatrix} F[1] & \cdots & F[N] \end{pmatrix} = \begin{pmatrix} f[1] & \cdots & f[N] \end{pmatrix} \begin{pmatrix} e_1[1] & \cdots & e_N[1] \\ \vdots & \ddots & \vdots \\ e_1[N] & \cdots & e_N[N] \end{pmatrix}$$

(For conciseness here I wrote the indices as $1, \ldots N$ instead of $0, \ldots N-1$). A critical reduction in the computational cost of doing this, from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log_2 N)$, the FFT, is described by a leading mathematician as *"the most important numerical algorithm of our lifetime."*

Unit circle in the complex plane: N^{th} roots of unity

A fundamental property in the area of discrete transforms is that the vectors $\{e_0, e_1, \ldots, e_{N-1}\}$ form an orthogonal system in the space $\mathbb{C}^N$ with the usual inner product, where the n^{th} element of e_k is given by: $e_k[n] = e^{-2\pi ink/N}$ for $n = 0, 1, 2, \ldots, N-1$ and $k = 0, 1, 2, \ldots, N-1$.

The k^{th} vector e_k has N elements and is a discretely sampled complex exponential with frequency k. Going around the circle in either direction, its n^{th} element is an N^{th} root of unity, namely the $(nk)^{th}$ power of the primitive N^{th} root of unity:



Applying the usual inner product $\langle u, v \rangle = \sum_{n=0}^{N-1} u[n]\overline{v[n]}$

it may be shown that the squared norm:

$$||e_k||^2 = \langle e_k, e_k \rangle = N\,.$$

In practice, N will normally be a power of 2 and it will correspond to the number of discrete data samples that we have (padded out, if necessary, with 0's to the next power of 2). N is also the number of samples we need of each complex exponential (see previous "unit circle" diagram).

In fact, using the sequence of vectors $\{e_0, e_1, \ldots, e_{N-1}\}$ we can represent any data sequence $f = (f[0], f[1], \ldots, f[N-1]) \in \mathbb{C}^N$ by the vector sum

$$f = \frac{1}{N} \sum_{k=0}^{N-1} \langle f, e_k \rangle e_k\,.$$

A crucial point is that only N samples of complex exponentials e_k are required, and they are all just powers of the primitive N^{th} root of unity.

Definition (Discrete Fourier Transform, DFT)

The sequence $F[k]$, $k \in \mathbb{Z}$, defined by

$$F[k] = \langle f, e_k \rangle = \sum_{n=0}^{N-1} f[n] e^{-2\pi i n k / N}$$

is called the N-point Discrete Fourier Transform of $f[n]$.

Similarly, for $n = 0, 1, 2, \ldots, N-1$, we have the inverse transform

$$f[n] = \frac{1}{N} \sum_{k=0}^{N-1} F[k] e^{2\pi i n k / N}.$$

Note that in both these discrete series which define the Discrete Fourier Transform and its inverse, all of the complex exponential values needed are $(nk)^{th}$ powers of the primitive N^{th} root of unity, $e^{2\pi i / N}$. This is the crucial observation underlying Fast Fourier Transform (FFT) algorithms, because it allows factorization and grouping of terms together, requiring vastly fewer multiplications.

Periodicity

Note that the sequence $F[k]$ is periodic, with period N, since

$$F[k+N] = \sum_{n=0}^{N-1} f[n] e^{-2\pi i n(k+N)/N} = \sum_{n=0}^{N-1} f[n] e^{-2\pi i n k/N} = F[k]$$

using the relation

$$e^{-2\pi i n(k+N)/N} = e^{-2\pi i n k/N} e^{-2\pi i n} = e^{-2\pi i n k/N}.$$

Importantly, note that a complete DFT requires as many ($= N$) Fourier coefficients $F[k]$ to be computed as the number ($= N$) of values in the sequence $f[n]$ whose DFT we are computing.

(Both of these sequences $f[n]$ and $F[k]$ having N values repeat endlessly, but in the case of the data sequence $f[n]$ this periodicity is something of an artificial construction to make the DFT well-defined. Obviously we approach the DFT with just a finite set of N data values.)

Properties of the DFT

The DFT satisfies a range of properties similar to those of the FT relating to linearity, and shifts in either the n or k domain.

However, the convolution operation is defined a little differently because the sequences are periodic. Thus instead of an infinite integral, now we need only a finite summation of N terms, but with discrete index shifts:

Definition (Cyclical convolution)

The cyclical convolution of two periodic sequences $f[n]$ and $g[n]$ of period N, signified with an asterisk $f * g$, is defined as

$$(f * g)[n] = \sum_{m=0}^{N-1} f[m]g[n-m]\,.$$

Implicitly, because of periodicity, if $[n-m]$ is negative it is taken mod N when only N values are explicit.

It can then be shown that the DFT of $f * g$ is the product $F[k]G[k]$ where F and G are the DFTs of f and g, respectively. Thus, again, convolution in one domain becomes just multiplication in the other.

Fast Fourier Transform algorithm

The Fast Fourier Transform (of which there are several variants) exploits some clever efficiencies in computing a discrete Fourier transform (DFT).

Since the explicit definition of each Fourier coefficient in the DFT is

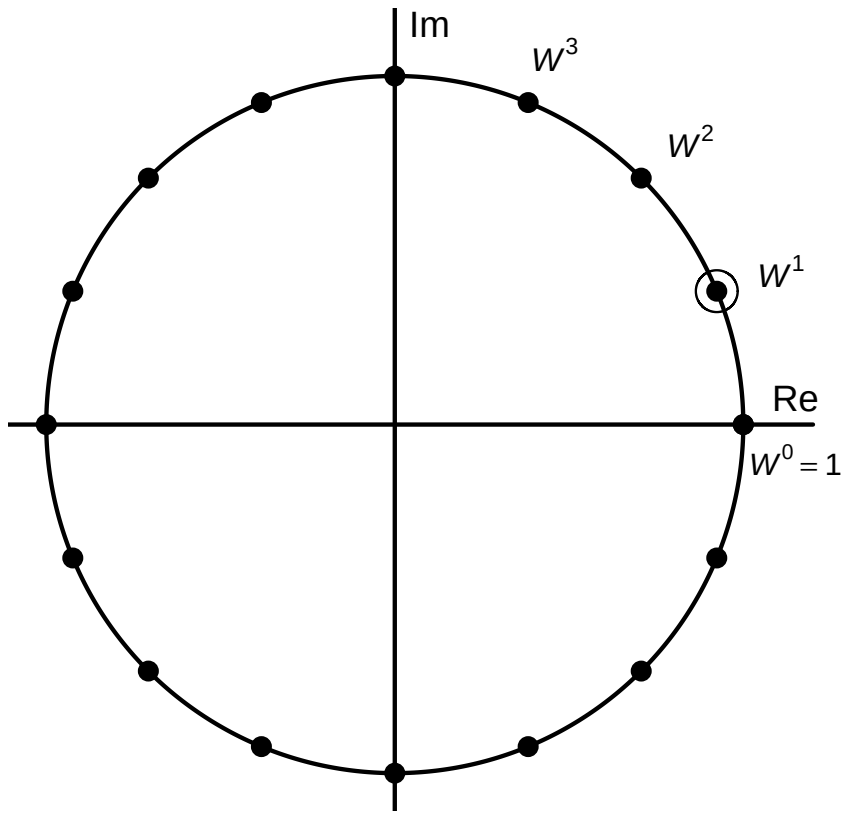
$$\begin{aligned} F[k] &= \sum_{n=0}^{N-1} f[n] e^{-2\pi i n k/N} \\ &= f[0] + f[1] e^{-2\pi i k/N} + \cdots + f[N-1] e^{-2\pi i k(N-1)/N} \end{aligned}$$

we can see that in order to compute one Fourier coefficient $F[k]$, using the complex exponential having frequency k, we need to do N (complex) multiplications and N (complex) additions. To compute all the N such Fourier coefficients $F[k]$ in this way for $k = 0, 1, 2, \ldots, N-1$ would thus require $2N^2$ such operations. Since the number N of samples in a typical audio signal (or pixels in an image) whose DFT we may wish to compute may be $\mathcal{O}(10^6)$, clearly it would be very cumbersome to have to perform $\mathcal{O}(N^2) = \mathcal{O}(10^{12})$ multiplications. Fortunately, very efficient Fast Fourier Transform (FFT) algorithms exist that instead require only $\mathcal{O}(N \log_2 N)$ such operations, vastly fewer than $\mathcal{O}(N^2)$ if N is large.

(Fast Fourier Transform algorithm, con't)

Recall that all the multiplications required in the DFT involve the N^{th} roots of unity, and that these in turn can all be expressed as powers of the primitive N^{th} root of unity: $e^{2\pi i/N}$.

Let us make that explicit now by defining this constant as $W = e^{2\pi i/N}$ (which is just a complex number that depends only on the data length N which is presumed to be a power of 2), and let us use W to express all the other complex exponential values needed, as the $(nk)^{th}$ powers of W: $e^{2\pi ink/N} = W^{nk}$.



(Fast Fourier Transform algorithm, con't)

Or going around the unit circle in the opposite direction, we may write:

$$e^{-2\pi ink/N} = W^{-nk}$$

The same N points on the unit circle in the complex plane are used again and again, regardless of which Fourier coefficient $F[k]$ we are computing using frequency k, since the different frequencies are implemented by skipping points as we hop around the unit circle.

Thus the lowest frequency $k = 1$ uses all N roots of unity and goes around the circle just once, multiplying them with the successive data points in the sequence $f[n]$. The second frequency $k = 2$ uses every second point and goes around the circle twice for the N data points; the third frequency $k = 3$ hops to every third point and goes around the circle three times; etc.

Because the hops keep landing on points around the unit circle from the same set of N complex numbers, and the set of data points from the sequence $f[n]$ are being multiplied repeatedly by these same numbers for computing the N needed Fourier coefficients $F[k]$, it is possible to exploit some clever arithmetic tricks and an efficient recursion.

(Fast Fourier Transform algorithm, con't)

Let us re-write the expression for Fourier coefficients $F[k]$ now in terms of powers of W, and divide the series into its first half plus second half in n.

$$
\begin{aligned}
F[k] &= \sum_{n=0}^{N-1} f[n] e^{-2\pi i n k/N} = \sum_{n=0}^{N-1} f[n] W^{-nk}, \quad k = 0, 1, \cdots, N-1 \\
&= \sum_{n=0}^{N/2-1} f[n] W^{-nk} + \sum_{n=N/2}^{N-1} f[n] W^{-nk} \\
&= \sum_{n=0}^{N/2-1} (f[n] + W^{-kN/2} f[n+N/2]) W^{-nk} \\
&= \sum_{n=0}^{N/2-1} (f[n] + (-1)^k f[n+N/2]) W^{-nk}
\end{aligned}
$$

where the last two steps capture the second half of the series within the first half, using also $W^{-kN/2} = e^{-\pi i k} = (-1)^k$. Note that a total of N Fourier coefficients $F[k]$ are still defined here, but the summation range for n is only half-length: the series now has only $N/2$ product terms.

(Fast Fourier Transform algorithm, con't)

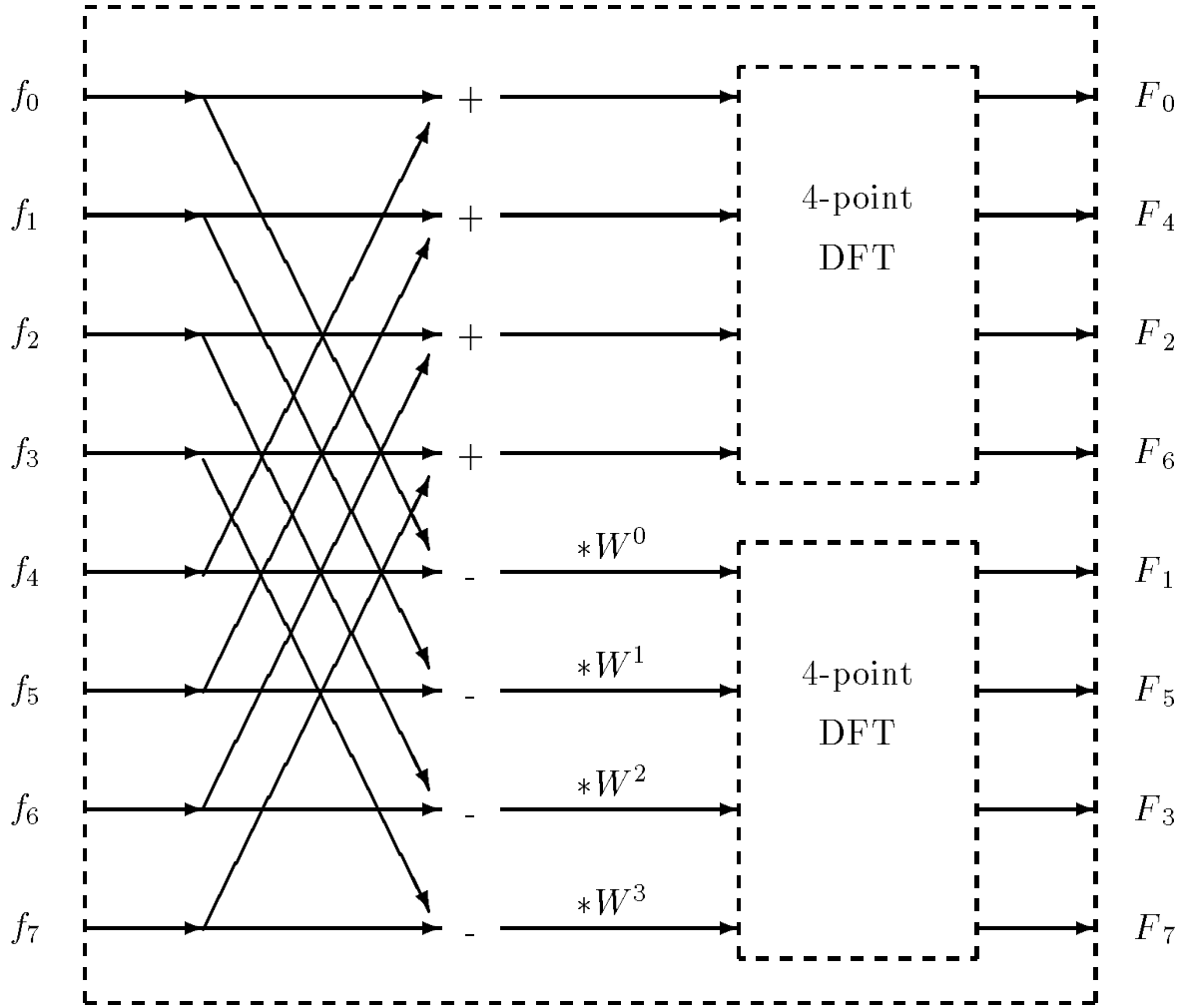
Separating this out into two different series we create $F_e[k]$ and $F_o[k]$, each with $N/2$ values, using $W^{-n(2k)}$ in $F_e[k]$ and $W^{-n(2k+1)}$ in $F_o[k]$:

$$F_e[k] = \sum_{n=0}^{N/2-1} (f[n] + f[n+N/2]) W^{-n(2k)}, \quad k = 0, 1, \cdots, N/2-1$$

$$F_o[k] = \sum_{n=0}^{N/2-1} (f[n] - f[n+N/2]) W^{-n} W^{-n(2k)}, \quad k = 0, 1, \cdots, N/2-1$$

where the odd sequence $F_o[k]$ has factored $W^{-n(2k+1)}$ to $W^{-n}W^{-n(2k)}$.

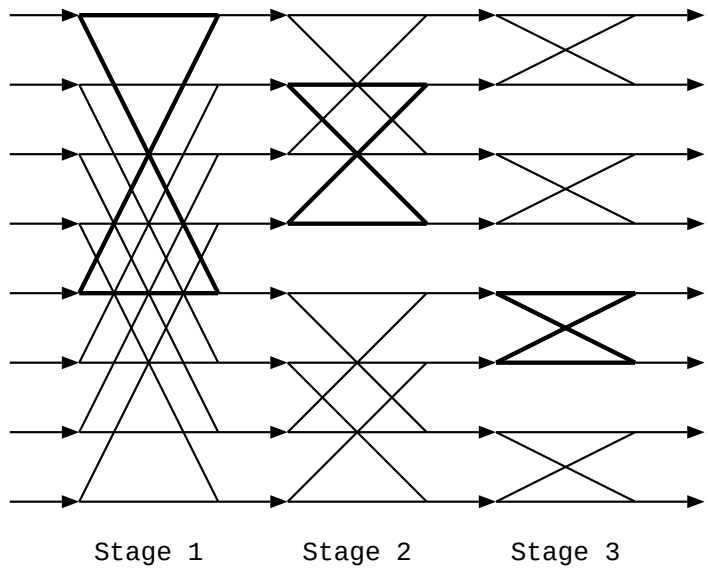
The beauty of this "divide and conquer" strategy is that we replace a Fourier transform of length N with two of length $N/2$, but each of these requires only one-quarter as many multiplications. The wonderful thing about the Danielson-Lanczos Lemma is that this can be done recursively: each of the half-length Fourier transforms $F_e[k]$ and $F_o[k]$ that we end up with can further be replaced by two quarter-length Fourier transforms, and so on down by factors of 2. At each stage, we combine input data (adding, or subtracting followed by complex multiplications by W^{-n}).



To compute the N Fourier coefficients $F[k]$ using this recursion we must perform $N/2$ complex multiplications every time we divide DFT size by 2, and given that initial data length N is some power of 2, we can do this $\log_2 N$ times until we end up with just a trivial 1-point transform. Thus, the complexity of this algorithm is $\mathcal{O}(N \log_2 N)$ for data of length N.

The repetitive pattern formed by adding or subtracting pairs of points halfway apart in each decimated sequence has led to this algorithm (popularized by Cooley and Tukey in 1965) being called the Butterfly.

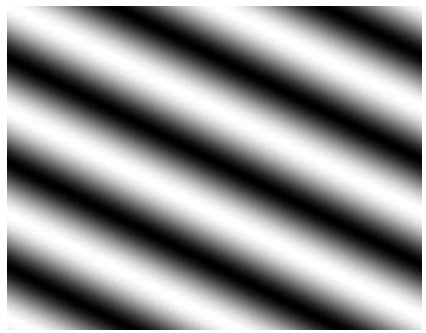
This pattern produces the output Fourier coefficients in bit-reversed positions: to locate $F[k]$ in the FFT output array, take k as a binary number of $\log_2 N$ bits, reverse them and treat as the index into the array. Storage requirements of this algorithm are only $\mathcal{O}(N)$ in space terms.



Extensions to higher dimensions

All of the Fourier methods we have discussed so far have involved only functions or sequences of a single variable. Their Fourier representations have correspondingly also been functions or sequences of a single variable.

But all Fourier techniques can be generalized and apply also to functions of any number of dimensions. For example, images (when pixelized) are discrete two-dimensional sequences $f[n, m]$ giving a pixel value at row n and column m. Their Fourier components are 2D complex exponentials having the form $f[n, m] = e^{2\pi i(kn/N+jm/M)}$ for an image of dimensions NxM pixels, and they have the following "plane wave" appearance with both a "spatial frequency" $\sqrt{k^2 + j^2}$ and an orientation $\arctan(j/k)$:



Similarly, crystallography uses 3D Fourier methods to infer atomic lattice structure from the phases of X-rays scattered by a slowly rotating crystal.

14. Wavelet representations of information

Wavelets are further bases for representing information, and have received much interest in both theoretical and applied fields over the past 30 years. They combine aspects of the Fourier (frequency-based) approaches with restored locality, because wavelets are size-specific **local** undulations.

Some of the most important work in this field was done by the Belgian physicist and mathematician Ingrid Daubechies, for whom a broad class of wavelets are named, and which were adopted as the basis for JPEG-2000 image compression (the successor to the JPEG image format).

Ingrid Daubechies

JPEG compression

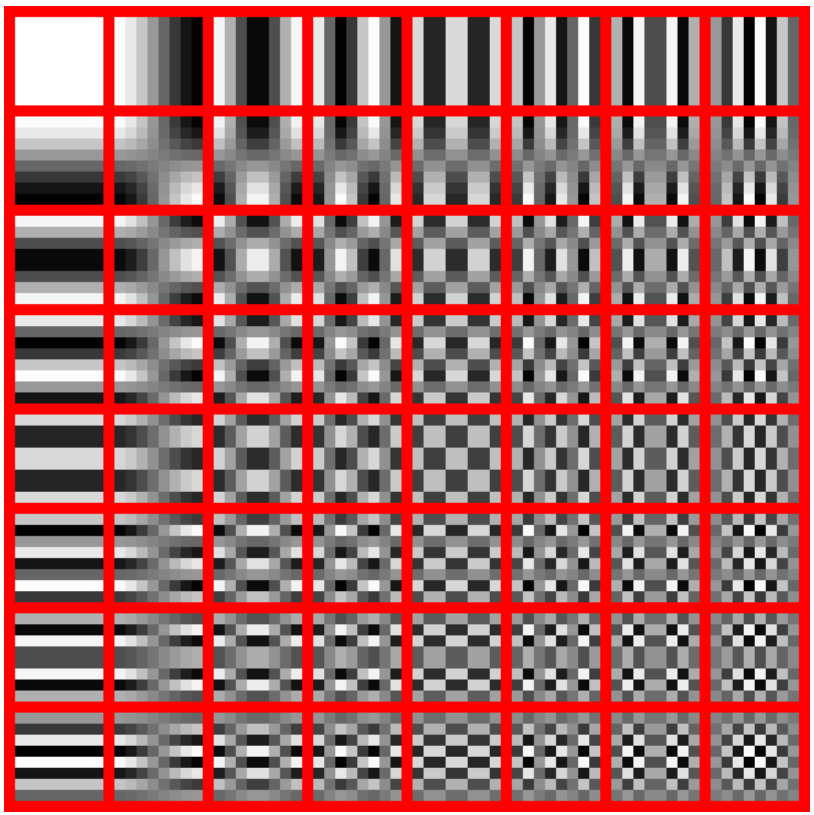
Fourier transforms have vast applications in information processing, but one area central to information theory is data and image compression. Images are compressible because neighbouring pixels tend to be highly correlated. Projecting such data onto a Fourier (or Fourier-like) basis produces highly decorrelated coefficients, and a great many that are 0 or are so small that they need not be encoded.

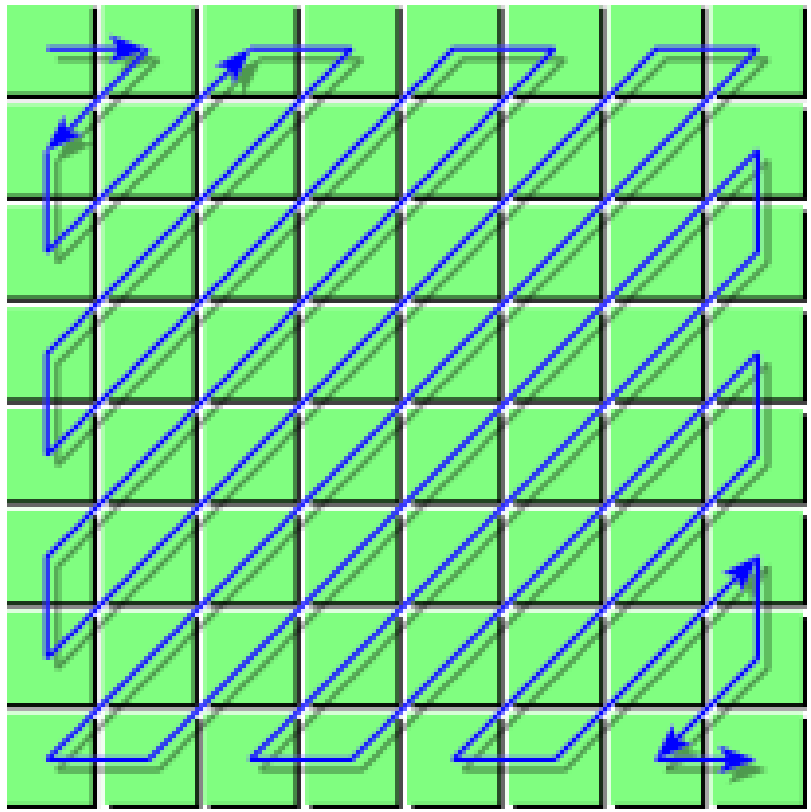
JPEG image compression is based on running a discrete cosine transform (DCT) on small "tiles" of an image, usually (8×8) pixels. For simplicity, discrete cosine functions whose 2D frequencies also form an (8×8) array are used for the projections. The resulting DCT coefficients are quantised, more coarsely for the higher frequencies so fewer bits are used to encode them, and there are long runs of zeroes captured by run-length coding.

Compression factors $> 10{:}1$ are achieved with minimal perceived loss. JPEG compression depends on aspects of human perception, which isn't bothered about the amplitudes of the higher frequencies being accurate. The lower frequencies do matter, but they are far fewer in number. A quantisation table specifies how severely quantised the coefficients are for different frequencies; *e.g.* 2 bits for high frequencies, 7 bits for low.

Discrete Cosine Transform

The 2D cosine patterns used in the DCT (as images are real-valued only) are shown on the left. Their 2D frequency vectors (ω_x, ω_y) form an (8×8) array. Pixel tiles that are also (8×8) arrays are projected onto these, and linear combinations of them can represent any such image. The read-out sequence on the right ensures that there are long runs of 'zeroes' coefficients towards the end, suitable for efficient RLE coding.





Example of JPEG compression by 20:1

Left: An uncompressed monochrome image, encoded at 8 bits per pixel (**bpp**). **Right**: JPEG compression by 20:1 (Qual = 10), **0.4 bpp**. The foreground water shows some blocking artifacts at 20:1, and some patches of the water texture are obviously represented by a single vertical cosine in an (8×8) pixel block.

JPEG: compression factor (CF) vs quality factor (QF)

Illustration of progressive variation (left-to-right) in CF $\downarrow$ and QF $\uparrow$

Dyadic wavelets

This approach fits into the general scheme of expanding a function $f(x)$ using orthonormal functions. Dyadic transformations of some generating wavelet $\Psi(x)$ spawn an orthonormal wavelet basis $\Psi_{jk}(x)$, for expansions of functions $f(x)$ by doubly-infinite series with wavelet coefficients c_{jk}:

$$f(x) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} c_{jk} \Psi_{jk}(x)$$

The wavelets $\Psi_{jk}(x)$ are generated by shifting and scaling operations applied to a single original function $\Psi(x)$, known as the mother wavelet.

The orthonormal "daughter wavelets" are all dilates and translates of their mother (hence "dyadic"), and are given for integers j and k by

$$\Psi_{jk}(x) = 2^{j/2} \Psi(2^j x - k)$$

If we were dealing with 2D wavelets $\Psi(x, y)$ as in image analysis, we would also include a parameter θ for rotation in the plane, as well as a second shift parameter for the second dimension.

Wavelet dilations and translations

The dyadic dilates of $\Psi(x)$, namely,

$$\ldots, \Psi(2^{-2}x), \Psi(2^{-1}x), \Psi(x), \Psi(2x), \Psi(2^2x), \ldots$$

have widths $\ldots, 2^2, 2^1, 1, 2^{-1}, 2^{-2}, \ldots$ respectively.

Since the dilate $\Psi(2^j x)$ has width 2^{-j}, its translates

$$\Psi(2^j x - k) = \Psi(2^j(x - k2^{-j})), \quad k = 0, \pm 1, \pm 2, \ldots$$

cover the whole x-axis. The computed coefficients c_{jk} constitute a Wavelet Transform of the function or data. There are many different possible choices for the mother wavelet function, tailored for different purposes. Of course, the wavelet coefficients c_{jk} that result will be different for those different choices of wavelets.

Just as with Fourier transforms, there are fast wavelet implementations that exploit structure. Typically they work in a coarse-to-fine pyramid, with each successively finer scale of wavelets applied to the difference between a down-sampled version of the original function and its full representation by all preceding coarser scales of wavelets.

Interpretation of c_{jk}

How should we intrepret the wavelet coefficients c_{jk}?

A common requirement for wavelets is that they have strictly compact support: they vanish outside of some interval determined by their scale parameter j.

Because wavelets are thus localised, their coefficients c_{jk} give us information about the behaviour of data near the point $x = k2^{-j}$, measured on the scale of 2^{-j}.

For example, the coefficients $c_{(-10,k)}$, $k = 0, \pm 1, \pm 2, \ldots$ correspond to variations that take place over intervals of length $2^{10} = 1024$, while the coefficients $c_{(10,k)}$ $k = 0, \pm 1, \pm 2, \ldots$ correspond to fluctuations over intervals of length 2^{-10}.

These observations help explain how wavelet representations extract local structure over many different scales of analysis and can be exceptionally efficient schemes for describing data. This makes them powerful tools for analyzing signals, capturing structure and recognizing patterns.

Properties of naturally arising data

Much naturally arising data is better represented and processed using wavelets, because wavelets are localized and better able to cope with discontinuities and with structures of limited extent than are global Fourier components. Many applied fields now make use of wavelets, including acoustics, signal and image processing, neurophysiology, magnetic resonance imaging, speech discrimination, optics, fractals, turbulence, EEG, ECG, earthquake prediction, radar, etc.

Another common aspect of naturally arising data is self-similarity across scales, similar to the fractal property. For example, nature abounds with concatenated branching structures at successive size scales. The dyadic generation of wavelet bases mimics this self-similarity.

Finally, wavelets are tremendously good at data compression. This is because they decorrelate data locally: the information is statistically concentrated in just a few wavelet coefficients. The old standard image compression tool JPEG was based on squarely truncated sinusoids. The new JPEG-2000, based on Daubechies wavelets, is a superior compressor.

Case study in image compression: comparison between DCT and Discrete Wavelet Transform (DWT) encodings

In 1994, the JPEG Standard was published for image compression using local 2D Fourier transforms (actually Discrete Cosine Transforms [DCT] since images are real, not complex) on small (8×8) tiles of pixels. Each transform produces 64 coefficients and so is not itself a reduction in data.

But because high spatial frequency coefficients can be quantised much more coarsely than low ones for satisfied human perceptual consumption, a quantisation table allocates bits to the Fourier coefficients accordingly. The higher frequency coefficients are resolved with fewer bits (often 0).

As we saw before, by reading out these coefficients in a low-frequency to high-frequency sequence, long runs of 0's arise which allow run-length codes (RLE coding) to be very efficient. About 10:1 image compression causes little perceived loss. Both encoding and decoding (compression and decompression) are easily implemented at video frame-rates.

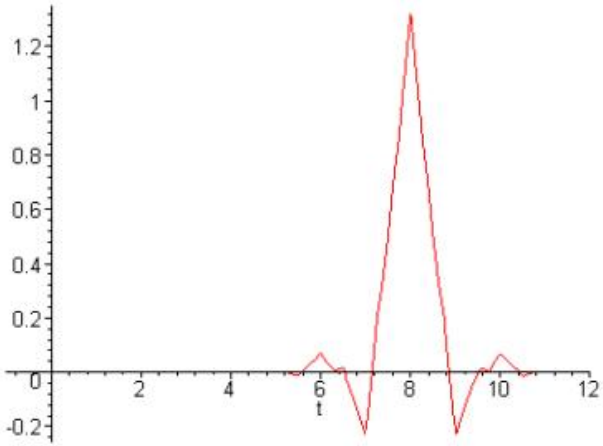
ISO/IEC 10918: *JPEG Still Image Compression Standard.*

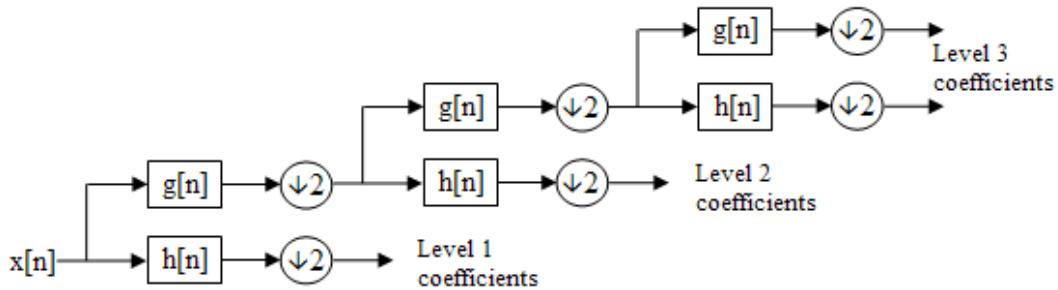
JPEG = Joint Photographic Experts Group `http://www.jpeg.org/`

(Image compression case study, continued: DCT and DWT)

Although JPEG performs well on natural images at compression factors below about 20:1, it suffers from visible block quantisation artifacts at more severe levels. The DCT basis functions are just square-truncated sinusoids, and if an entire (8×8) pixel patch must be represented by just one (or few) of them, then the blocking artifacts become very noticeable.

In 2000 a more sophisticated compressor was developed using encoders like the Daubechies 9/7 wavelet shown below. Across multiple scales and over a lattice of positions, wavelet inner products with the image yield coefficients that constitute the Discrete Wavelet Transform (DWT): this is the basis of JPEG-2000. It can be implemented by recursively filtering and downsampling the image vertically and horizontally in a scale pyramid.





ISO/IEC 15444: *JPEG2000 Image Coding System.* `http://www.jpeg.org/JPEG2000.htm`

Comparing image compressor bit-rates: DCT vs DWT

Left: JPEG compression by 20:1 (Q-factor 10), **0.4 bpp**. The foreground water already shows some blocking artifacts, and some patches of the water texture are obviously represented by a single vertical cosine in an (8×8) pixel block.
Right: JPEG-2000 compression by 20:1 (same reduction factor), **0.4 bpp**. The image is smoother and does not show the blocking quantization artifacts.

Comparing image compressor bit-rates: DCT vs DWT

Left: JPEG compression by 50:1 (Q-factor 3), **0.16 bpp**. The image shows severe quantization artifacts (local DC terms only) and is rather unacceptable.

Right: JPEG-2000 compression by 50:1 (same reduction factor), **0.16 bpp**. At such low bit rates, the Discrete Wavelet Transform gives much better results.

15. Kolmogorov complexity; minimal description length

- *Minimal description length of data.*
- *Algorithmic complexity; computability. Fractals.*

Any set of data can be created by a program, even if (in the worst case) that program simply consists of data statements. The length of such a program defines its algorithmic complexity.

A fundamental idea is the measure known as Kolmogorov complexity: the complexity of a string of data can be defined as the length of the shortest executable program for computing the string. Thus the complexity is the data string's "minimal description length."

It is an amazing fact that the Kolmogorov complexity K of a string is approximately equal to the entropy H of the distribution from which the data string is a randomly drawn sequence. Thus Kolmogorov complexity is sometimes called algorithmic information theory, and indeed K is one way to define the ultimate limits of data compression.

Reducing the data to a program that generates it exactly is obviously a way of compressing it. Running the program is a way of decompressing it.

(Kolmogorov complexity, continued)

It is very important to draw a clear distinction between the notions of *computational complexity* (measured by program execution time), and *algorithmic complexity* (measured by program length). Kolmogorov complexity is concerned with descriptions which minimize the latter.

Most sequences of length n (all possible permutations of n bits) have Kolmogorov complexity K close to n. The complexity of a truly random binary sequence is as long as the sequence itself. However, it is not clear how to be certain of discovering that a given string has a complexity much lower than its length. It might be clear that the string

0101010101010101010101010101010101010101010101010101010101

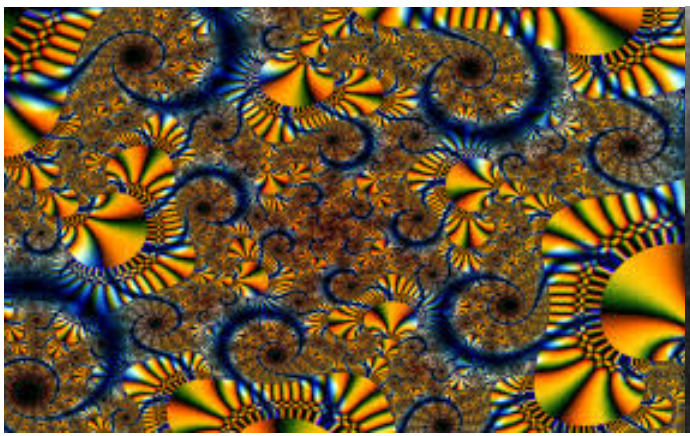
has a complexity much less than 32 bits; indeed, its complexity is the length of the program: `Print 32 "01"s`. But consider the string

0110101000001001111001100110011111110011101111001100100100001000

which looks random and passes most tests for randomness. How could you discover that this sequence is in fact just the binary expansion for the irrational number $\sqrt{2}-1$, so therefore it can be specified concisely?

(Kolmogorov complexity, continued)

Fractals are examples of entities that look very complex but in fact are generated by very simple programs (*i.e.*, iterations of a mapping). Therefore, the Kolmogorov complexity of fractals is nearly zero.



In general, Kolmogorov complexity K is not computable: you never know for sure that you have found the shortest possible description of a pattern. Lacking a decision procedure, finding K illustrates the halting problem.

A sequence $x_1, x_2, x_3, ..., x_n$ of length n is deemed *algorithmically random* if its Kolmogorov complexity seems at least n (*i.e.*, the shortest program that can generate the sequence is a listing of the sequence itself):

$$K(x_1 x_2 x_3 ... x_n | n) \geq n$$

(Kolmogorov complexity, continued)

An infinite string is defined to be *K-incompressible* if its Kolmogorov complexity, in the limit as the string gets arbitrarily long, approaches the length n of the string itself:

$$\lim_{n\to\infty} \frac{K(x_1x_2x_3...x_n|n)}{n} = 1$$

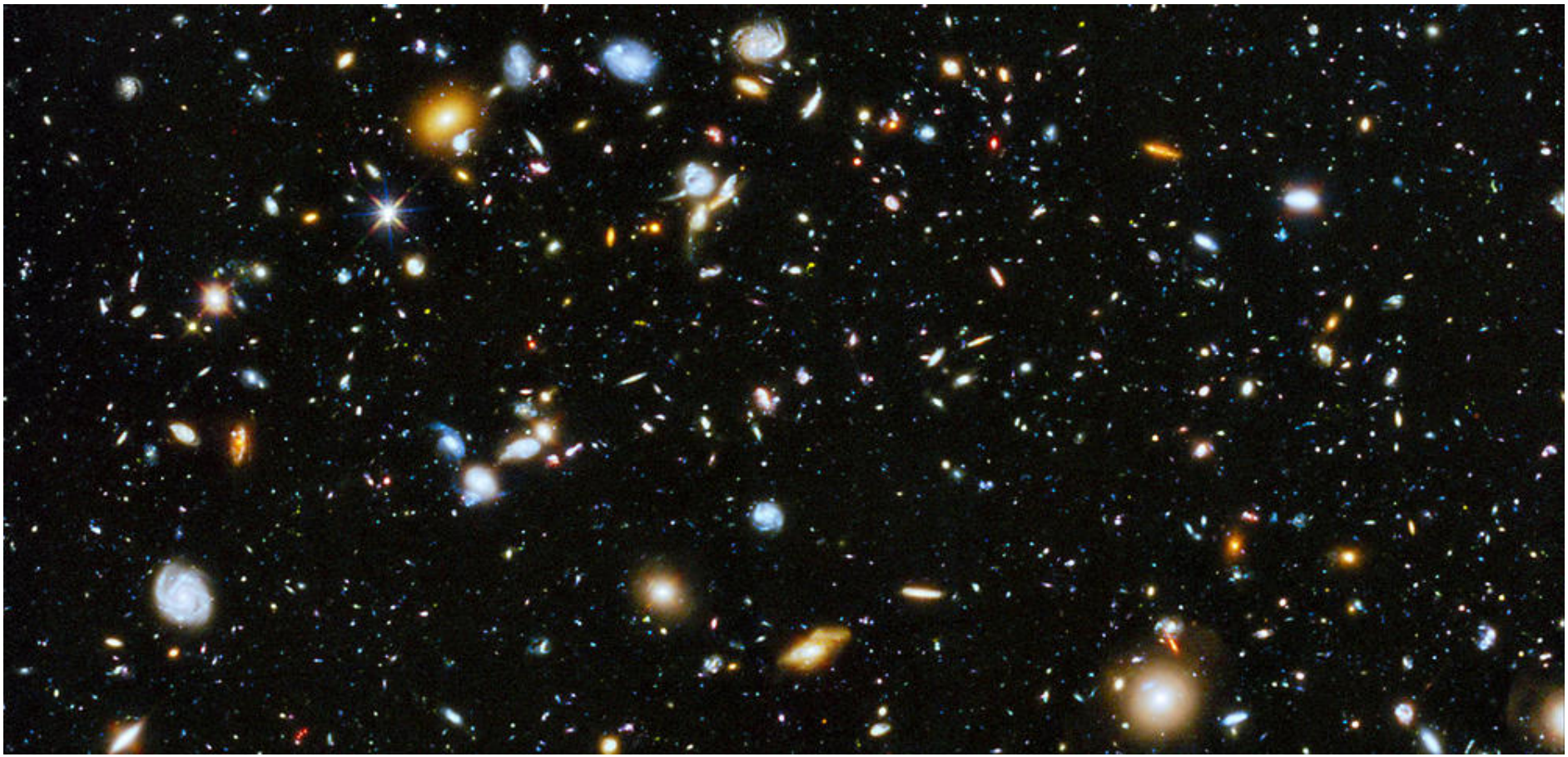
An interesting theorem, called the Strong Law of Large Numbers for Incompressible Sequences, asserts that the proportions of 0's and 1's in any incompressible string must be nearly equal.

Moreover, any incompressible sequence must satisfy all computable statistical tests for randomness. In this sense the algorithmic test for randomness is the ultimate test, since it includes within it all other computable tests for randomness.

There are considered to be deep connections between data compression and artificial intelligence: machine learning is about making predictions from data, and predictability is the basis of data compression.

16. Some scientific applications of information theory

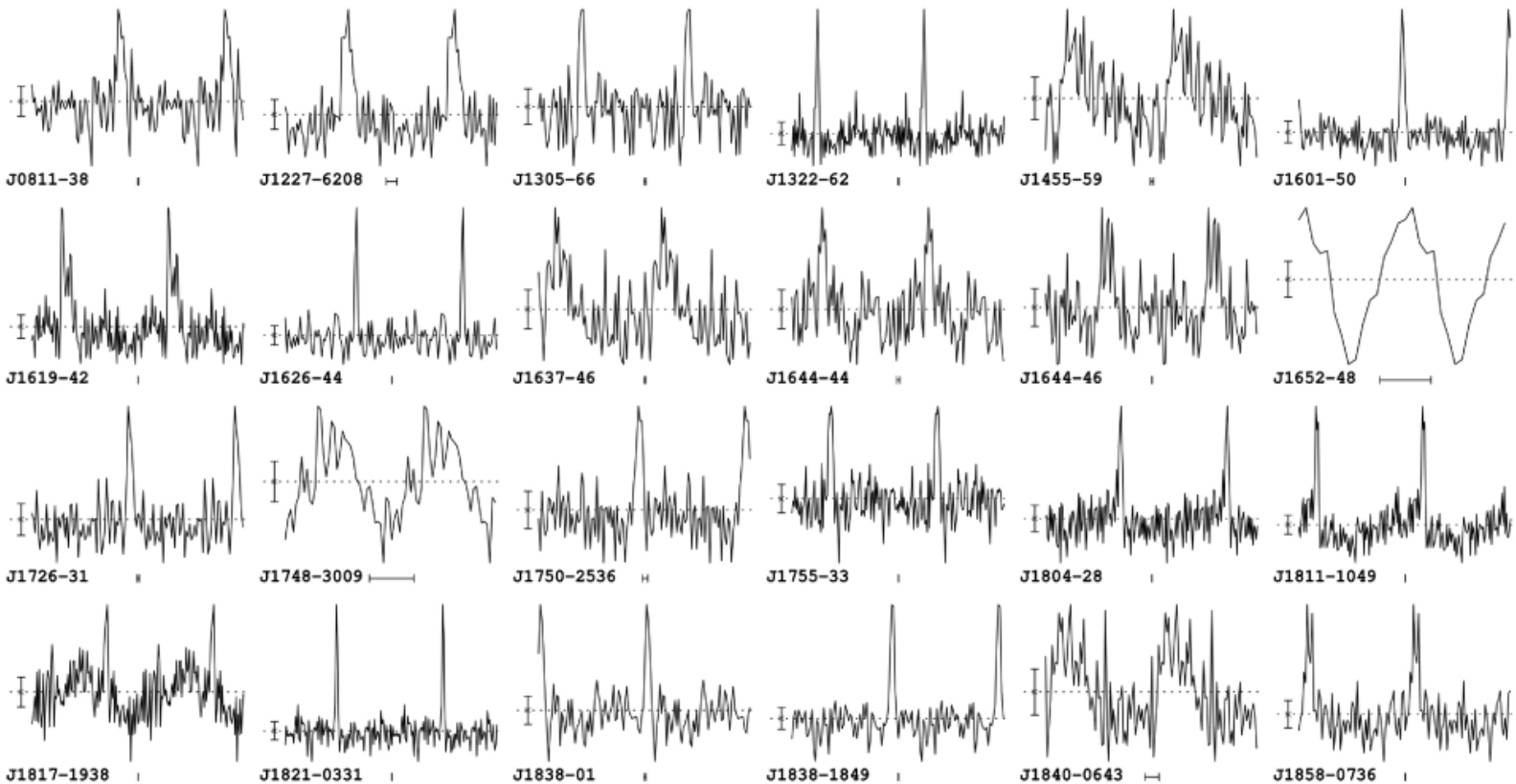
- *Astrophysics; pulsar detection. Extracting signals buried in noise.*
- *Information theory perspectives in genomics. Family trees. Why sex?*
- *Information theory perspectives in neuroscience.*
- *Biometric pattern recognition.*



Hubble Ultra Deep Field. (Points are stars in our galaxy; diffuse structures are entire distant galaxies.)

Interpreting astrophysical signals buried in noise

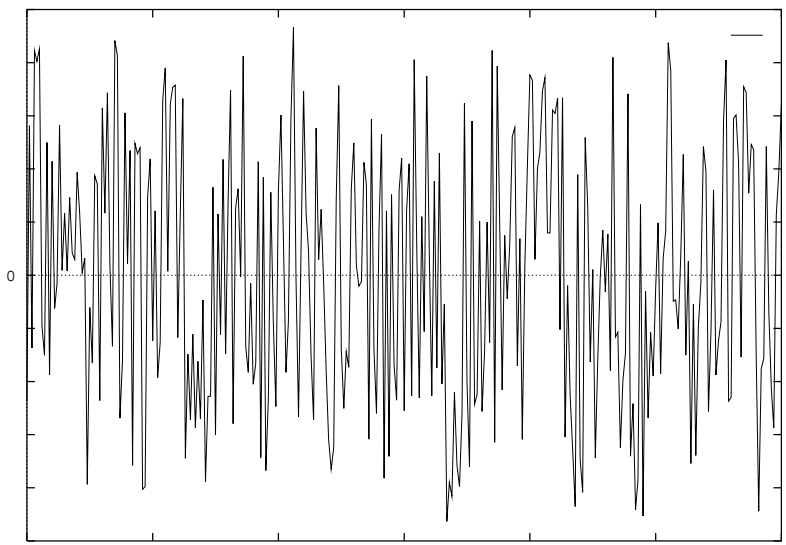
Pulsars are collapsed neutron stars, the dark remnants of a supernova. Because of their tiny size but high mass density, they spin very rapidly (up to 1,000 rotations/sec). Having a strong magnetic dipole, a pulsar's spinning causes emission of an electromagnetic radio beam. If the earth happens to be in the cone of the beam, the signal arrives pulsed at a very precise frequency, the rotation frequency, rather like a lighthouse beacon. But these signals are very faint and buried in the "galactic noise."

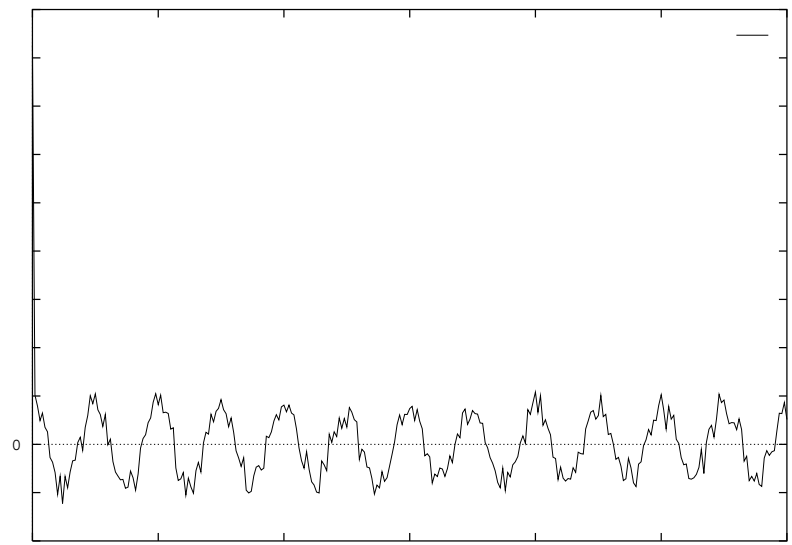


Extracting signals buried in noise: auto-correlation function

Although the noise may be much larger, the signal has the advantage of being coherent (periodic with a precise frequency). The auto-correlation integral $\rho_f(t)$ extracts this periodic component from the combined $f(t)$ (left panel), as the incoherent noise tends to average out against itself:

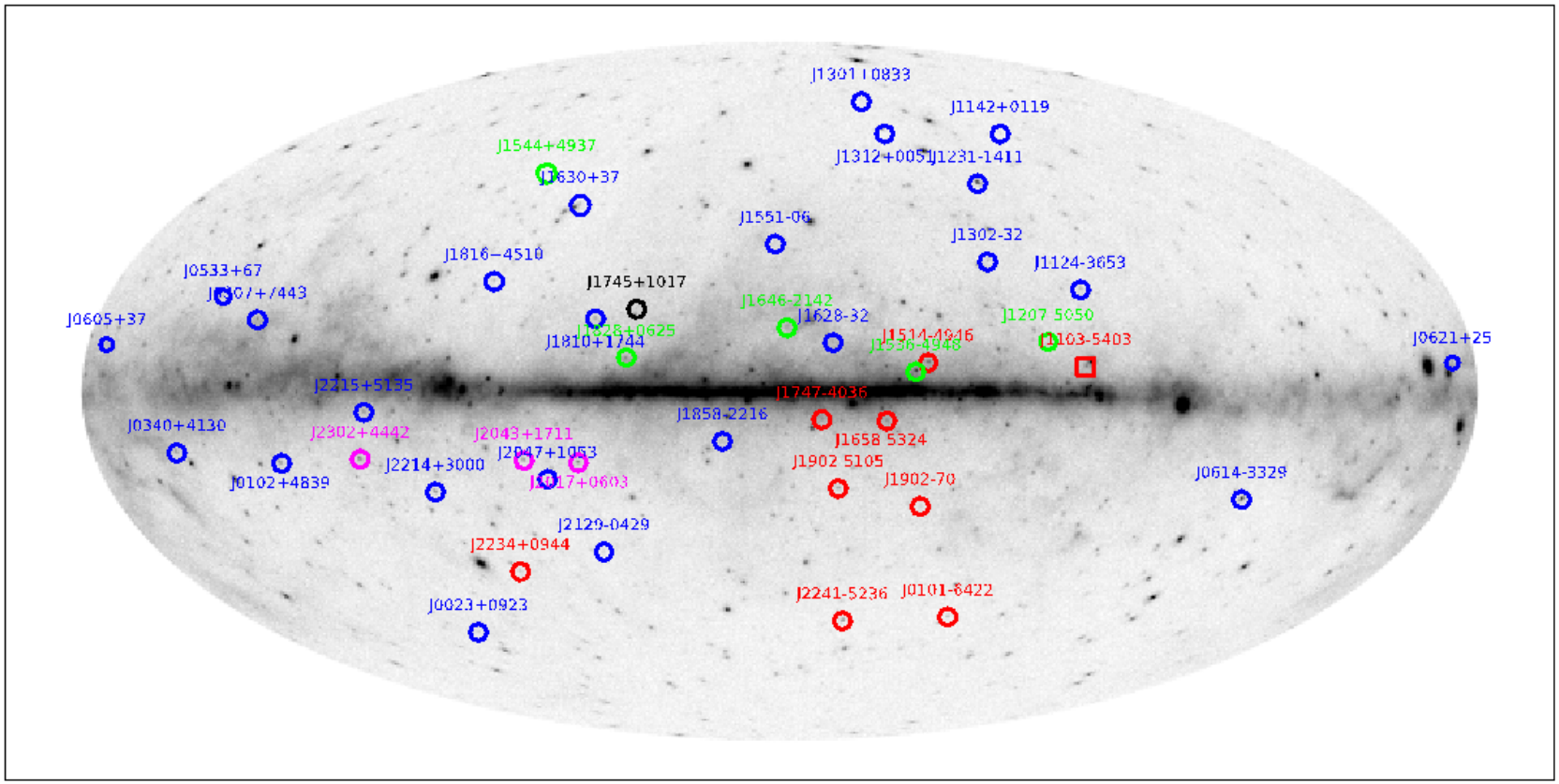
$$\rho_f(t) = \int_{-\infty}^{\infty} f(\tau) f(t+\tau) d\tau \quad \text{(right panel).}$$





Note the resemblance to the convolution of a function with itself, except without the flip: we have $f(t+\tau)$ instead of $f(t-\tau)$ inside the integral.

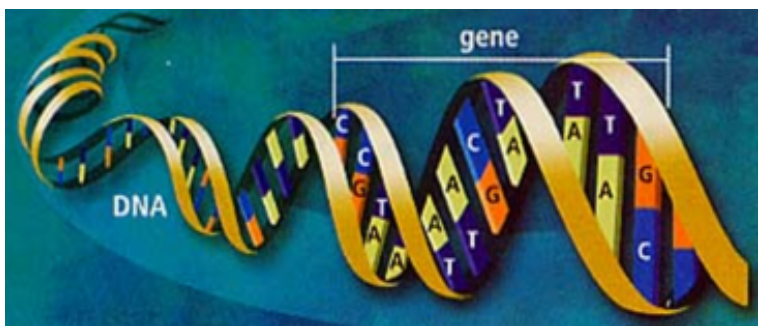
Pulsar detections within our galaxy (viewed "on-edge")



It is convenient to compute auto-correlation using Fourier transforms, because if the combined signal $f(t)$ has $\mathcal{FT}\{f(t)\} = F(\omega)$, then the Fourier transform of the auto-correlation function $\rho_f(t)$ that we seek is simply the power spectral density of $f(t)$: $\mathcal{FT}\{\rho_f(t)\} = F(\omega)F^*(\omega)$. Thus $\rho_f(t) = \mathcal{FT}^{-1}\{F(\omega)F^*(\omega)\}$ is extracted by the inverse transform.

Information theory perspectives on genomics

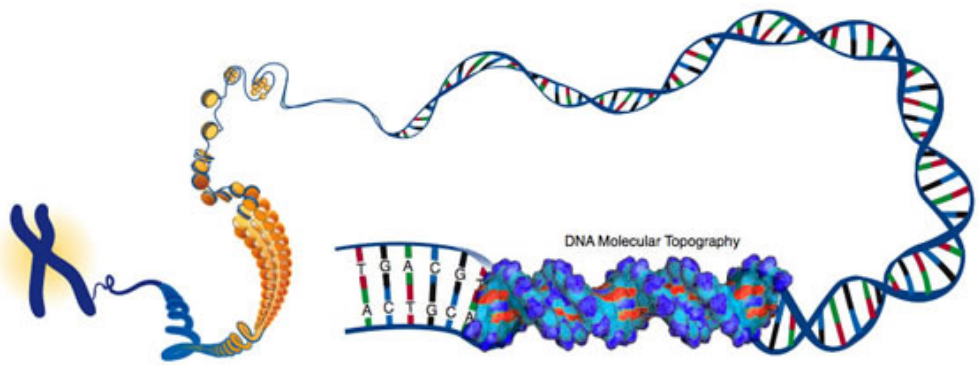
- The information in DNA is coded in terms of ordered triples of four nucleotide bases (Adenine, Guanine, Cytosine, Thymine: A,G,C,T).
- These ordered triples are called codons. They create 64 possible permutations (4 bases taken 3 at a time: $4^3 = 64$ permutations).
- Codons specify amino acids, the building blocks of proteins. There are only 20 primary amino acids, so any given one may correspond to more than one codon. (For example: six codons all mean *serine*.)
- The human genome consists of about 6 billion nucleotide bases, or about 2 billion codons each able to signify ~ 6 bits of information.
- DNA has even been proposed for general-purpose data storage.



- Two well-known statistics from comparative genomics:
  1. About 99.9% of DNA sequences are the same between persons.
  2. Humans and chimpanzees share about 97% of their DNA sequences.

(Information theory perspectives on genomics, con't)

- Only about 3% of the human genome specifies genes, of which there are some 20,000. Average gene length $\sim$ 8,000 nucleotide base pairs.
- Since each choice of {A,G,C,T} entails 2 bits, an average gene *could* specify about 16,000 bits at this molecular level.
- Differences among individuals (apart from traits that are epigenetic: outside of genetic control) arise from very sparse DNA differences.
- Since a typical gene comprises about 8,000 base pairs but 99.9% of our DNA sequences are the same, it appears that differences in a gene involve on average only about $\frac{8{,}000}{1{,}000} = 8$ base pairs.
- The process of translating the genetic code into corresponding amino acids can be regarded as an error-prone information channel.
- Errors in translation (mutations) are both the engine of evolution (if passed on), and the causes of cancers. Most mutations are fatal.

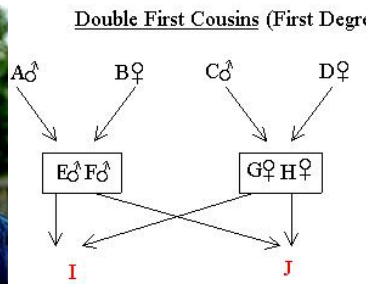


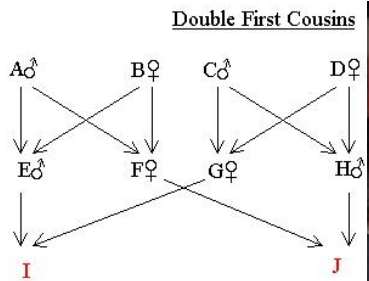
(Information theory perspectives on genomics, con't)

- Most (not all) of your genes may be inherited from either of your parents with equal probability, so the entropy of gene inheritance from either parent is about 1 bit per gene.
- Going back N generations to 2^N ancestors (all presumed different), this probability is 2^{-N} and the entropy of gene inheritance becomes:

$$H = -\sum_{1}^{2^N} \frac{1}{2^N} \log_2 \frac{1}{2^N} = -\log_2(2^{-N}) = N \quad \text{bits per gene.}$$

- You share $\frac{1}{2}$ of your genes with a parent; $\frac{1}{4}$ with any grandparent; $\frac{1}{2}$ with full siblings; $\frac{1}{2}$ with any double cousins of 1^{st} degree; and 100% with a monozygotic twin (or inevitably coming, a clone).





Shared genes: 100% 50% 25% 50%

Catastrophe: where are all of your missing ancestors??

- Going back N generations, your family tree necessarily acquires 2^N "slots". Let's go back to shortly before the Norman Conquest.
- Assuming a new generation is born every 20 to 25 years on average, 40 to 50 generations were spawned during these past 10 centuries.
- But $2^{40} \approx 10^{12}$ is a trillion ancestors. $2^{50} \approx 10^{15}$ is a million billions.
- WHERE will you find trillions of ancestors, when the population of Europe was only a few million people?
- The inevitable answer: everybody then living (who had descendants) was your ancestor; and on average, each such person re-appears in your family tree millions, or even billions, of times.
- Family trees are not branching trees for long; they quickly absorb entire reproductive populations and then collapse in on themselves. A common ancestor for the entire indigenous population of Europe existed in early 1400's (MRCA: "Most Recent Common Ancestor").
- The genetic isopoint is the time in history when *everyone* (who has descendants) is an ancestor of everyone now. For Europeans today, the isopoint is in the 10th Century. For a large well-mixed population of size m, the isopoint was about $N \approx 1.77 \log_2(m)$ generations ago.

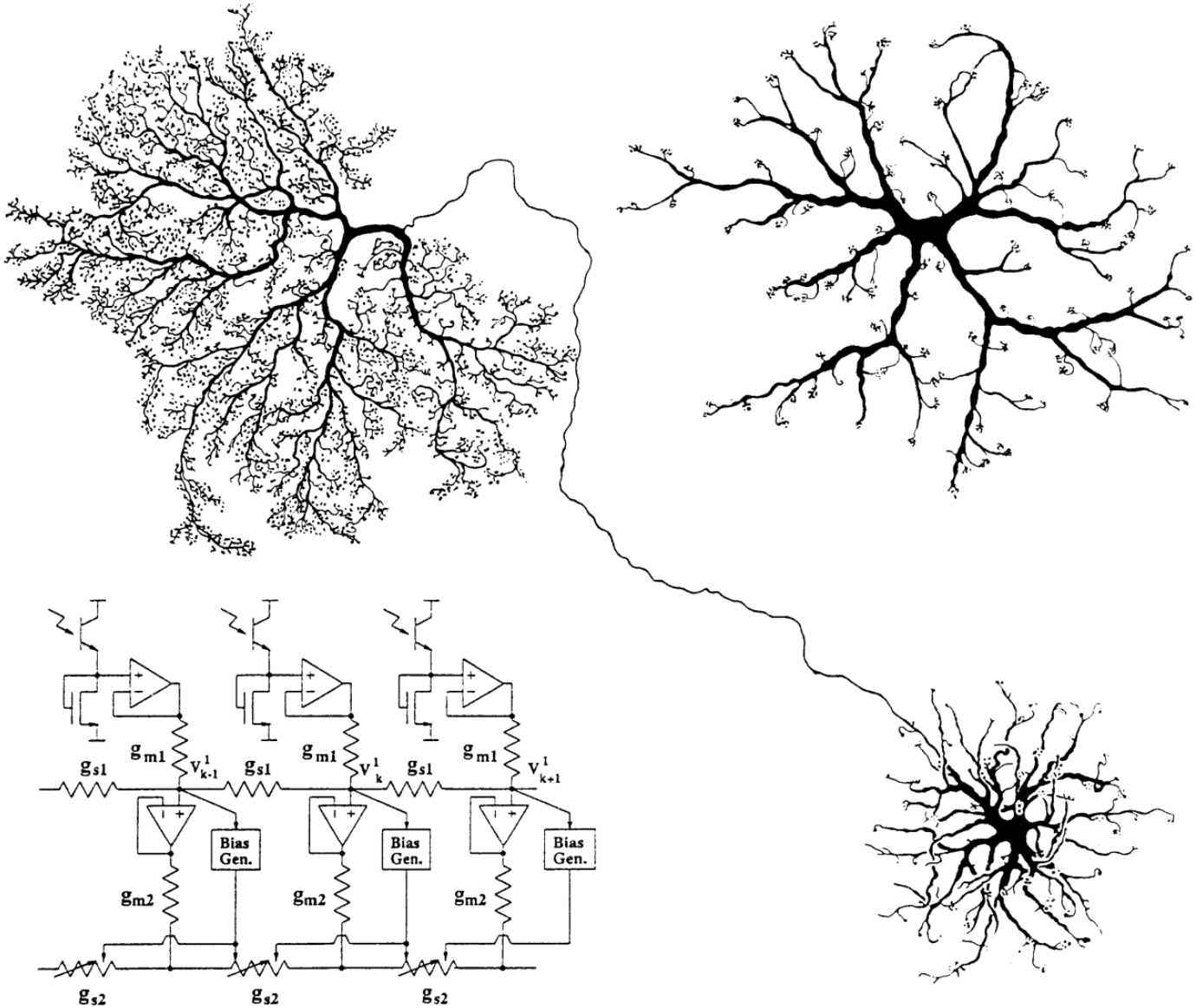
Communal co-ancestry of descendants after N generations

- By the same 2^N logic, almost everyone around you today will become the co-ancestors of all of your descendants, within a few generations.
- From the viewpoint of information theory, the real essence of sexual (as opposed to asexual) reproduction is the perpetual mixing, and re-mixing, of the gene pool. Decisions about whether to reproduce, and with whom, are almost moot.*

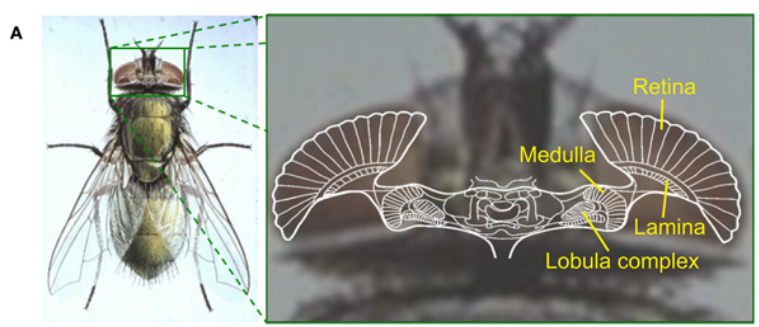
(*Famous 1984 song by Tina Turner, 'queen of rock-n-roll': *"What's Love Got To Do With It?"*)

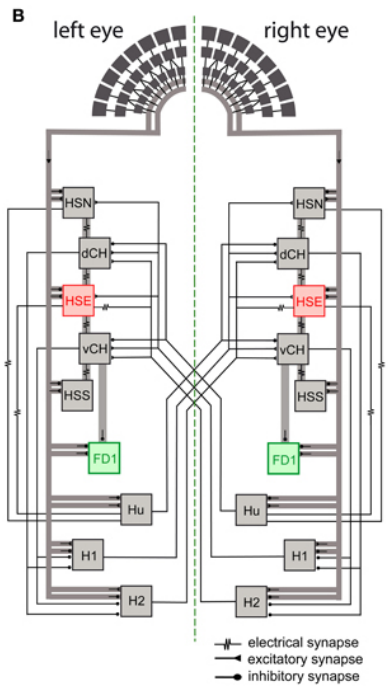
Information theory perspectives in neuroscience

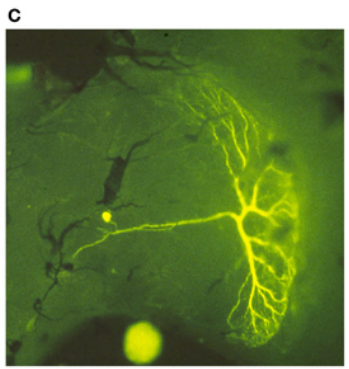
- At a synapse, what is the typical rate of information transfer?

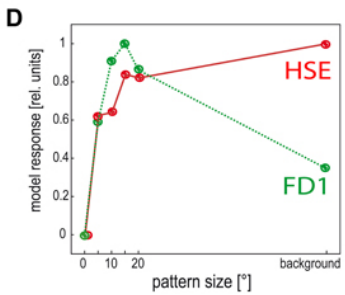


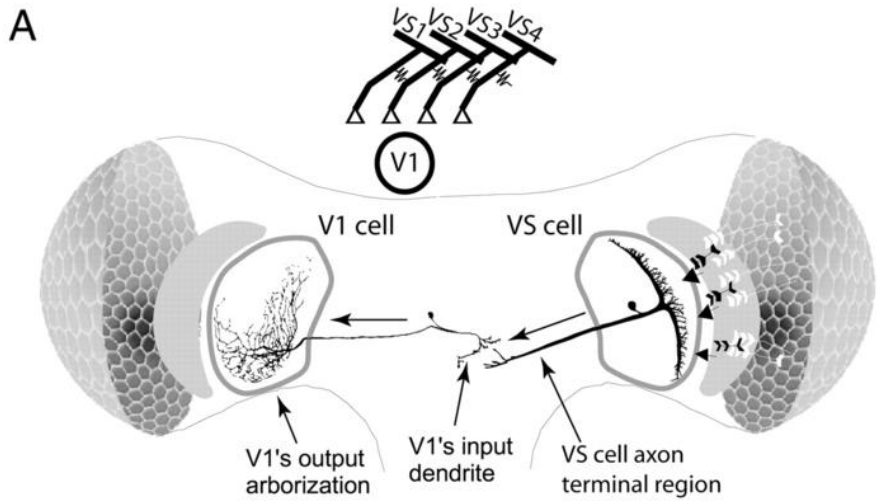
(Information theory perspectives in neuroscience, con't)

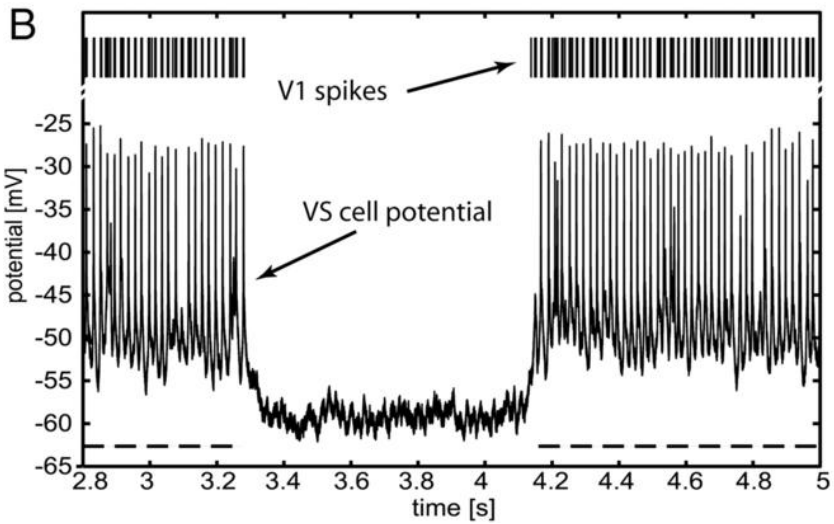








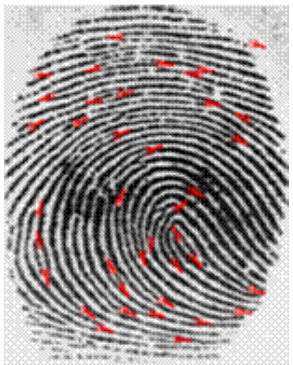




Information transmission rate by *blowfly* photoreceptors, estimated by SNR(ω) analysis at synapses using $C = \int \log_2(1 + \mathrm{SNR}(\omega))d\omega$, is up to 1,650 bits/second. (de Ruyter, van Steveninck, & Laughlin, *Nature,* 1996)

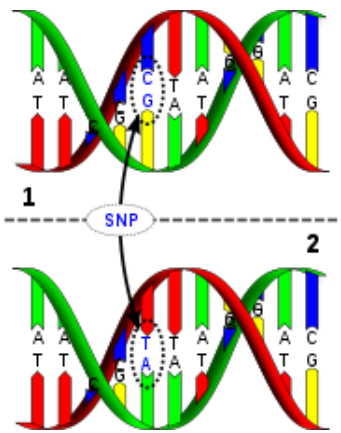
Biometric pattern recognition

Some examples of biometric methods and applications



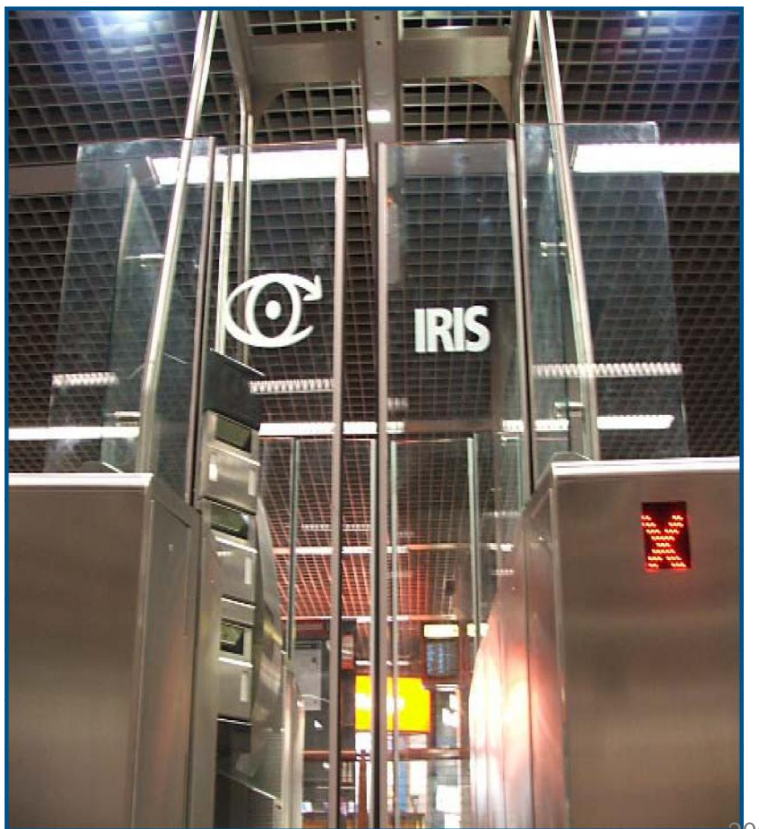
Forensics

"IrisKids" (US) missing children registration and identification

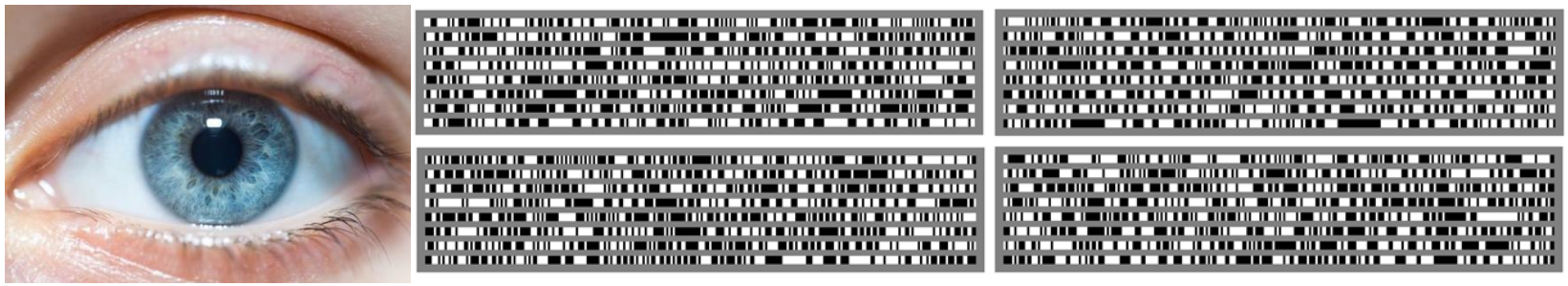


Face recognition ??

Home Office
Border & Immigration Agency

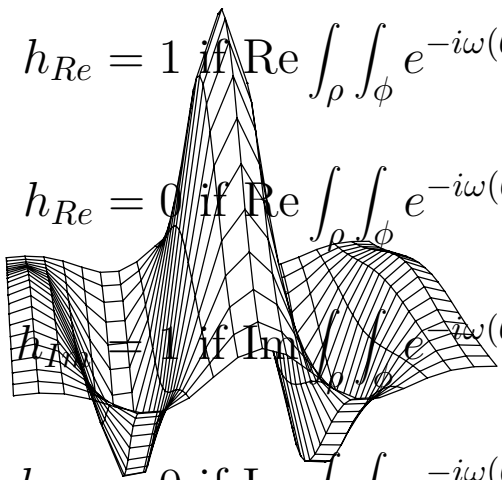


Entropy is the key to biometric collision avoidance



- Entropy measures the amount of random variation in a population: the number of possible patterns, and uniformity of their probabilities.
- The discriminating power of a biometric reflects its entropy; surviving large database searches without False Matches requires high entropy.
- The four IrisCodes shown above were computed from four different eyes. Bit values $\{0, 1\}$ are equiprobable, so 50% of bits should agree by chance. It is "statistically impossible" ($p < 10^{-6}$) for different persons to agree in more than about 67% of an eye's IrisCode bits.
- Non-heritable features make best biometrics. (About 1% of persons have an "identical" twin; but uninherited features are uncorrelated. For the same reason, anybody's R/L eye IrisCodes are uncorrelated.)

Setting Bits in an IrisCode by Wavelet Demodulation

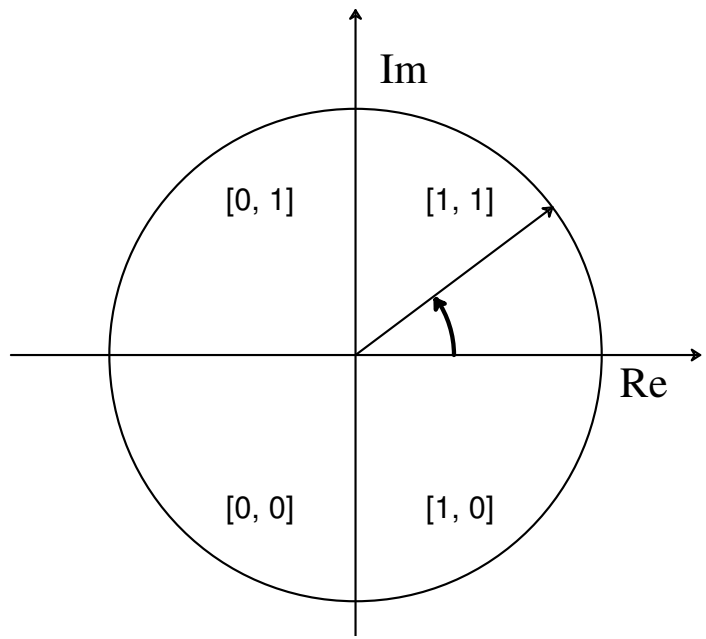


$$h_{Re} = 1 \text{ if } \mathrm{Re} \int_\rho \int_\phi e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} I(\rho,\phi)\rho d\rho d\phi \geq 0$$

$$h_{Re} = 0 \text{ if } \mathrm{Re} \int_\rho \int_\phi e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} I(\rho,\phi)\rho d\rho d\phi < 0$$

$$h_{Im} = 1 \text{ if } \mathrm{Im} \int_\rho \int_\phi e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} I(\rho,\phi)\rho d\rho d\phi \geq 0$$

$$h_{Im} = 0 \text{ if } \mathrm{Im} \int_\rho \int_\phi e^{-i\omega(\theta_0-\phi)} e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} I(\rho,\phi)\rho d\rho d\phi < 0$$



Why phase is a good variable for biometric encoding

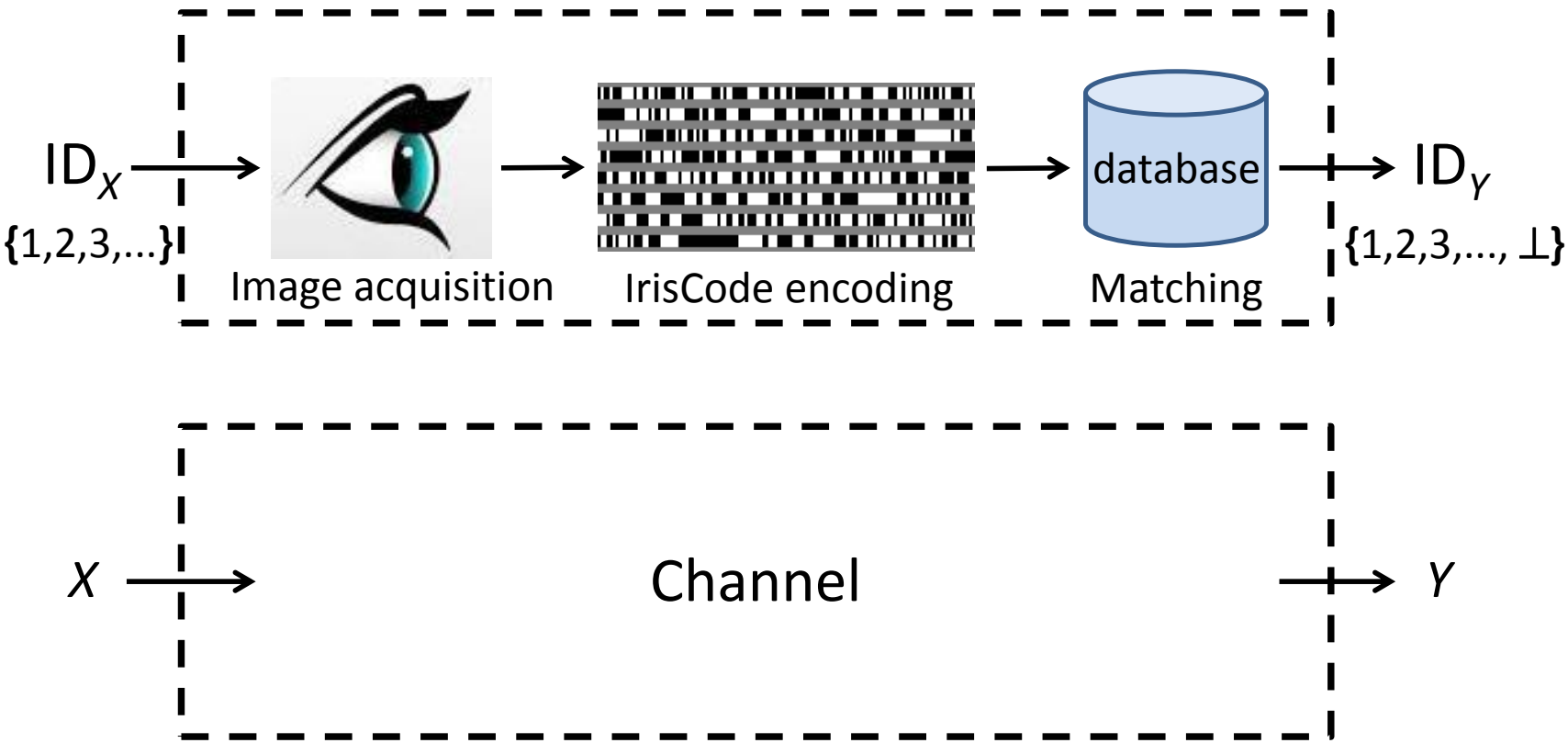
- Phase encodes structural information, independent of amplitude
- Phase encoding thereby achieves some valuable invariances
- Phase information has much higher entropy than amplitude
- In Fourier encoding terms, phase "does most of the work"
- Phase classification is equivalent to a clustering algorithm
- Phase can be very coarsely quantised into a binary string, such as the IrisCode (but a question remains: how finely to resolve phase? – into how many sectors should the phasor diagram be divided?)

Gabor wavelets encode phase naturally, but in a frequency-specific way.

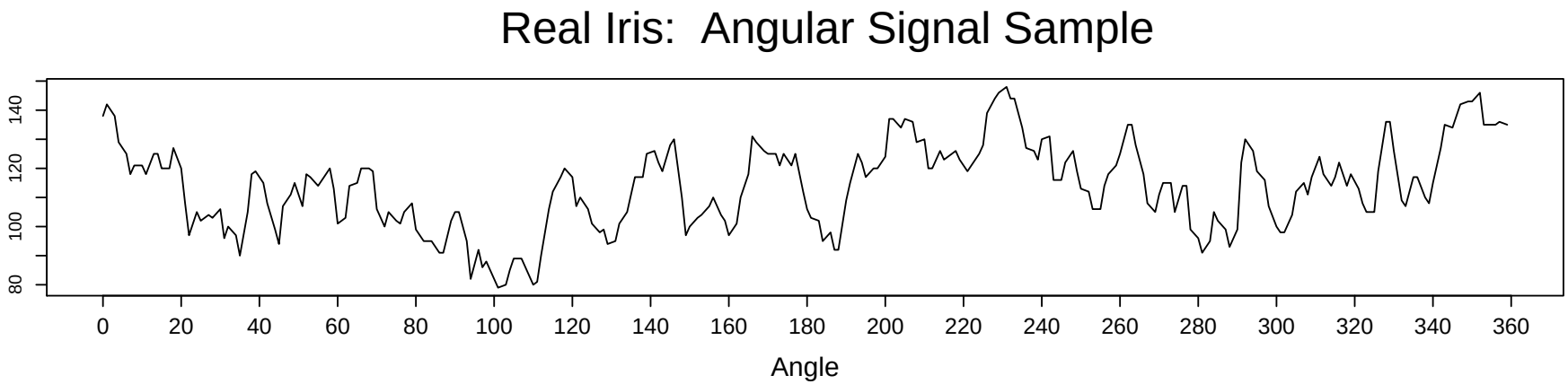
Alternatives exist that encode phase in a total way (independent of scale or frequency), such as the Analytic function (the signal $f(x)$ minus its Hilbert Transform $i\, f_{\mathrm{Hi}}(x)$ cousin): $f(x) - i\, f_{\mathrm{Hi}}(x)$, which is a complex function whose real and imaginary parts are "in quadrature".

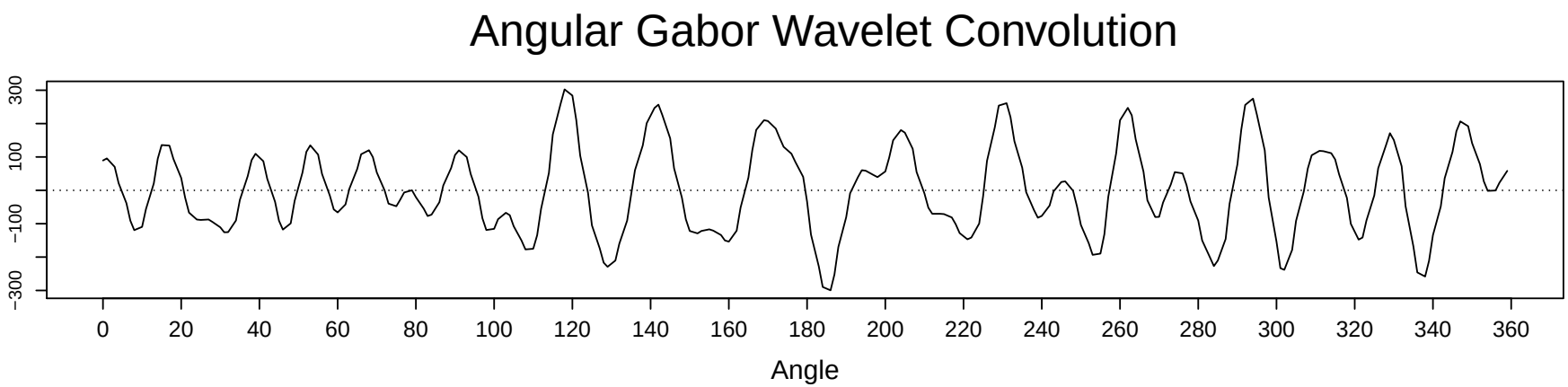
Information theory and the IrisCode

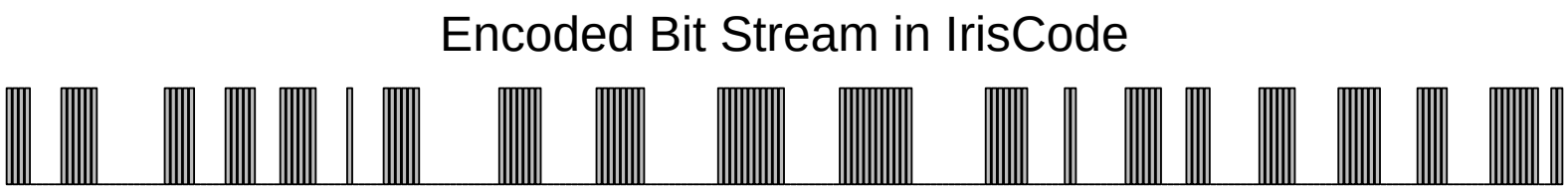
- We can regard the IrisCode itself as a kind of a 'channel'
- What is the channel capacity of the IrisCode, in bits per bit encoded?
- How much of this channel capacity is actually used by natural irises?



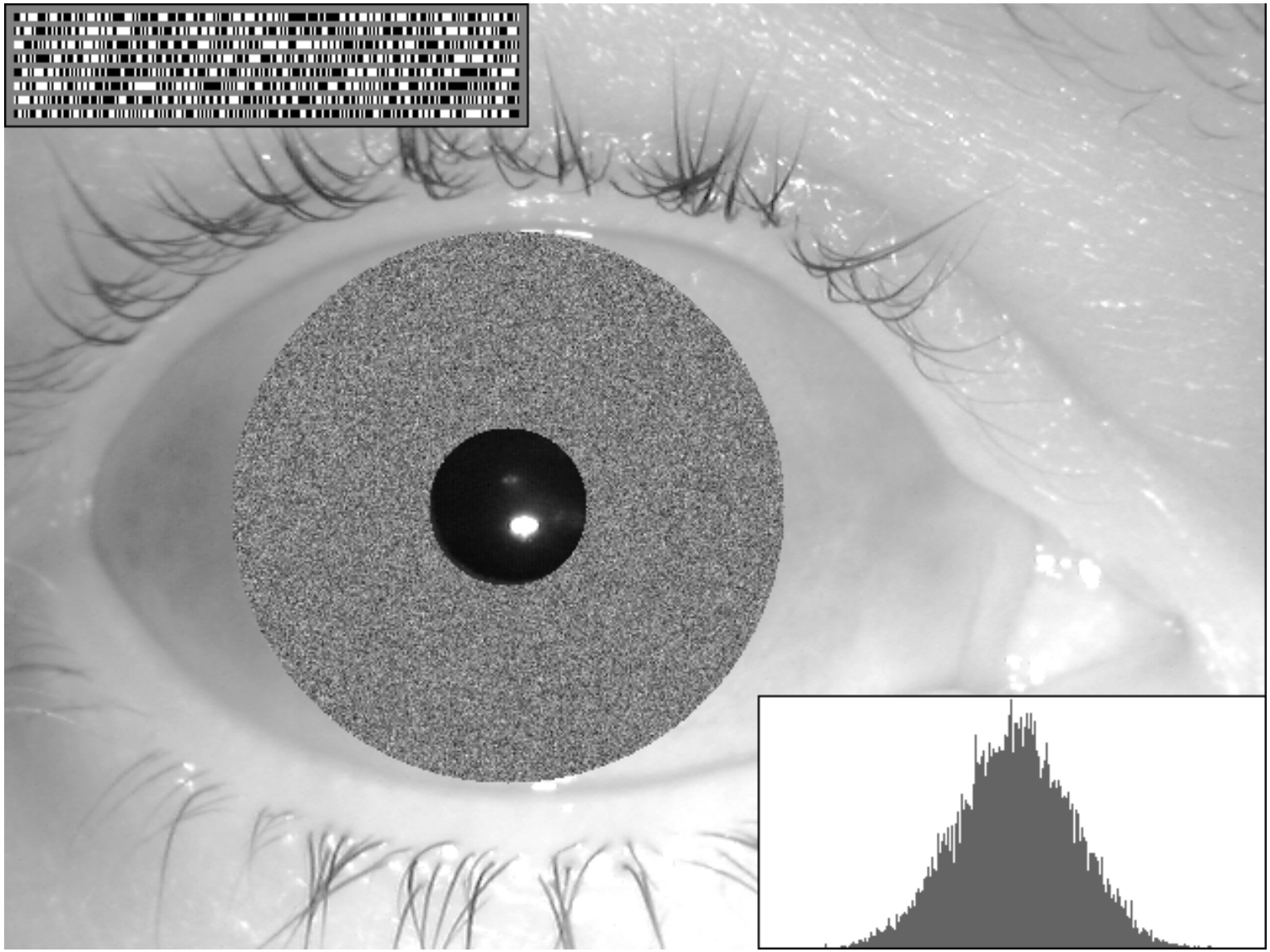
(Information theory and the IrisCode, con't)





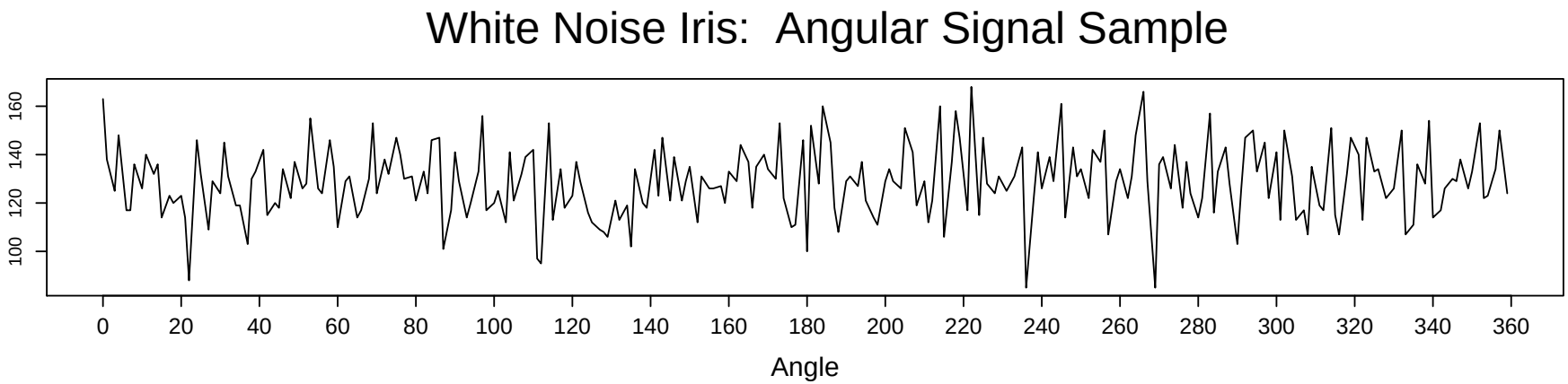


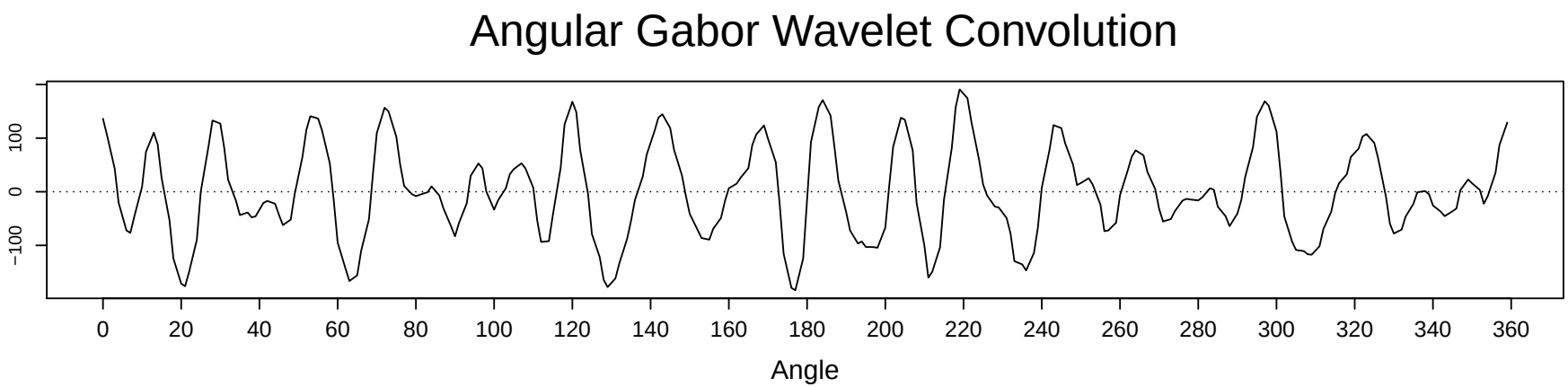
(Information theory and the IrisCode, con't)

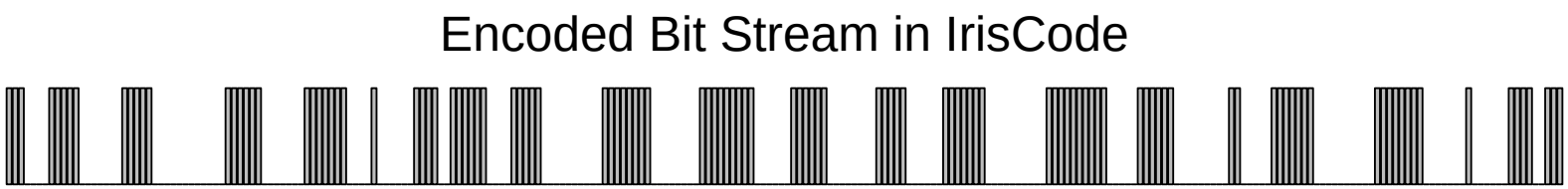


Synthesizing a *white noise iris* (Gaussian noise process for pixel values) in order to estimate the channel capacity of the IrisCode, as a channel

(Information theory and the IrisCode, con't)



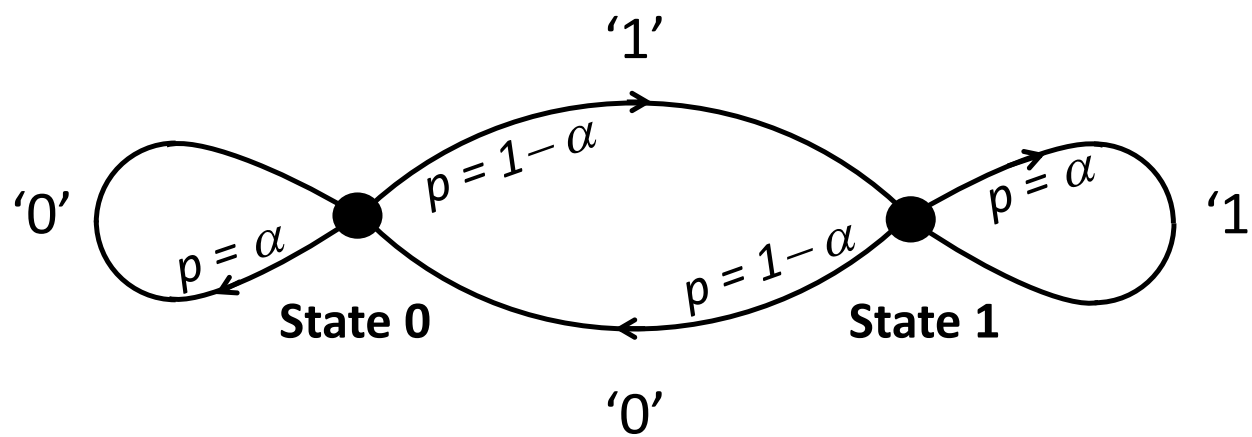




(Information theory and the IrisCode, con't)

IrisCodes computed both from the natural and from the white noise iris patterns are well-modelled by the bits emitted from a 'sticky oscillator' two-state Markov process, but with differing values of the parameter α

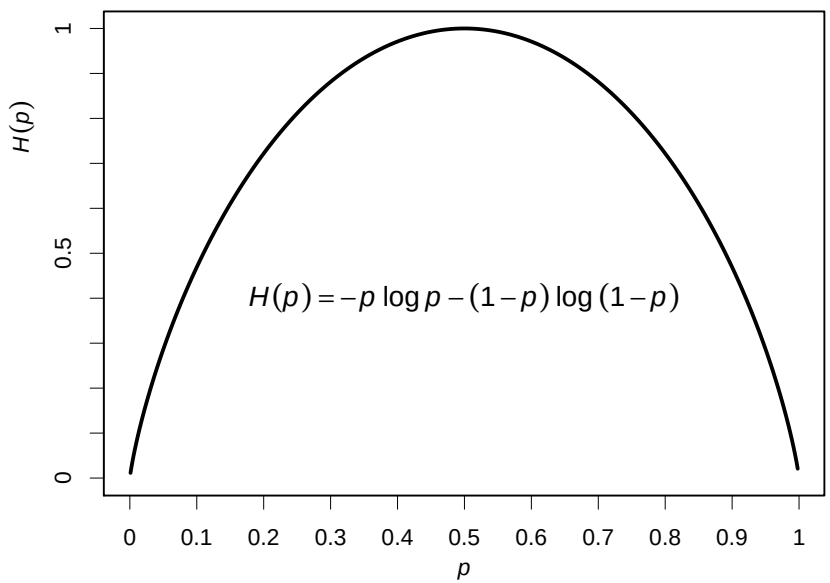
Two-State Markov Process



Entropy $H(\alpha) = -\alpha \log_2(\alpha) - (1-\alpha)\log_2(1-\alpha)$ bits, *per bit emitted*

(Information theory and the IrisCode, con't)

- Statistical distributions for IrisCodes computed from the white noise iris patterns are well-modelled using parameter $\alpha = 0.867$, giving a channel capacity of 0.566 bits per bit encoded in the IrisCode.
- This is smaller than 1 bit/bit (which would require $\alpha = 0.5$) because 2D Gabor wavelet encoders introduce correlations (both in amplitude and in phase) in their outputs. Correlations reduce entropy.
- Of this available IrisCode capacity, natural iris patterns actually generate only 0.469 bits of entropy per bit encoded ($\alpha = 0.90$).
- This further loss of entropy reflects the existence of anatomical correlations within natural iris patterns, especially radially.

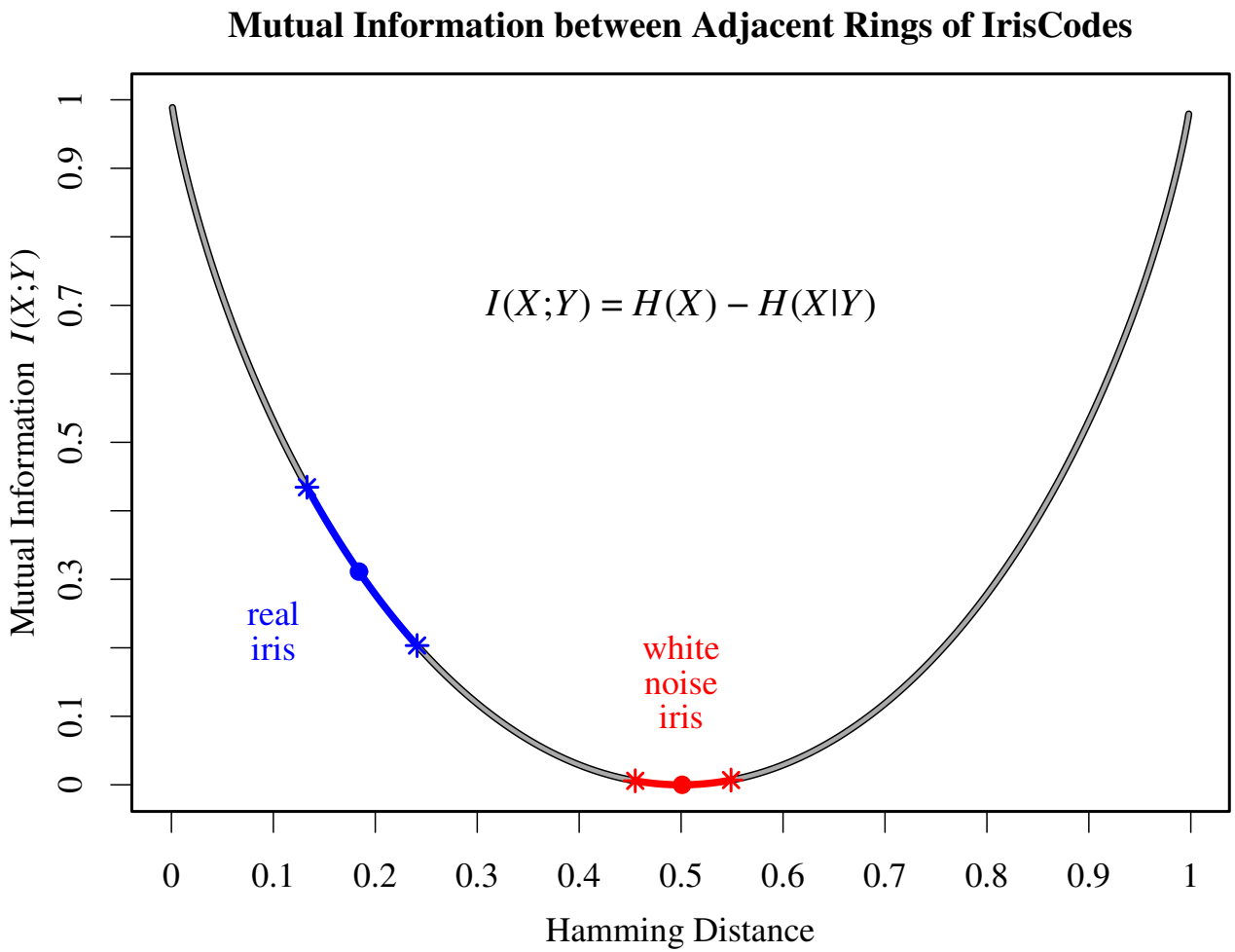


(Information theory and the IrisCode, con't)

Radial correlations are a striking aspect of natural human iris patterns.

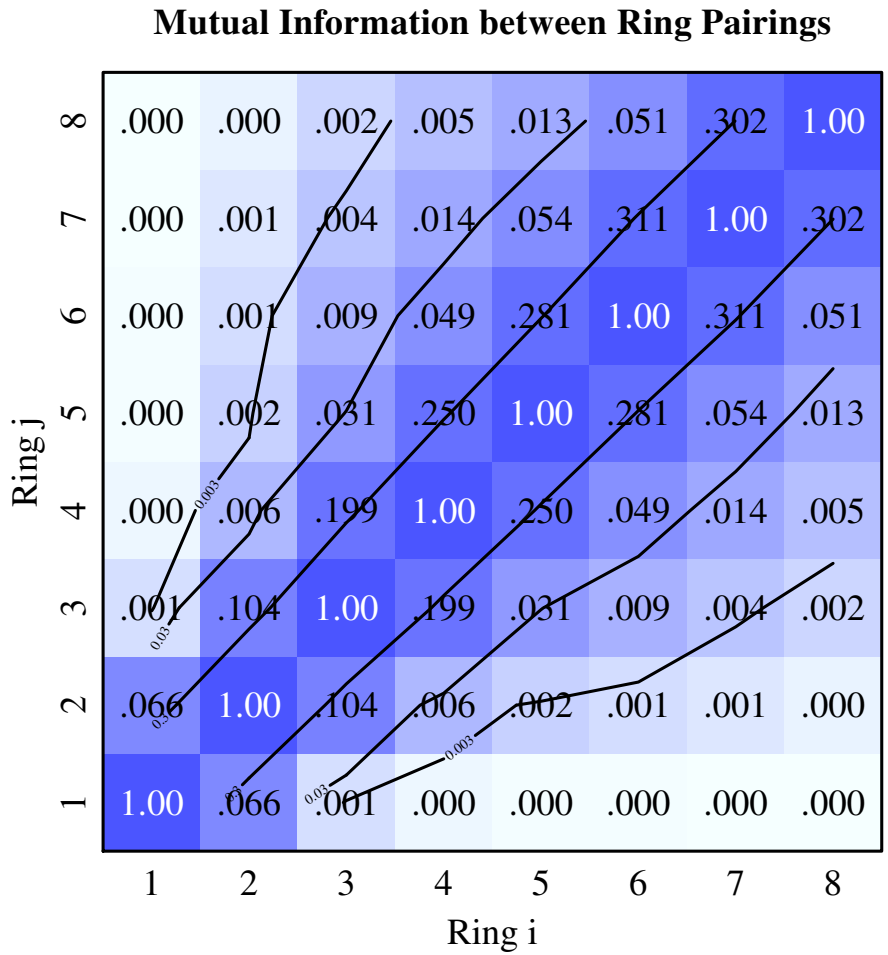
- But correlations reduce entropy, reducing the discriminating power of these biometric patterns. How much mutual information exists between concentric annuli (let's define eight "rings") of the iris?
- How does it compare with the case of "white noise" iris images?

(Information theory and the IrisCode, con't)



- The highest mutual information exists between rings 6 & 7 (among the 8 concentric rings), where on average it is 0.311 bits per bit pair.
- But mutual information is 0.0 bits per bit pair between rings for the case of white noise iris images, as expected.

(Information theory and the IrisCode, con't)



- It is informative to see how far the mutual information extends radially, in a large population of natural human iris patterns.
- Comparing non-adjacent rings, the mutual information attenuates quickly back to 0.0 when radial distance skips across two or more.

Why IrisCode matching is so fast, parallelisable, scalable

Bit streams A and B are data words of two IrisCodes.
Bit streams C and D are their respective mask words.

(data) A	1	0	0	1	0	1	1	0	0	0	1	0	1	1	1	0	···
(data) B	0	1	0	1	1	1	0	0	1	0	0	1	0	1	1	0	···
A⊕B	1	1	0	0	1	0	1	0	1	0	1	1	1	0	0	0	···
(mask) C	1	1	1	0	1	0	1	1	0	0	1	1	1	0	1	1	···
(mask) D	0	1	1	1	1	1	0	1	0	1	1	1	0	1	1	1	···
C∩D	0	1	1	0	1	0	0	1	0	0	1	1	0	0	1	1	···
(A⊕B)∩C∩D	0	1	0	0	1	0	0	0	0	0	1	1	0	0	0	0	···

Note that for these 16 bit chunks, only 8 data bits were mutually unmasked by C∩D.

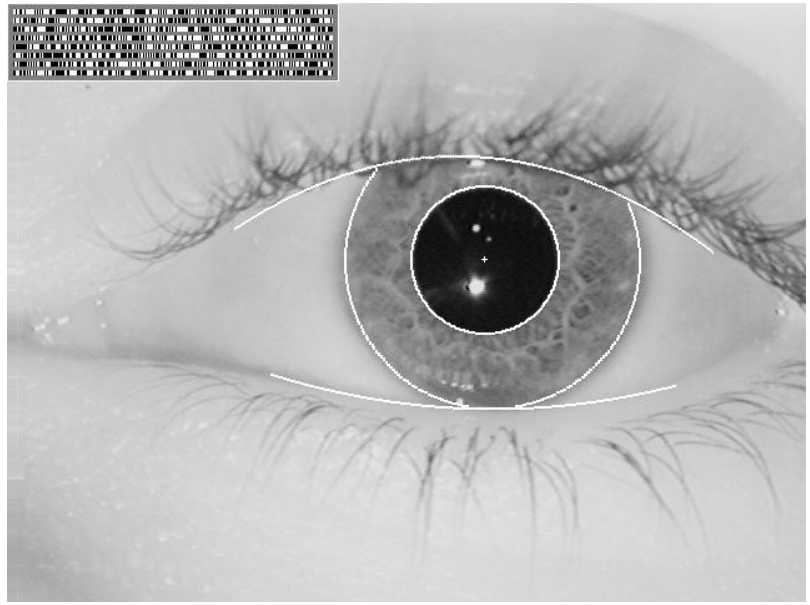
Of those 8, they agreed in 4 and disagreed in 4, so raw Hamming distance is $4/8 = 0.5$ which is typical for comparisons between "Impostors" (unrelated IrisCodes).

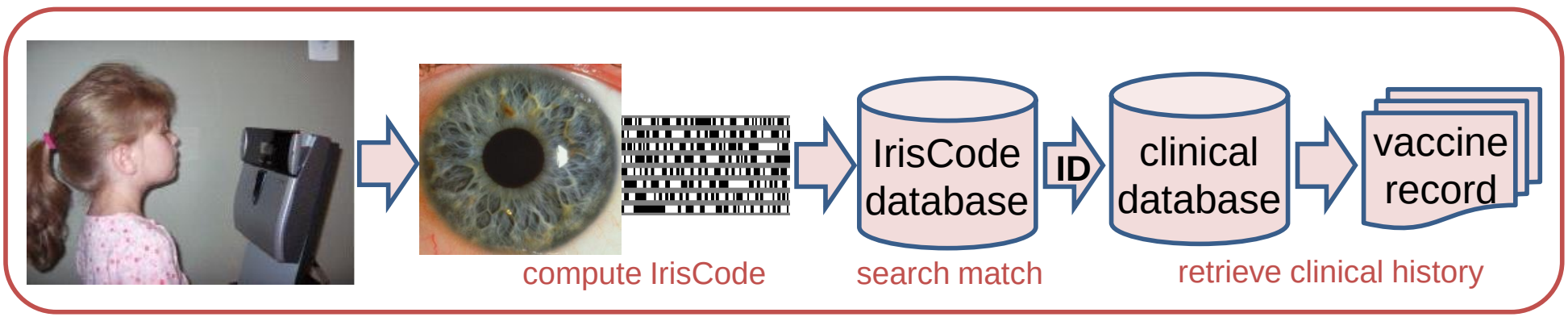
Bit-parallel logic programming allows all of this to be done in a single line of C-code, operating on word lengths up to the word-length of the CPU (*e.g.* 64 bits at once):

```
result = (A ^ B) & C & D;
```

Each of the 3 logical parallel operators executes in a single "clock tick" (*e.g.* at 3 GHz).

(Gates Foundation study): Using Iris Recognition to Retrieve Vaccination and Healthcare Records





UNIVERSITY OF CAMBRIDGE