Transform Methods & Signal Processing

lecture 03

Matthew Roughan
<matthew.roughan@adelaide.edu.au>

Discipline of Applied Mathematics
School of Mathematical Sciences
University of Adelaide

July 27, 2009

Discrete signals

In theory there is no difference between theory and practice. In practice there is.

Yogi Berra

This lecture considers real signals (which are almost all discrete) and the Discrete Fourier Transform (DFT), and its properties.

Discrete signals

Real signals (these days) are discrete

- Moore's law (speed of digital hardware increases by a factor of two every 18 months, or the number of transistors on a chip doubles, or the cost halves).

  "Cramming more components into integrated circuits", Gordon E. Moore, Electronics, Vol. 38, No. 8, April, 1965.

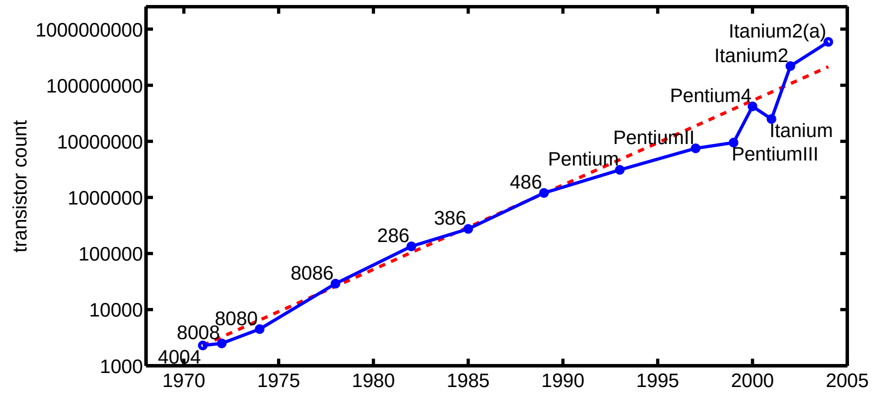
- Easier/cheaper to use standard DSP solution.

  e.g. CD players — we can get nominally better results from a LP record, and a really good player, but CD's cost orders of magnitude less for **almost** indistinguishable results.

- If it isn't cheap enough today, it will be in a year.

Moore's Law

Moore's law: the speed of digital hardware increases by a factor of two every 18 months, or the number of transistors on a chip doubles, or the cost halves.



Actually looks more like a factor of 2 every 2 years.

Intel's pages on Moore's law:
`http://www.intel.com/technology/mooreslaw/index.htm`
`ftp://download.intel.com/research/silicon/moorespaper.pdf`

Other links to Moore's law:
`http://en.wikipedia.org/wiki/Moore's_law`
`http://www.thocp.net/biographies/papers/moores_law.htm`
`http://www.firstmonday.org/issues/issue7_11/tuomi/`
`http://www.hyperdictionary.com/computing/moore's+law`
`http://www.physics.udel.edu/wwwusers/watson/scen103/intel.html`
`http://www.ziplink.net/~lroberts/Forecast69.htm`

Gates's law

Gates's Law: The speed of software halves every 18 months.

Gates's law does not apply to DSPs (they use small embedded OSes).

Parkinson's Law of Data: Data expands to fill the space available for storage

Parkinson's law of data **does** typically apply. As chips get faster, we sample at higher resolution, and faster sampling rates...

Real signals

> In theory there is no difference between theory and practice. In practice there is.
>
> Yogi Berra

Real data is

- finite (integrals convergence much easier)
- discrete time
- discrete valued

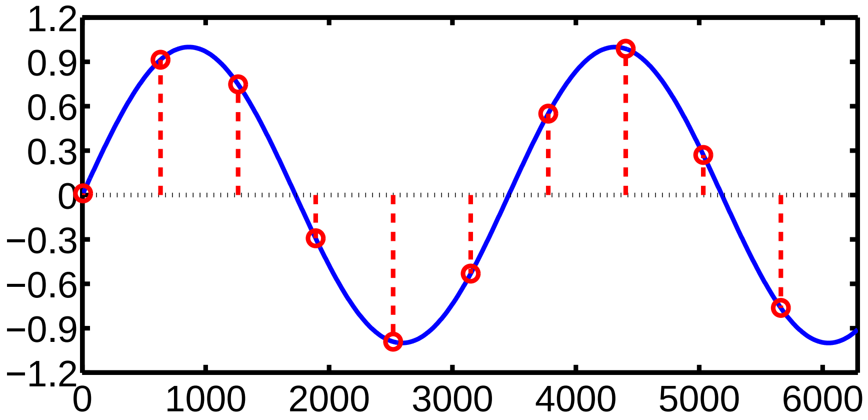
Gate's law isn't entirely a joke, e.g. see
`http://hubpages.com/hub/_86_Mac_Plus_Vs_07_AMD_DualCore_You_Wont_Believe_Who_Wins`

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.5/80

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.6/80

Discrete time

- Real signals are **discrete-time**
- We can **sample** a continuous function to get a discrete approximation

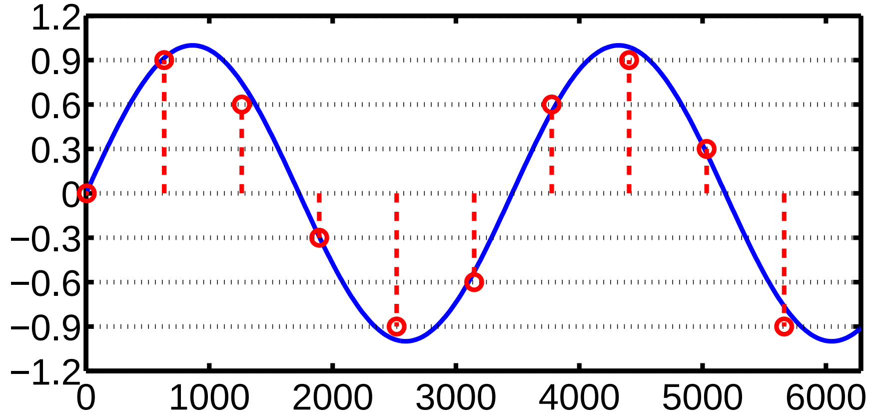


The x-axis is discretised, but the y values are still exact. We have sampled the function at a set of sample points.

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.7/80

Quantization: discrete-value

- Real signals are **discrete-valued**
- Analogue to Digital conversion: sample in time, and quantise

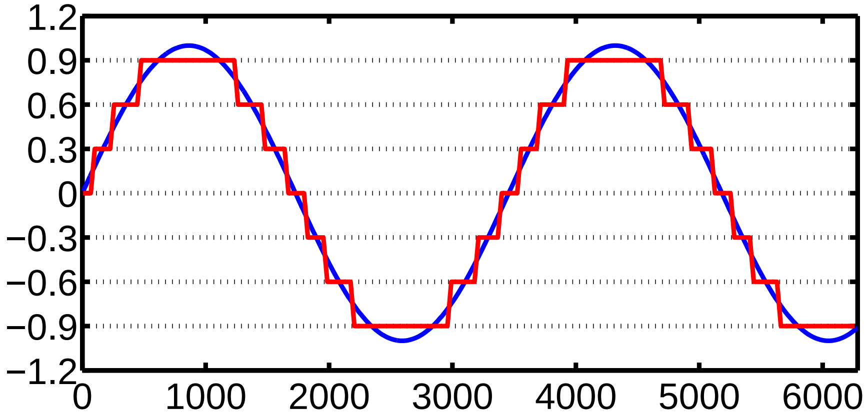


Now the y-axis is also discretised, so now we only have an approximation of the function, recorded only at certain time-points called sample points.

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.8/80

Approximation

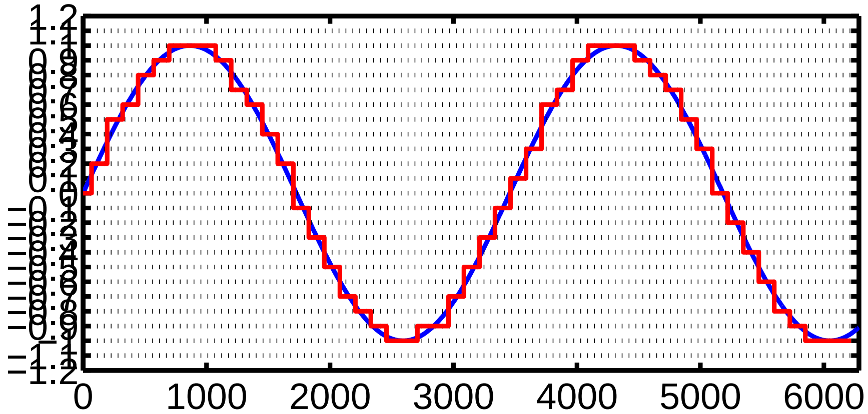
- Faster sampling => better approximation
- More details later



Given a set of sample points, we can try to reconstruct the original continuous signal in a number of ways (this is called interpolation). The illustration is a simple (but crude) method where we assume the signal takes the value of the sample until we get to the next sample. This is sometimes called nearest neighbor, or piecewise constant interpolation.

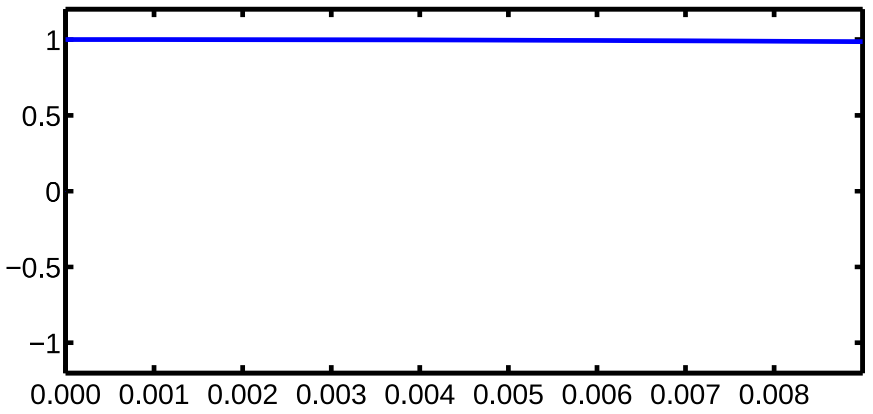
Approximation

- Finer quantization => better approximation
- More details later



Approximation

- Longer data sets => better approximation
- More details later



Sampling

Sampling produces a new time series, with discrete index, e.g.

$$x(n) = f(nt_s)$$

where t_s is the **sampling interval**

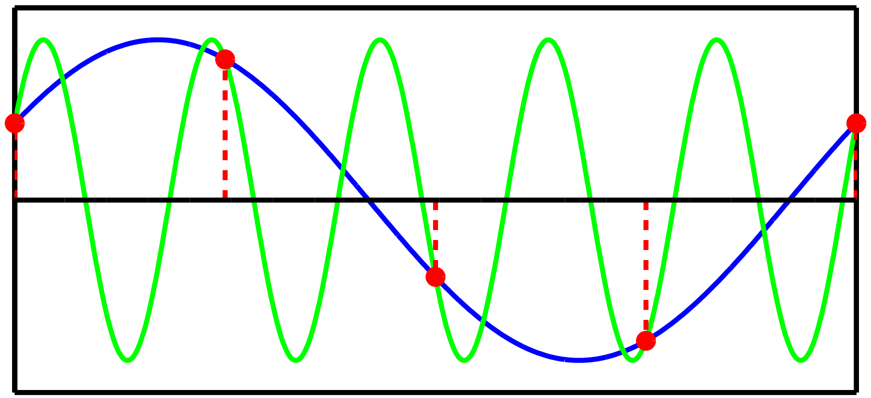
The **sampling frequency** is $f_s = 1/t_s$.

e.g. sampling frequency for CDs is 44.1 kHz

The plot in the example shows a segment of a cosine function. However, over the range displayed the function looks constant, or maybe there is a small linear decrease.

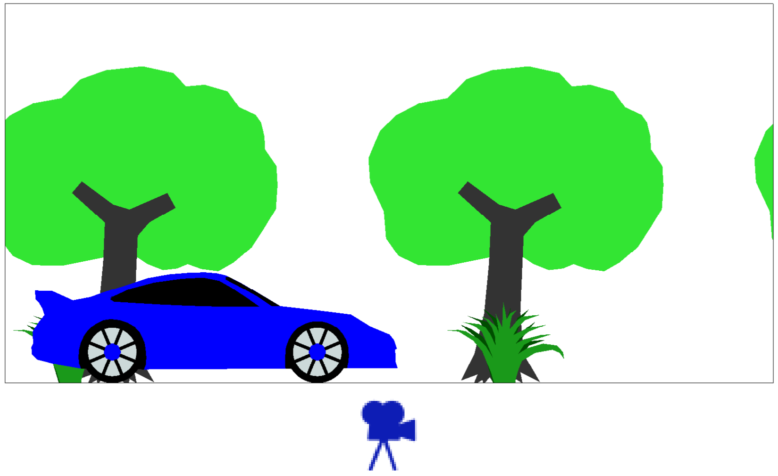
Aliasing

A critical issue for sampling is **aliasing**



Samples might be caused by different underlying signals.

Another example of aliasing



The plot shows three different sinusoids, but each produces exactly the same sample values at the points shown. This type of ambiguity is called aliasing (think of Superman's alias — Clark Kent – they are really the same person).

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.14/80

Aliasing: time domain view

- Signal with frequency f_0, given by $f(t) = \sin(2\pi f_0 t)$
- Sampling interval t_s, and sampling frequency $f_s = 1/t_s$.
- Sampled signal is $x(n) = f(nt_s) = \sin(2\pi f_0 n t_s)$
- We can always add $2\pi m$ (where m is an integer) to a sin function without impact, e.g.

$$\begin{aligned} x(n) &= \sin(2\pi f_0 n t_s) \\ &= \sin(2\pi f_0 n t_s + 2\pi m) \\ &= \sin\left(2\pi\left[f_0 + \frac{m}{nt_s}\right] nt_s\right) \\ &= \sin(2\pi[f_0 + f_s k]\, nt_s) \text{ where } m = kn. \end{aligned}$$

So there is an ambiguity in $x(n)$ about frequencies $f_0 + f_s k$ for integer k.

Aliasing: frequency domain view

Consider a **delta train** or **Dirac comb** defined by

$$d(t) = \sum_{n=-\infty}^{\infty} \delta(t-n)$$

We can consider sampling of a function $f(t)$ to be equivalent to taking the product with a delta train, e.g.

$$x(t) = d(t/t_s) f(t)$$

From the convolution, and the duality theorems, we can see that the FT of $x(t)$ will be the convolution of the FTs of $d(t)$ and $f(t)$.

The FT of the delta train is $\mathcal{F}\{d(t/t_s)\} = |t_s| d(t_s s)$

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.15/80

Poisson summation formula sketch of why $\mathcal{F}\{d(t/t_s)\} = |t_s| d(t_s s)$

$$\begin{aligned} D(s) &= \mathcal{F}\{d(t)\} \\ &= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(t-n)\, e^{-i2\pi st}\, dt \\ &= 1 + 2\sum_{n=1}^{\infty} 0.5\left[e^{-i2\pi sn} + e^{i2\pi sn}\right] \\ &= 1 + 2\sum_{n=1}^{\infty} \cos(2\pi sn) \end{aligned}$$

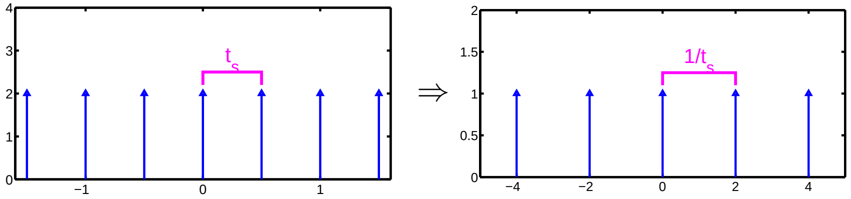
For s an integer, each term in the sum is 1, and so the sum diverges.
For s, not an integer, the sum looks like the integral $\int_{-\pi}^{\pi} \cos(x)\, dx = 0$.

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.16/80

Delta train

A train of delta functions $d(t/t_s) = \sum_{n=-\infty}^{\infty} \delta(t/t_s - n)$ has Fourier transform which is also a delta train, e.g.

$$\mathcal{F}\{d(t/t_s)\} = |t_s| d(t_s s)$$



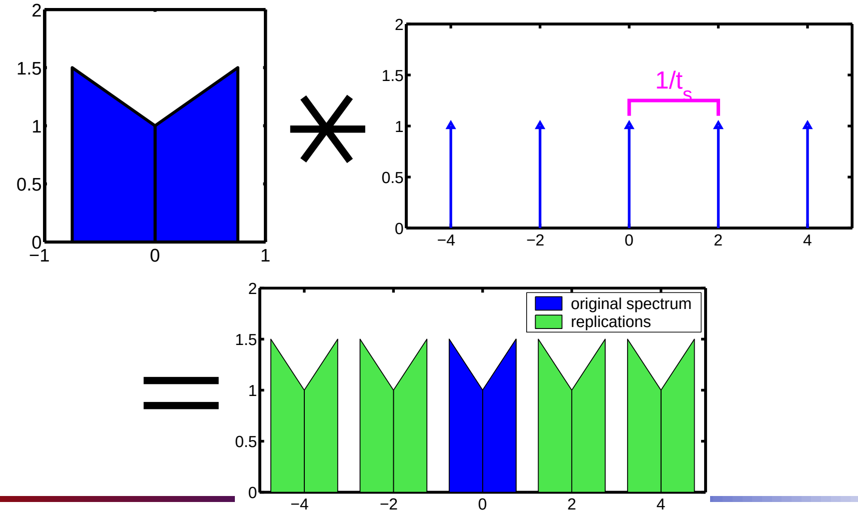
The above signal might variously be called a delta train, a delta comb, a Dirac comb, a Dirac train or some other variant. Comb comes from the shape (like a comb), whereas train comes from the fact that we have a train of deltas in sequence.

Bracewell also uses the Cyrillic letter shah, Ш, because of its shape.

Aliasing: frequency representation

$x(t) = d(t/t_s) f(t) \Rightarrow X(s) = d(t_s s) * F(s)$

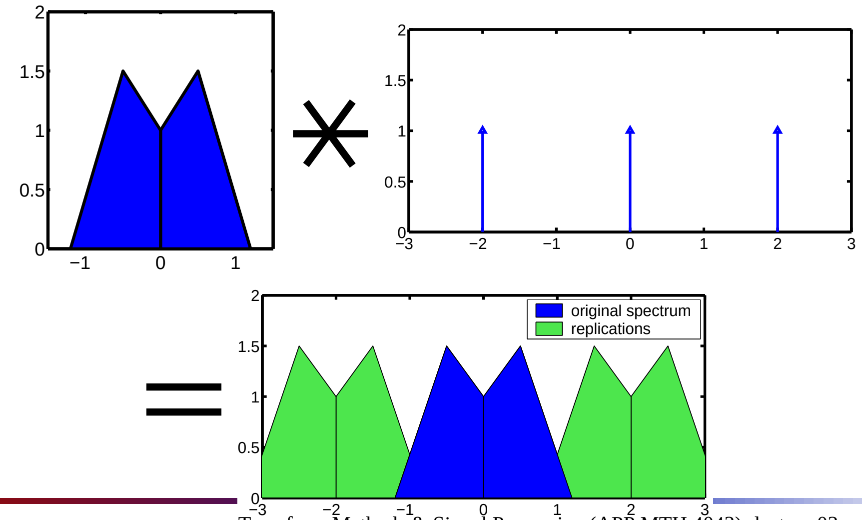
Convolution of a delta train with a function looks like:



Because the blocks aren't overlapping, we can use the fact that we know the blue spectrum is "band limited" and restrict out attention in the Fourier domain to just this part.

Aliasing: frequency representation

$x(t) = d(t/t_s) f(t) \Rightarrow X(s) = d(t_s s) * F(s)$
Convolution of a delta train with a function looks like:



Nyquist sampling theorem

Assume the spectrum of the signal is zero above a critical frequency f_c. We call this the bandwidth of the signal.
For sampling frequencies $f_s > 2f_c$, the spectra above won't overlap. If $f_s < 2f_c$ aliasing becomes a problem.

- the critical sampling rate referred to by, e.g. the Nyquist rate, or Shannon (1949) or Whittaker (1935) sampling theorem.
- the sampling frequency must be greater than twice the highest frequency present in the signal
- need to bandlimit the input signal before sampling
- bandwidth does not need to be centered on zero Hz.

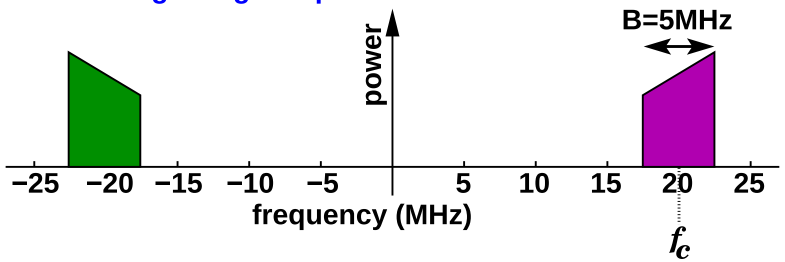
Even though the input signal is band limited, the resulting spectra overlap, because the maximum frequency $f_c > 1 = f_s/2$. The overlapping is a problem which we must avoid.

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.19/80

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.20/80

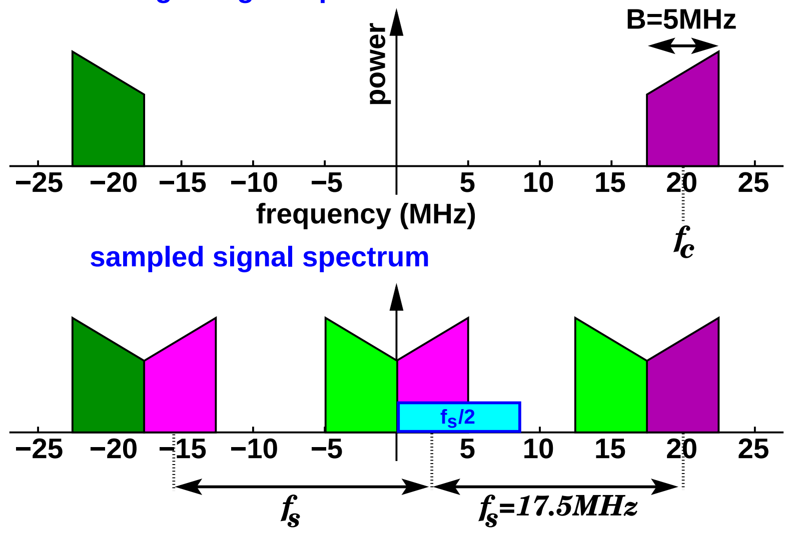
Example

Analogue signal with central frequency 20 MHz, and 5 MHz bandwidth.



To include entire spectrum, we need to sample at $2 \times 22.5 = 45$ MHz.

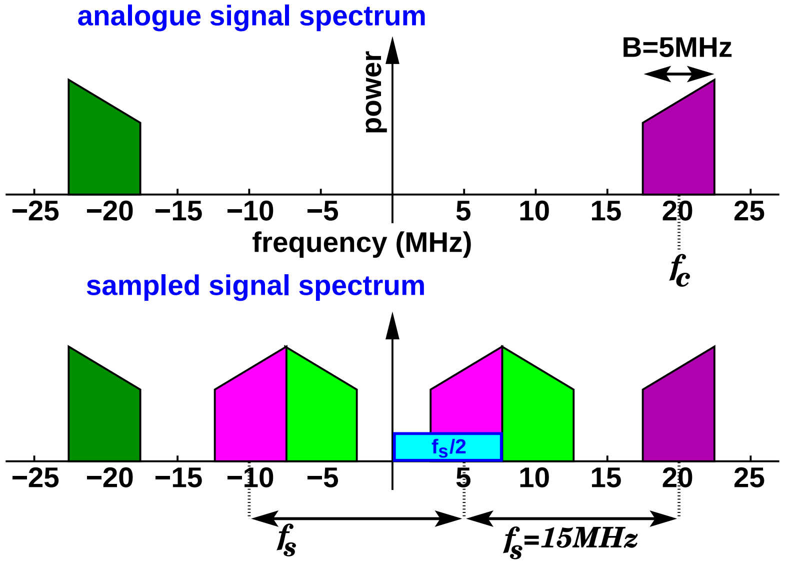
Example: Sample at 17.5 MHz



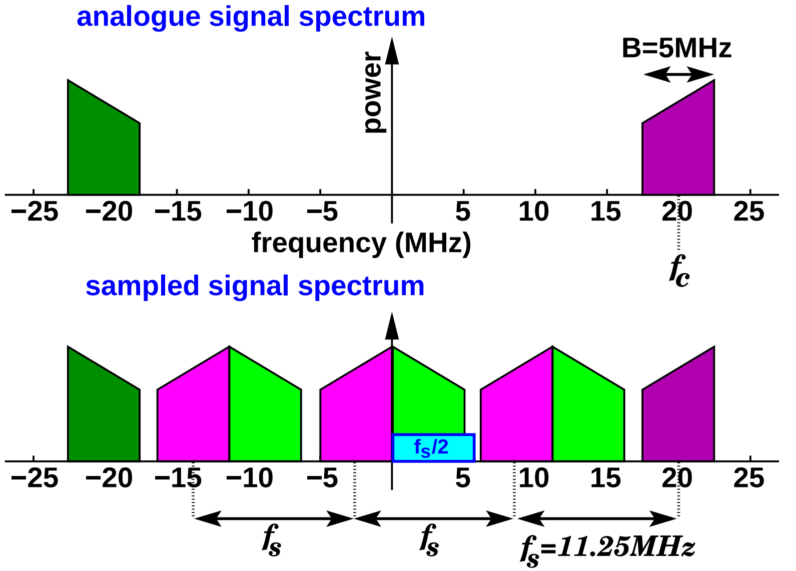
Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.21/80

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.22/80

Example: Sample at 15 MHz



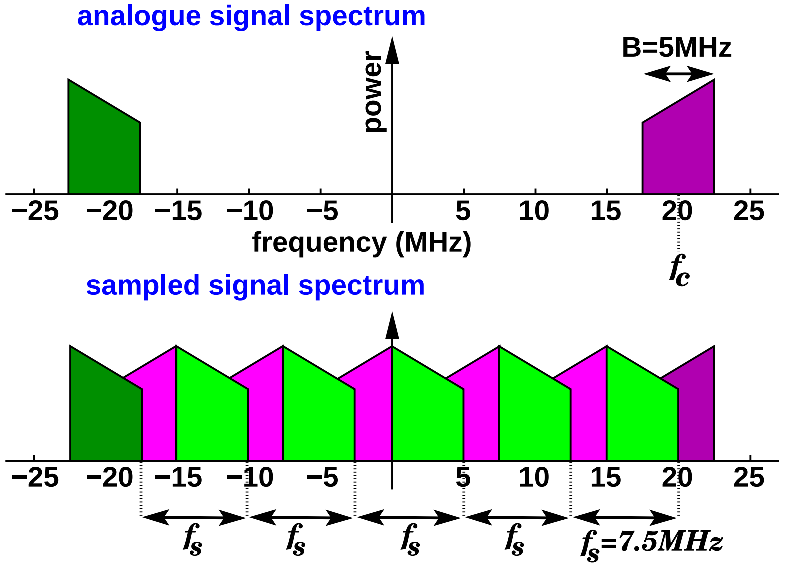
Example: Sample at 11.25 MHz



Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.23/80

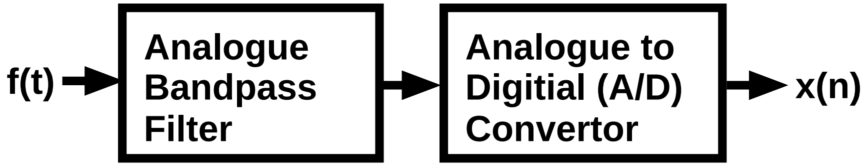
Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.24/80

Example: Sample at 7.5 MHz



Bandlimiting

When sampling from real signals, one **must** bandlimit the input!



Shouldn't push the boundaries with sampling, and filters

- analogue filter might not be ideal
- sample clock generation instabilities
- imperfections in A/D quantization.

Hence, include guard bands around bandwidth of interest.

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.25/80

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.26/80

Some more sampling theory

Shannon Sampling Theorem:
"If a function $f(t)$ contains no frequencies higher than W cycles per second, it is completely determined by giving its ordinates at a series of points spaced $(1/2W)$ seconds apart."

- so we can reconstruct $f(t)$ from its samples
  - if the signal is bandlimited
  - samples spaced $(1/2W)$
  - Hence Nyquist result

Shannon theorem

Proof sketch: Assume function is bandlimited so $F(s) = 0$ for $|s| > W$, then the IFT is

$$f(t) = \int_{-\infty}^{\infty} F(s) e^{i2\pi st} \, ds = \int_{-W}^{W} F(s) e^{i2\pi st} \, ds$$

If instead, we make, F periodic, with period $2W$ then we can find a Fourier series for it, e.g.

$$F(s) = \sum_{n=-\infty}^{\infty} A_n e^{i\pi ns/W}$$

where,

$$A_n = \frac{1}{2W} \int_{-W}^{W} F(s) e^{-i\pi ns/W} \, ds = \frac{1}{2W} f\left(\frac{n}{2W}\right)$$

See:
Claude Shannon, "Communications in the presence of noise", Proc.IRE, 37, pp.10–21, 1949.
H.Nyquist, "Certain topics in telegraph transmission theory", AIEE Trans., 47, pp.617–644, 1928.

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.27/80

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.28/80

Shannon theorem

Proof sketch:
We can represent $F(s)$ perfectly with the Fourier series coefficients A_n, but these are just proportional to the function sampled at uniform intervals, e.g. $A_n \propto f\left(\frac{n}{2W}\right)$.

Hence, the samples completely define the FT F, and hence the function f. $\square$

Shannon interpolation

Reconstruction of original signal from IFT

$$
\begin{aligned}
f(t) &= \int_{-W}^{W} F(s) e^{-i2\pi st}\, ds \\
&= \int_{-W}^{W} \sum_{n=-\infty}^{\infty} A_n e^{i\pi ns/W} e^{i2\pi st}\, ds \\
&= \sum_{n=-\infty}^{\infty} A_n \int_{-\infty}^{\infty} r(s/2W) e^{i2\pi s(-t+n/2W)}\, ds \\
&= \sum_{n=-\infty}^{\infty} 2WA_n \int_{-\infty}^{\infty} r(-s) e^{i2\pi s(2Wt-n)}\, ds \\
&= \sum_{n=-\infty}^{\infty} f\left(\frac{n}{2W}\right) \operatorname{sinc}(2Wt-n)
\end{aligned}
$$

The last step follows because

- The IFT of $r(s)$ is $sinc(t)$
- When $t = m/2W$ for m an integer, then $2Wt - n$ is also an integer $m - n$. Note that $\operatorname{sinc}(m-n) = \delta_{mn}$.
- Hence at those points we get

$$f(m/2W) = \sum_{n=-\infty}^{\infty} 2WA_n \operatorname{sinc}(2Wt-n) = \sum_{n=-\infty}^{\infty} 2WA_n \delta_{mn} = 2WA_m$$

Shannon interpolation

Assume we sampled at the Nyquist rate, i.e. $f_s = 2W$, or $t_s = 1/2W$, then the sample points would be

$$f\left(\frac{n}{2W}\right)$$

The summation

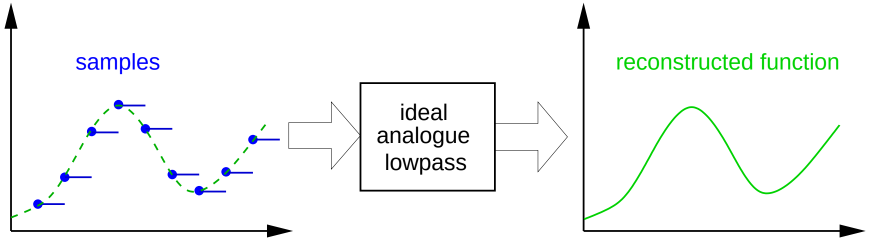
$$f(t) \quad = \quad \sum_{n=-\infty}^{\infty} f\left(\frac{n}{2W}\right) \operatorname{sinc}(2Wt - n)$$

The above formula represents a "convolution" of the sampled signal with a sinc function. We will learn about convolutions later, but note that this convolution acts to (perfectly) filter out high frequencies.

Digital to Analogue converter

Interpretation

- convolution with sinc
- equivalent to ideal analogue low-pass filter



- this is essentially what a Digital to Analogue converter tries to do
- have to build analogue filter — hard to make it ideal

A Digital-to-Analogue converter is sometimes abbreviated to a D-to-A converter, or DAC.

The samples are read into the DAC, which must first convert these into continuous voltages. This is typically done using a type of "store and hold" operation. The sample value is held until the next sample, so that the output is a piece-wise constant curve (that looks a bit like a staircase). The mechanism to perform this step is sometimes called a latch, because it latches onto values.

The green (dashed) curve shows the original signal, which has been sampled at the blue dots. The new analogue signal is represented by the blue line segments.

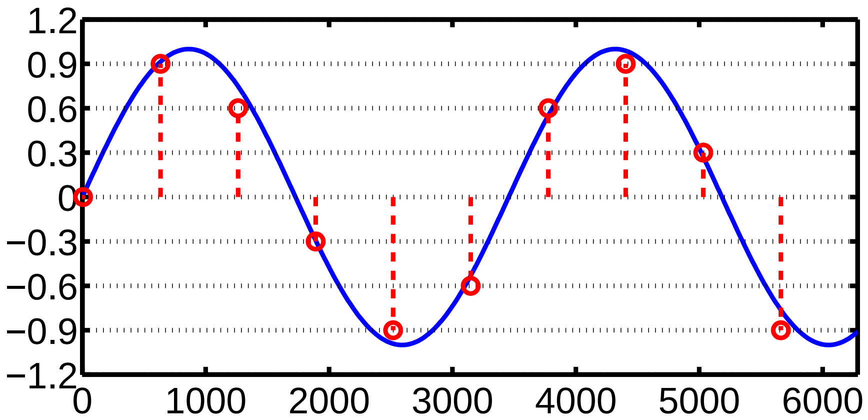
Obviously, the staircase curve is only a crude approximation to the original smooth curve. We get back the original curve by convolving filtering the signal with a (preferably) ideal low-pass filter, that smooths the curve, and removes the nasty harmonics introduced by the steps.

Perfect analogue filters are unrealizable. Even good analogue filters are expensive (compared to digital filters) so often digital tricks (e.g. upsampling) are used before the DAC, to make this step easier.

For an intuitive description of some of the issues see
`http://www.audioholics.com/education/audio-formats-technology/exploring-digital-audio-myths-and-reality-part-1`.

Quantization: discrete-value

Quantise real number values so they can be represented on a computer (or DSP) in a binary format. This is the essence of "digital" technology.



Dynamic range

Dynamic range expresses the range of values we can represent in our digital format, e.g.

- assume fixed point representation with b bits.
- largest value representable is $(2^b - 1)\delta$
- smallest value representable is δ
- dynamic range $= 20\log_{10}\frac{(2^b-1)\delta}{\delta} \simeq b20\log_{10}2 = 6.02b$ dB
- 6 dB per bit

CD's use 16 bit fixed point, so the dynamic range of a CD recorded sound is approximately $16 \times 6 = 96dB$.
Compare to somewhere between 50-70 dB for LPs, depending on the quality of the pressing.

Some background: all **digital** data is represented as numbers, e.g. the data on a CD is represented as numbers. The numbers are usually represented in some binary format, for instance, we might write a number in terms of binary "bits" where each bit is either 0 or 1

$$\begin{aligned} 0 &= 000 \\ 1 &= 001 \\ 2 &= 010 \\ 3 &= 011 \\ 4 &= 100 \\ 5 &= 101 \\ 6 &= 110 \\ 7 &= 111 \end{aligned}$$

We would often use an extra bit at the start to indicate sign, e.g.

$$\begin{aligned} 4 &= 0100 \\ -4 &= 1100 \end{aligned}$$

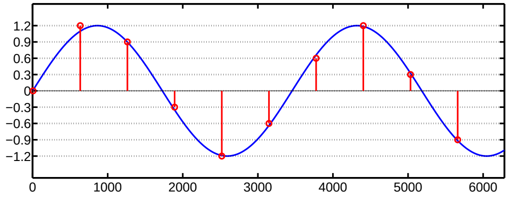
Note that, each number can also be mapped to a new value, e.g. for the uniform quantization shown above, the values might be mapped by taking $\delta \times n$ where n is the number represented by the binary digits. Note the above approach is called "fixed point", which is often used in DSP in preference to "floating point" numbers often used more generally. Arithmetic for fixed point is easier, and there are some other good arguments for using it when you have a limited number of bits.

There is a fair bit more of interest here – we will talk more about it close to the end of the lecture.

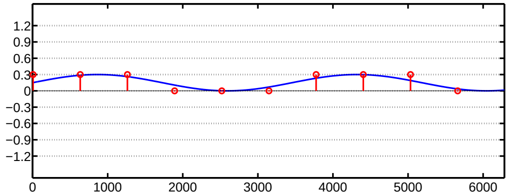
For instance, there is some confusion about the role of the signal representation in the above calculation, for instance CD's use 16 bit fixed point, with one bit used for sign, and 15 bits for value so maybe the dynamic range for CDs should be 90dB, e.g. see `http://www.hydrogenaudio.org/forums/lofiversion/index.php/t45165.html`
The real answer is that the above calculation is a clumsy approximation, but it is often used as a rule of thumb to get a ball-park figure. The figures quoted for CD dynamic range vary from 98dB (using a slightly better approximation to the above) to much significantly lower values using more accurate modelling of the possible signals you can obtain with real hardware.

Dynamic range: examples

When the signal just fills the range of possible values, the maximum amplitude of the signal will be $(2^b - 1)\delta$.



The smallest signal (other than zero) that we can represent has maximum amplitude δ.



Photographers use terms other than dynamic range (e.g. exposure range, luminosity range, f-stops, etc.).
`http://www.cambridgeincolour.com/tutorials/dynamic-range.htm`
Although the terminology was often developed for analogue photography, its sometimes now used for digital cameras. Most digital cameras use a 10 to 14-bit A/D converter (the CCD), but typical image formats use 8 bits for each color. The total number of bits is therefore 24, but in terms of "intensity" we have about 8 bits available.

TVs and computer monitors often use terminology like contrast ratio. For example

Bits	Dynamic range (approx)	Contrast ratio
8	48 dB	256:1
12	72 dB	4096:1
16	96 dB	65536:1

Dynamic range of the human senses

Human senses aren't really digital, but for purposes of comparison we will consider them here. They are pretty amazing.

- We have already seen that the human ear has about 130 dB dynamic range.
- The human eye has about 100 dB dynamic range.

Although the dynamic range is very large, its important to note that our senses can't achieve this range simultaneously.

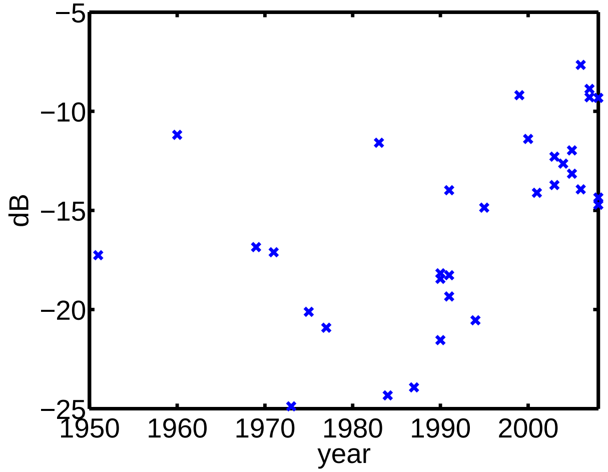
- Loud sounds can mask quieter sounds
- Our eye needs time to adjust to the level of brightness – the range of contrasts is can simultaneously perceive is much smaller.

Ear:
`http://web.mit.edu/2.972/www/reports/ear/ear.html`
`http://www.silcom.com/~aludwig/EARS.htm`
`http://hyperphysics.phy-astr.gsu.edu/Hbase/sound/earsens.html`
`http://en.wikipedia.org/wiki/Ear`

Eye:
`http://en.wikipedia.org/wiki/Eye`
Digital Image Processing, Gonzalez and Woods, pp. 35–44.

Death of Dynamic Range

In recent year there is a trend in Pop music to aim for "louder" music at the expense of dynamic range.



Introduced noise

The noise introduced by quantization is of the order of δ the smallest value representable value. We want to compute the SNR (Signal to Noise Ratio).

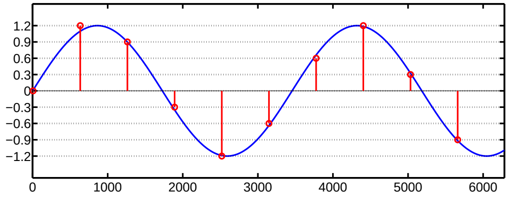
- assume fixed point representation with b bits.
- noise is of the order of δ.
- SNR depends on loading factor.
  - lightly loaded, then δ is relatively large, and so SNR is small.
  - fully loaded, then SNR is similar to dynamic range (6 dB per bit).
  - overloaded, clipping occurs, and SNR drops.

More accurate calculations in "Understanding Digital Signal Processing", Lyons.

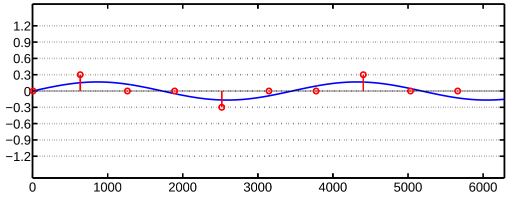
See
http://georgegraham.com/compress.html
http://en.wikipedia.org/wiki/Loudness_war
http://www.cdmasteringservices.com/dynamicrange.htm

Quantization noise notes

When the signal fully loads range of possible values, the maximum amplitude of the signal will be $(2^b - 1)\delta$



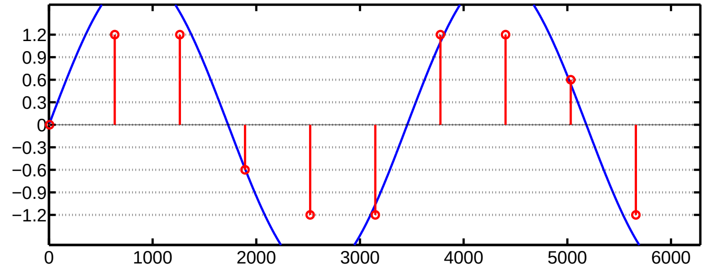
As long as clipping doesn't occur, then the errors will be of order δ, but this is **relatively** larger for small signals



Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.39/80

Clipping

If the signal is too large we get clipping, which results in large amounts of quantization noise, e.g.



Sometimes clipping is used deliberately to alter sounds, for example in a guitar amp, clipping is used to produce distortion (e.g. for heavy-metal music). However, clipping is usually very bad.

BTW, guitar amps are often analogue amplifiers, and so don't "clip" in quite the way described above.

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.40/80

Example

Compact Discs (CDs) are recorded

- using 16 bits
- 44.1 kHz
- so that they record sound frequencies up to 22.05 kHz with a theoretical dynamic range $\simeq$ 96dB.
- Human hearing goes up to about 15 kHz
- LPs have at most 70 dB dynamic range, so CDs should be effectively perfect.
  - audiophiles argue about this
  - some say you lose upper harmonics (not audible but effect tone), or perhaps you loose transient?
  - but I can't tell the difference

- most consumer audio gear will have a noise floor significantly worse than 90dB so arguing about the precise value given 16 bits is not all that useful. So dynamic range/quantization noise doesn't need more than 16 bits.
- 22.05kHz should have all audible frequencies

Example	Sound Pressure Level (dB)	Sound Intensity (watts/m^2)
Snare drums, played hard at 6 inches	150	1000
Fender guitar amplifier, full volume at 10 inches	110	0.1
Typical home stereo listening level	80	0.0001
Conversational speech at 1 foot away	60	10^{-6}
Quiet conversation	40	10^{-8}
Quiet recording studio	10	10^{-11}
Threshold of hearing for healthy youths	0	10^{-12}

For some information on audio equipment, and perception see
`http://www.silcom.com/~aludwig/EARS.htm`
`http://www.cco.caltech.edu/~boyk/spectra/spectra.htm`

Example: extreme audio

Some audio formats propose 96 kHz sampling, at 24 bits. Ignoring audiophile fantasies, why would I want better digital recordings?

- Even if you can't hear it, what about in the studio. In mixing, noise from multiple inputs could add to increase noise floor.
- When an audio signal is dithered to remove structure from the quantization noise, this adds a little noise, so its helpful to have a lower noise floor to start with when recording audio.
- Stereo imaging: requires very finely adjusted time-of-arrival of wavefronts which might be distorted by sampling???

Sampling at 44.1kHz, the sample interval is 1/44100 = 0.000022676 seconds.

At ground level and at 0^o C the speed of sound is approximately 331.5 meters per second. So in one sample, a sounds wave will have moved 0.007517007 meters, or about 7.5 mm.

The wavelength of the note we call A=440Hz. proves to be about 753 mm. So the distortion in one sample at A is about 1% of the wavelength. For a very low note, e.g. A=55Hz, it would be more like 8%.

Is this enough to impact stereo imaging – I don't know?

Discrete Fourier Transform

Mathematics compares the most diverse phenomena and discovers the secret analogies that unite them.

Jean Baptiste Joseph Fourier

Discrete transformation

Discrete-time transformation

- Discrete Fourier transform
- Discrete Cosine (and sin) transforms
- Discrete Wavelet transform
- Z-transform

Discrete-value transformation

- Probability generating function

Discrete Fourier Transformation

Continuous Fourier transform $F(s) = \int_{-\infty}^{\infty} f(t) e^{-i2\pi st}\, dt$

But note that for a finite length, discrete-time signal, it can be written as

$$x(t) = \sum_{n=0}^{N-1} f(nt_s)\delta(t - nt_s)$$

The Fourier transform can then be written

$$X(s) = \sum_{n=0}^{N-1} f(nt_s) e^{-i2\pi s n t_s}$$

The result is simpler to compute, but its still redundant.

Discrete Fourier Transformation

If we have N data points, we would like a (frequency domain) representation that only needs N data points as well. Hence no redundancy.

Use $s = \frac{k}{Nt_s}$ for $k = 0, 1, \ldots, N-1$ and we get

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-i2\pi kn/N},$$

where $x(n)$ are the N discrete samples from the continuous time process.

This is the Discrete Fourier Transform **(DFT)**

Inverse DFT

DFT

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-i2\pi kn/N},$$

Inverse DFT (IDFT)

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{i2\pi kn/N},$$

Examples (i)

Take $x(n) = (1,0,0,0)$

$$\begin{aligned} X(k) &= \textstyle\sum_{n=0}^{N-1} x(n) e^{-i2\pi kn/N} \\ \\ X(0) &= e^{-i2\pi 0/4} && = 1 \\ X(1) &= e^{-i2\pi 0/4} && = 1 \\ X(2) &= e^{-i2\pi 0/4} && = 1 \\ X(3) &= e^{-i2\pi 0/4} && = 1 \end{aligned}$$

So $X(k) = (1,1,1,1)$

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.47/80

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.48/80

Examples (i) IDFT

Take $X(k) = (1,1,1,1)$

$$
\begin{aligned}
x(n) &= \tfrac{1}{N}\textstyle\sum_{n=0}^{N-1} X(k) e^{i2\pi kn/N} \\
x(0) &= \tfrac{1}{4}\left(e^{-i2\pi 0/4} + e^{-i2\pi 0/4} + e^{-i2\pi 0/4} + e^{-i2\pi 0/4}\right) \\
&= \tfrac{1}{4}(1+1+1+1) &= 1 \\
x(1) &= \tfrac{1}{4}\left(e^{-i2\pi 0/4} + e^{-i2\pi 1/4} + e^{-i2\pi 2/4} + e^{-i2\pi 3/4}\right) \\
&= \tfrac{1}{4}(1+i-1-i) &= 0 \\
x(2) &= \tfrac{1}{4}\left(e^{-i2\pi 0/4} + e^{-i2\pi 2/4} + e^{-i2\pi 4/4} + e^{-i2\pi 6/4}\right) \\
&= \tfrac{1}{4}(1-1+1-1) &= 0 \\
x(3) &= \tfrac{1}{4}\left(e^{-i2\pi 0/4} + e^{-i2\pi 3/4} + e^{-i2\pi 6/4} + e^{-i2\pi 9/4}\right) \\
&= \tfrac{1}{4}(1-i-1+i) &= 0
\end{aligned}
$$

So $x(n) = (1,0,0,0)$

Examples (ii)

Take $x(n) = (0,1,0,0)$

$$
X(k) = \textstyle\sum_{n=0}^{N-1} x(n) e^{-i2\pi kn/N}
$$

$$
\begin{aligned}
X(0) &= e^{-i2\pi 0/4} &&= 1 \\
X(1) &= e^{-i2\pi 1/4} &&= e^{-i\pi/2} &&= -i \\
X(2) &= e^{-i2\pi 2/4} &&= e^{-i\pi} &&= -1 \\
X(3) &= e^{-i2\pi 3/4} &&= e^{-i\pi 3/2} &&= i
\end{aligned}
$$

So $X(k) = (1,-i,-1,i)$

Examples (iii)

Take $x(n) = (1, 1, 0, 0)$

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-i2\pi kn/N}$$

$$\begin{aligned}
X(0) &= e^{-i2\pi 0/4} + e^{-i2\pi 0/4} &= 1+1 &= 2 \\
X(1) &= e^{-i2\pi 0/4} + e^{-i2\pi 1/4} &= e^0 + e^{-i\pi/2} &= 1-i \\
X(2) &= e^{-i2\pi 0/4} + e^{-i2\pi 2/4} &= e^0 + e^{-i\pi} &= 0 \\
X(3) &= e^{-i2\pi 0/4} + e^{-i2\pi 3/4} &= e^0 + e^{-i\pi 3/2} &= 1+i
\end{aligned}$$

So $X(k) = (2, 1-i, 0, 1+i)$

DFT basis

Once again we are simply changing basis, when we perform the transform (or its inverse).

The basis functions are a discrete set of sin and cosine functions.

Note, now we are operating in a finite dimensional space $\mathbb{R}^N$, so we can write the transform as

$$X = Ax \qquad \text{analysis}$$

The inverse transform is just

$$x = A^{-1}X \qquad \text{synthesis}$$

Where both x and X are just vectors in $\mathbb{R}^N$.

DFT transform matrix

$$X = Ax$$

$$A = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-i2\pi 1/N} & e^{-i2\pi 2/N} & \cdots & e^{-i2\pi(N-1)/N} \\ 1 & e^{-i2\pi 2/N} & e^{-i2\pi 4/N} & \cdots & e^{-i2\pi 2(N-1)/N} \\ 1 & e^{-i2\pi 3/N} & e^{-i2\pi 6/N} & \cdots & e^{-i2\pi 3(N-1)/N} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & e^{-i2\pi(N-1)/N} & e^{-i2\pi 2(N-1)/N} & \cdots & e^{-i2\pi(N-1)(N-1)/N} \end{pmatrix}$$

Examples (i)

Take $x(n) = (1, 0, 0, 0)$

$$X = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-i2\pi 1/4} & e^{-i2\pi 2/4} & e^{-i2\pi 3/4} \\ 1 & e^{-i2\pi 2/4} & e^{-i2\pi 4/4} & e^{-i2\pi 6/4} \\ 1 & e^{-i2\pi 3/4} & e^{-i2\pi 6/4} & e^{-i2\pi 9/4} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.53/80

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.54/80

Frequency resolution

Frequencies of basis functions are $k = 0, 1, 2, \ldots, (N-1)$ cycles over the data set. If the data set has N samples at sampling frequency f_s, then its duration is $T = N/f_s$. To convert from data units to absolute units, we take $k/T = \frac{kf_s}{N}$

Frequency resolution is $\frac{f_s}{N}$

- higher sampling frequencies reduce frequency resolution
- longer data, improves frequency resolution

Getting units right

Note that absolute frequency depends on sample frequency f_s, so we need to convert.
The component $X(m)$ will correspond to frequency

$$X(m) \equiv F\left(\frac{mf_s}{N}\right)$$

Output magnitude of DFT will be amplitude of sin wave signal A times $N/2$. Alternative definitions of DFT exist

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-i2\pi kn/N}, \qquad x(n) = \sum_{n=0}^{N-1} X(k)e^{i2\pi kn/N}$$

$$X(k) = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} x(n)e^{-i2\pi kn/N}, \qquad x(n) = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} X(k)e^{i2\pi kn/N}$$

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.55/80

e.g. Given we sample at 20 kHz (i.e. $t_s = 0.05$ ms) for 5 seconds (i.e. $N = 100{,}000$), and we measure frequency context at $X(20)$, i.e. 20 cycles/measurement period, then the frequency of the original signal will be

$$\frac{20 \times 20{,}000}{100{,}000} = 4Hz$$

Note that we care more about relative magnitudes, not absolute values, so the different scalings in the DFT don't really matter. Except on class exercise solutions :-)

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.56/80

Matlab

Note, indexes in Matlab run from 1 to N (not 0 to $N-1$).

$$\text{fft}(x(n)) = X(k) = \sum_{n=1}^{N} x(n) e^{-i2\pi(k-1)(n-1)/N}, \quad k = 1,\ldots,N.$$

$$\text{ifft}(X(k)) = x(n) = \frac{1}{N} \sum_{k=1}^{N} X(k) e^{i2\pi(k-1)(n-1)/N}, \quad n = 1,\ldots,N.$$

$X(1)$ is the DC term, $X(n)$ is the f_s term. To plot symmetric power spectrum use, e.g.

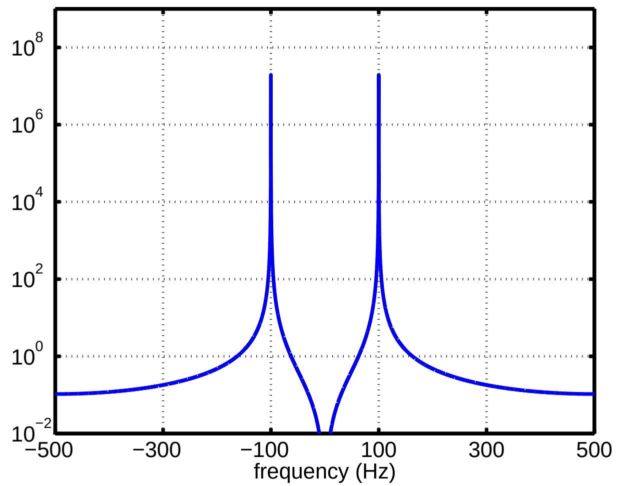
```
f_s = 1000;
f_0 = 100;
x = 1:1/f_s:10;
y = sin(2*pi*f_0*x);
semilogy(-f_s/2+f_s/N:f_s/N:f_s/2, abs(fftshift(fft(y))).^2);
set(gca, 'ylim', 10.^[-2 9]);
xlabel('frequency (Hz)');
```


Note that the actual implementation of the DFT is not performed as its described above. In actuality we use an algorithm called the Fast Fourier Transform (FFT), which we will discuss in lecture 7. Hence the function names in matlab, e.g., `fft` and `ifft` (for inverse FFT).

Note the use of `fftshift` in the above code. This is used in matlab to display the DFT symmetrically around the DC term. See what happens without it in the following slide.

Matlab example

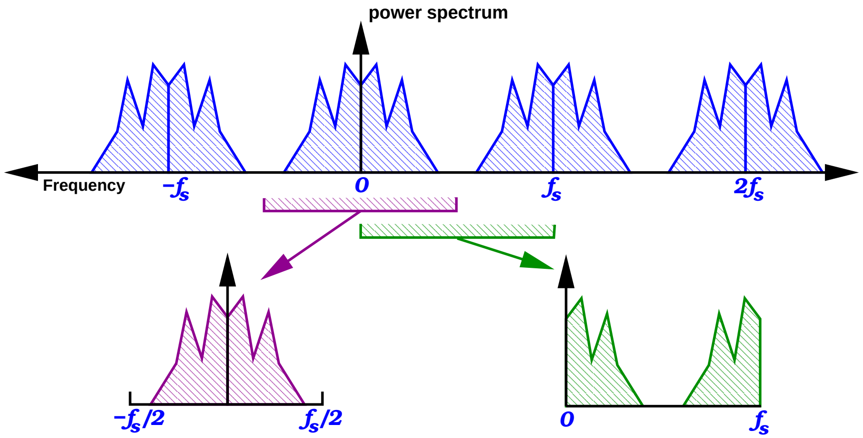
matlab_ex_1.m




```
% MATLAB_EX_1 shows a simple example of fft in practice
%
% file:       matlab_ex_1.m, (c) Matthew Roughan, Sat Aug  7 2004
% directory:   /home/mroughan/Classes/Transformations/2004/Matlab/
%
%
f_s = 1000;             % sampling frequency
f_0 = 100;              % frequency of the signal
x = 1:1/f_s:10;         % sample points
N = length(x);
y = sin(2*pi*f_0*x);   % sampled signal
semilogy(-f_s/2+f_s/N:f_s/N:f_s/2, abs(fftshift(fft(y))).^2, 'linewidth', 3);

%%%% make the axes pretty and add labels
grid on
set(gca, 'ylim', 10.^[-2 9], 'ytick',  10.^[-2:2:9], 'xtick', [-500:200:500]);
set(gca, 'linewidth', 3, 'fontsize', 18);
xlabel('frequency (Hz)');

%%%% print out a copy
print('-depsc', 'Plots/matlab_ex_1.eps');
```

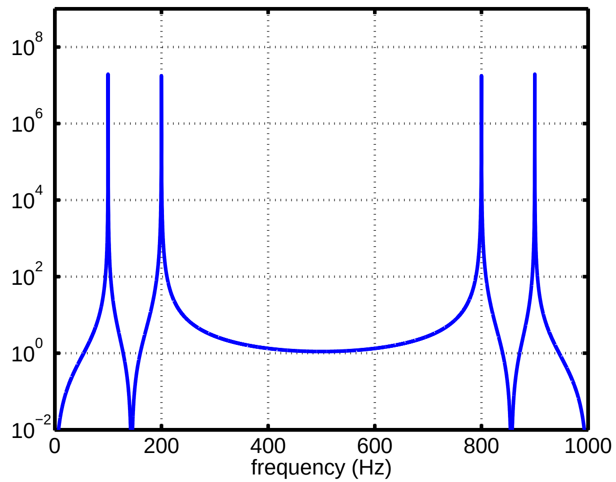
Symmetry

Discrete power spectrum is **even** and **periodic** so we can display in a number of ways.



Matlab example 2

matlab_ex_2.m




```
% MATLAB_EX_1 shows a simple example of fft in practice
%
% file:       matlab_ex_1.m, (c) Matthew Roughan, Sat Aug  7 2004
% directory:   /home/mroughan/Classes/Transformations/2004/Matlab/
%
%
f_s = 1000;              % sampling frequency
x = 1:1/f_s:10;          % sample points
N = length(x);

%%%%  sampled signal
f_0 = 100;               % frequency 0 in the signal
f_1 = 200;               % frequency 1 in the signal
y = sin(2*pi*f_0*x) + sin(2*pi*f_1*x);

%%%%  FFT of data
z = fft(y);
freq = (0:N-1) * f_s/N;

%%%%  plot the data
semilogy(freq, abs(z).^2, 'linewidth', 3);

%%%% make the axes pretty and add labels
grid on
set(gca, 'ylim', 10.^[-2 9], 'ytick',  10.^[-2:2:9], 'xtick', [0:200:1000]);
set(gca, 'linewidth', 3, 'fontsize', 18);
xlabel('frequency (Hz)');

%%%% print out a copy
print('-depsc', 'Plots/matlab_ex_2.eps');
```

Properties of the DFT

Mostly the same as Continuous FT

- invertible
- no redundancy so it is efficient
- Linearity: $ax_1(n) + bx_2(n) \rightarrow aX_1(k) + bX_2(k)$
- Time shift: $x(n - n_0) \rightarrow X(k)e^{-i2\pi k n_0}$
- Time scaling: a bit more complicated!
- Duality: a bit more complicated!
- Frequency shift: $x(n)e^{-i2\pi k_0 n} \rightarrow X(k - k_0)$
- Convolution: $x_1(n) * x_2(n) \rightarrow X_1(k)X_2(k)$

Now n and k are integers, with the result that we are missing properties related to derivatives.

Convolution is still really important. There is a java applet to play with at
`http://www.jhu.edu/~signals/discreteconv2/index.html`

Duality and the DFT

The duality property is a little changed from before: given a signal $x(n)$ for $n = 0, \ldots, N-1$, with DFT $X(k)$ for $k = 0, \ldots, N-1$, then the DFT of $X(n)$ is

$$\begin{aligned} DFT(X;k) &= \begin{cases} Nx(0), & \text{for } k = 0 \\ Nx(N-k), & \text{for } k \neq 0 \end{cases} \\ &= Nx(N - k \bmod N) \end{aligned}$$

The result is similar to previous duality results if we think of the points cyclically, i.e.

$$x(-k \bmod N) = x(N - k \bmod N)$$

That works well with the periodic representation of frequency spectrum that we get for a sampled signal.

The duality property is different from the continuous case for a couple of reasons:

- our definition of the DFT is not symmetric — the inverse transform has a factor of $1/N$ that doesn't appear in the forward transform. Remember that we are using a symmetric definition for the continuous Fourier transform. We use the asymmetric definition here for ease, because it is consistent with Matlab.
- the signal itself is no longer symmetric — we assume it has N points $x(n)$ for $n = 0, \ldots, N-1$, so it only makes sense to discuss $x(-n)$ in the cyclical sense above.

Proof: take the DFT of $X(n)$ (and noting that $e^{i2\pi Nn/N} = 1$ for $n \in \mathbb{N}$), for $k \neq 0$

$$\begin{aligned} \sum_{n=0}^{N-1} X(n)e^{-i2\pi kn/N} &= \sum_{n=0}^{N-1} X(n)e^{i2\pi Nn/N}e^{-i2\pi kn/N} \\ &= \sum_{n=0}^{N-1} X(n)e^{i2\pi(N-k)n/N} \\ &= Nx(N-k) \end{aligned}$$

as it has become N times an IDFT. So the DFT of $X(n)$ is $Nx(N-k)$.

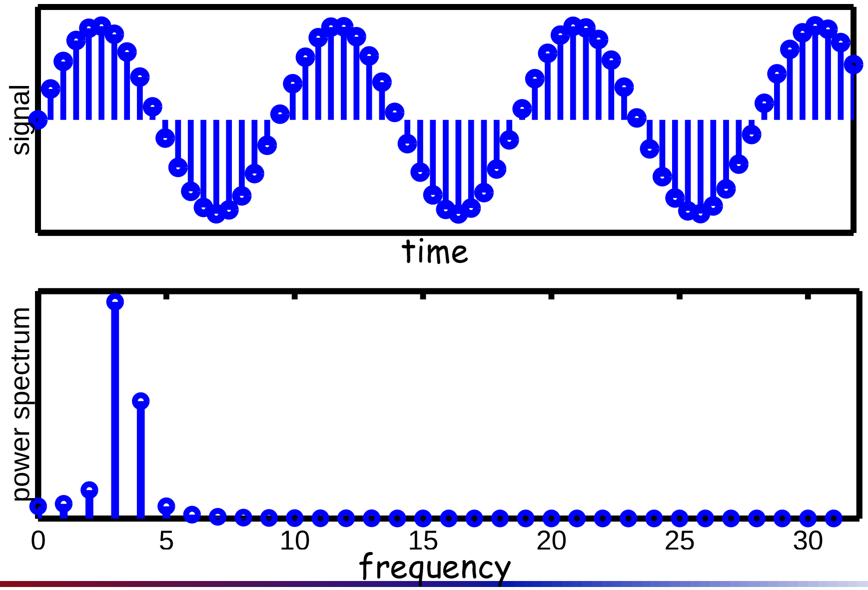
Properties of the DFT

There are some new properties unique to DFTs

- Leakage that fits exactly our discrete frequencies
- Padding (packing)
- Similarity (discrete version of time scaling)

See below for details.

Leakage example



- Leakage – what happens when the signal doesn't have a period that fits exactly our discrete frequencies
- Padding (packing) – what happens when we put zeros at the end of a set of data
- Similarity – what happens when we interleave zeros in a signal

Properties of the DFT: Leakage

DFT is different from the continuous time FT is that the DFT suffers from **Leakage**.

- Unlike Continuous transform, DFT uses a finite number of frequencies.
- Not all signals fit this mold exactly: what happens to sinusoids with non-integral frequencies?
- Their power is spread over a few frequencies.
- Note we are representing the signal by a series of numbers $X(k)$ which represent the correlation of the signal to a particular sinusoid with freq. k/N,
- Note that, as the data gets longer, the frequency resolution improves

Leakage has also been called "window splatter" for reasons that will become clear around lecture 8 when we considering windowing.

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.65/80

DFT properties: padding

We can pad (or pack) a sequence with zeros to extend its length

$$y(n) = \begin{cases} x(n), & \text{if } 0 \le n \le N-1 \\ 0, & \text{if } N \le n < KN \end{cases}$$

The resulting DFT is

$$\mathcal{F}\{y\} = Y(k) = X\left(\frac{k}{K}\right)$$

Transform Methods & Signal Processing (APP MTH 4043): lecture 03 – p.66/80

Padding (packing) example (ii)

Data $x(n) = (0,1,0,0)$ with transform $X(k) = (1,-i,-1,i)$
Pad to get $y(n) = (0,1,0,0,0,0,0,0)$ then the DFT

$$Y(k) = \sum_{n=0}^{N-1} y(n) e^{-i2\pi kn/N}$$

$$\begin{aligned}
Y(0) &= e^{-i2\pi 0/8} && = 1 \\
Y(1) &= e^{-i2\pi 1/8} && = e^{-i\pi/4} = (1-i)/\sqrt{2} \\
Y(2) &= e^{-i2\pi 2/8} && = e^{-i\pi/2} = -i \\
Y(3) &= e^{-i2\pi 3/8} && = e^{-i\pi 3/4} = (-1-i)/\sqrt{2} \\
Y(4) &= e^{-i2\pi 4/8} && = e^{-i\pi} = -1 \\
Y(5) &= e^{-i2\pi 5/8} && = e^{-i\pi 5/4} = (-1+i)/\sqrt{2} \\
Y(6) &= e^{-i2\pi 6/8} && = e^{-i\pi 3/2} = i \\
Y(7) &= e^{-i2\pi 7/8} && = e^{-i\pi 7/4} = (1+i)/\sqrt{2}
\end{aligned}$$

Padding (packing) example (ii)

Data $x(n) = (0,1,0,0)$ with transform $X(k) = (1,-i,-1,i)$
Pad to get $y(n) = (0,1,0,0,0,0,0,0)$ then the DFT

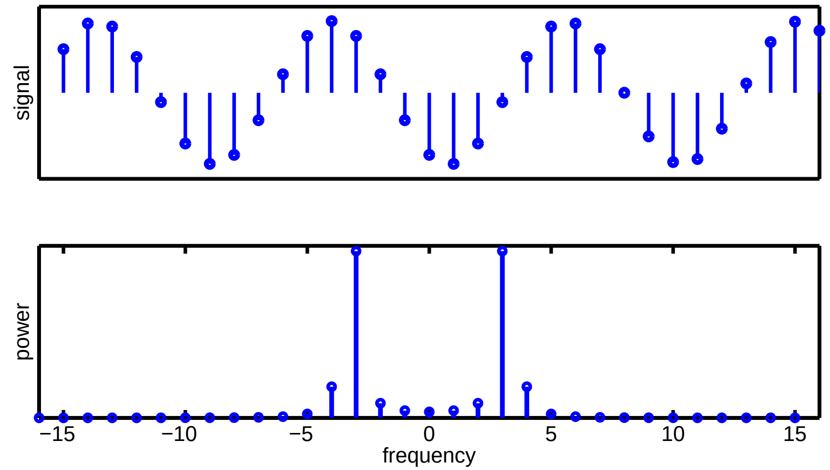
$$\begin{aligned}
Y(0) &= X(0) \\
Y(2) &= X(1) \\
Y(4) &= X(2) \\
Y(6) &= X(3)
\end{aligned}$$

So the relationship $Y(k) = X(k/2)$ holds, with $K = 2$, for even values of k.

Note we cannot derive $Y(k)$ for odd values of k, or if K is not an integer, but the relationship still tells us how to scale the frequency units, when we pad.

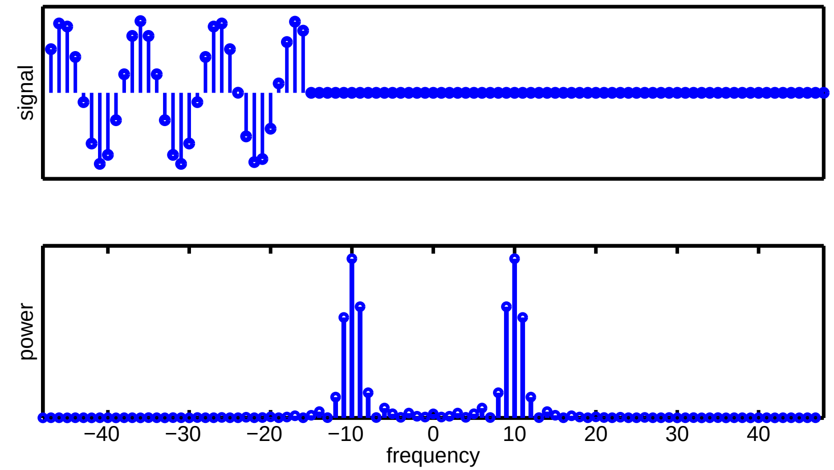
Padding (packing) example

Original data length $N = 32$ (frequency = 3.333)



Padding (packing) example

$K = 3$, new sequence length $KN = 96$. (frequency = 10/K)




```
% file:       padding.m, (c) Matthew Roughan, Thu Aug  5 2004
%
N = 32;
x = (1:N)/N;
f1 = 3.3333;
y1 = sin(2*pi*f1*x);

figure(1)
subplot(2,1,1);
hold off
plot([x; x], [zeros(size(y1)); y1],'b', 'linewidth', 3);
hold on
plot(x, y1, 'bo', 'linewidth', 4);
set(gca, 'linewidth', 3, 'fontsize', 18, 'ylim', [-1.2 1.2], 'xlim', [0 max(x)], 'xtick', [], 'ytick', []);
ylabel('signal');
z1 = fftshift(abs(fft(y1)).^2);
subplot(2,1,2);
hold off
plot([-N/2:N/2-1; -N/2:N/2-1], [zeros(1, N); z1], 'b', 'linewidth', 4);
hold on
plot(-N/2:N/2-1, z1, 'bo', 'linewidth', 3);
set(gca, 'linewidth', 3, 'fontsize', 18, 'xlim', [-N/2 N/2], 'ytick', []);
ylabel('power');
xlabel('frequency');
set(gcf, 'PaperUnits', 'centimeters', 'PaperPosition', [0 0 30 15])
print('-depsc', 'Plots/padding.eps');

%%%%%%% pad the data
K = 3;
pad = zeros(1,length(y1)*K);
pad((1:length(y1))) = y1;
x_u = (1:K*N)/(K*N);
figure(2)
subplot(2,1,1);
hold off
plot([x_u; x_u], [zeros(size(pad)); pad],'b', 'linewidth', 3);
hold on
plot(x_u, pad, 'bo', 'linewidth', 4);
```

DFT properties: similarity

We can interleave a sequence with zeros, e.g.

$$y(n) = \begin{cases} x(n/K), & \text{if } n = 0, K, 2K, \ldots, (N-1)K \\ 0, & \text{otherwise} \end{cases}$$

The resulting DFT is

$$\mathcal{F}\{y\} = Y(k) = \begin{cases} X(k) & k = 0, \ldots, N-1 \\ X(k-N) & k = N, \ldots, 2N-1 \\ \vdots & \\ X(k-(K-1)N) & k = (K-1)N, \ldots, KN-1 \end{cases}$$

Similarity example (ii)

Data $x(n) = (0,1,0,0)$ with transform $X(k) = (1,-i,-1,i)$
Interleave zeros to get $y(n) = (0,0,1,0,0,0,0,0)$ then

$$\begin{aligned}
Y(k) &= \textstyle\sum_{n=0}^{N-1} y(n) e^{-i2\pi kn/N} \\
Y(0) &= e^{-i2\pi 0/8} &&= 1 \\
Y(1) &= e^{-i2\pi 2/8} &&= e^{-i\pi/2} &&= -i \\
Y(2) &= e^{-i2\pi 4/8} &&= e^{-i\pi} &&= -1 \\
Y(3) &= e^{-i2\pi 6/8} &&= e^{-i\pi 3/2} &&= i \\
Y(4) &= e^{-i2\pi 8/8} &&= e^{-i2\pi} &&= 1 \\
Y(5) &= e^{-i2\pi 10/8} &&= e^{-i\pi 5/2} &&= -i \\
Y(6) &= e^{-i2\pi 12/8} &&= e^{-i\pi 3} &&= -1 \\
Y(7) &= e^{-i2\pi 14/8} &&= e^{-i\pi 7/2} &&= i
\end{aligned}$$

So $Y(k) = (1,-i,-1,i,1,-i,-1,i)$ (or $X(k)$ repeated twice)

Duality applies with similarity, i.e.,

$$DFT(X;k) = \begin{cases} x(0), & \text{for } k = 0 \\ Nx(N-k), & \text{for } k \neq 0 \end{cases}$$

so if we repeat a signal in the time domain, we can compute its Fourier transform by interleaving zeros in the Fourier domain.

Sampling, Quantization, Dithering and Half-toning

The properties we have just seen leed to some direct applications. In particular, we don't always get a signal in the form we want it, so we may have to change its sampling rate, or quantization, and we can exploit our new mathematically derived intuition to start work out how to do this (we'll see more later).

Similarity application

Practical use: upsampling (interpolation)

> We have a sequence sampled every t_s seconds, e.g. at a rate $f_s = 1/t_s$, but we need a sequence sampled at rate $K f_s$.

Approach: produce a new sequence with $K-1$ zeros interleaved between each original data point.

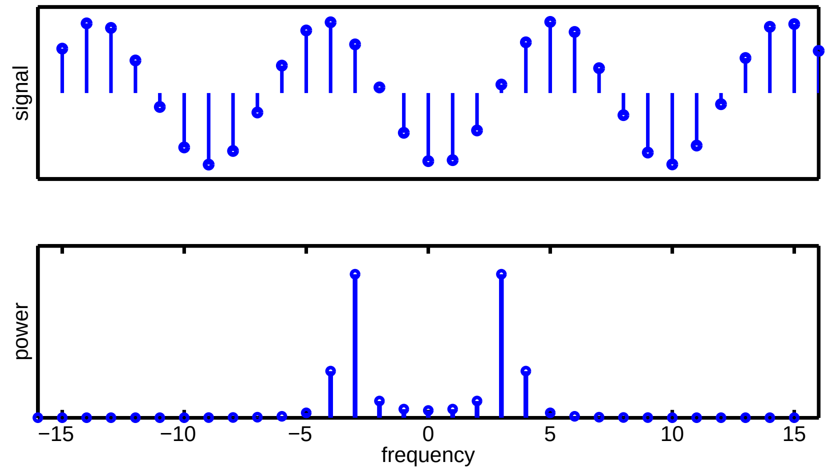
Similarity application: upsampling

Given $K-1$ zeros interleaved between each original sample.

- max frequency in original data is $f_s/2$, with frequency resolution f_s/N, and $N/2$ points in frequency domain.
- upsampled data has max frequency $Kf_s/2$, with frequency resolution f_s/N, and $KN/2$ points in frequency domain.
- the frequency resolution doesn't change, but now we have K repeats of the original spectrum at intervals f_s/N.
- to get a signal with the same original band-limited power-spectrum, we apply a low-pass filter, smoothing the data.

Upsampling example

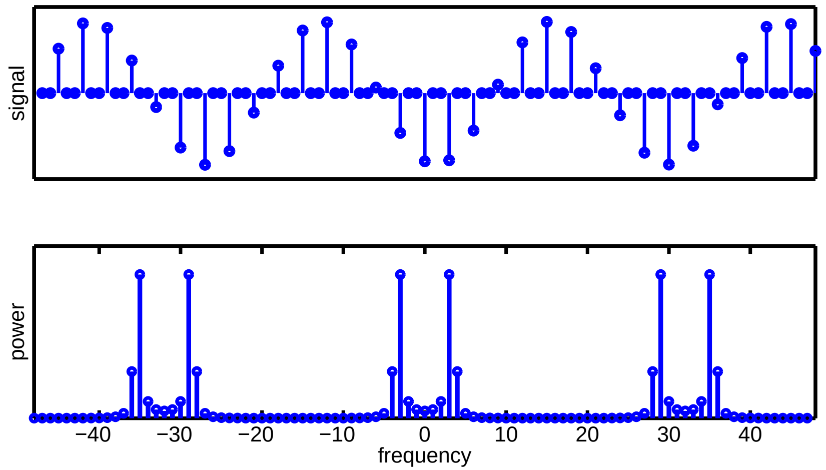
32 samples (frequency 3.4 cycles)




```
% file:       upsampling_1.m, (c) Matthew Roughan, Thu Aug  5 2004
%
N = 32;
x = (1:N)/N;
f1 = 3.4;
y1 = sin(2*pi*f1*x);

figure(1)
subplot(2,1,1);
hold off
plot([x; x], [zeros(size(y1)); y1],'b', 'linewidth', 3);
hold on
plot(x, y1, 'bo', 'linewidth', 4);
set(gca, 'linewidth', 3, 'fontsize', 18);
ylabel('signal');
set(gca, 'ylim', [-1.2 1.2], 'xlim', [0 max(x)], 'xtick', [], 'ytick', []);

z1 = fftshift(abs(fft(y1)).^2);
subplot(2,1,2);
hold off
plot([-N/2:N/2-1; -N/2:N/2-1], [zeros(1, N); z1], 'b', 'linewidth', 4);
hold on
plot(-N/2:N/2-1, z1, 'bo', 'linewidth', 3);
set(gca, 'linewidth', 3, 'fontsize', 18);
set(gca, 'xlim', [-N/2 N/2], 'ytick', []);
ylabel('power');
xlabel('frequency');

set(gcf, 'PaperUnits', 'centimeters', 'PaperPosition', [0 0 30 15])
print('-depsc', 'Plots/upsampling.eps');
```

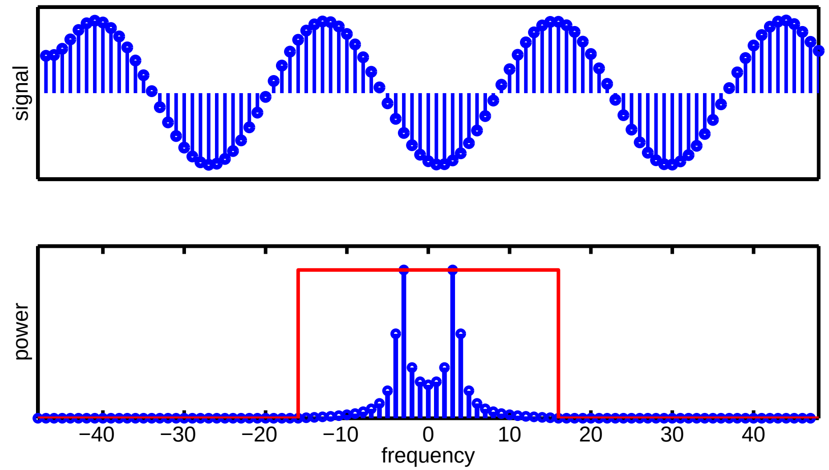
Upsampling example

3 ×'s upsampled (96 samples)



Upsampling example

low pass filter, then IDFT




```
% file:       upsampling_2.m, (c) Matthew Roughan, Thu Aug  5 2004
%
N = 32;
x = (1:N)/N;
f1 = 3.4;
y1 = sin(2*pi*f1*x);
K = 3; % upsample by 3
upsample = zeros(1,length(y1)*K);
upsample(K*(1:length(y1))) = y1;
x_u = (1:K*N)/(K*N);

figure(2)
subplot(2,1,1);
hold off
plot([x_u; x_u], [zeros(size(upsample)); upsample],'b', 'linewidth', 3);
hold on
plot(x_u, upsample, 'bo', 'linewidth', 4);
set(gca, 'linewidth', 3, 'fontsize', 18);
ylabel('signal');
set(gca, 'ylim', [-1.2 1.2], 'xlim', [0 max(x)], 'xtick', [], 'ytick', []);

z2 = fftshift(abs(fft(upsample)).^2);
subplot(2,1,2);
hold off
plot([-K*N/2:K*N/2-1; -K*N/2:K*N/2-1], [zeros(1, K*N); z2], 'b', 'linewidth', 4);
hold on
plot(-K*N/2:K*N/2-1, z2, 'bo', 'linewidth', 3);
set(gca, 'linewidth', 3, 'fontsize', 18);
set(gca, 'xlim', [-K*N/2 K*N/2], 'ytick', []);
ylabel('power');
xlabel('frequency');

set(gcf, 'PaperUnits', 'centimeters', 'PaperPosition', [0 0 30 15])
```



```
% file:       upsampling_3.m, (c) Matthew Roughan, Thu Aug  5 2004
%
N = 32;
x = (1:N)/N;
f1 = 3.4;
y1 = sin(2*pi*f1*x);
K = 3;
upsample = zeros(1,length(y1)*K);
upsample(K*(1:length(y1))) = y1;
x_u = (1:K*N)/(K*N);
z = fft(upsample);
z(N/2+1:end-N/2+1) = 0; %%% filter the data (using perfect low-pass in freq. domain).
y3 = real(ifft(z));

figure(3)
subplot(2,1,1);
hold off
plot([x_u; x_u], [zeros(size(y3)); y3],'b', 'linewidth', 3);
hold on
plot(x_u, y3, 'bo', 'linewidth', 4);
set(gca, 'linewidth', 3, 'fontsize', 18);
ylabel('signal');
set(gca, 'ylim', [-1.2 1.2]/K, 'xlim', [0 max(x)], 'xtick', [], 'ytick', []);

subplot(2,1,2);
hold off
plot([-K*N/2:K*N/2-1; -K*N/2:K*N/2-1], [zeros(1, K*N); abs(fftshift(z))], 'b', 'linewidth', 4);
hold on
plot(-K*N/2:K*N/2-1, abs(fftshift(z)), 'bo', 'linewidth', 3);
YY = get(gca,'ylim');
plot([-K*N/2 -N/2 -N/2 N/2 N/2 K*N/2], [0 0 max(abs(z)) max(abs(z)) 0 0], 'r', 'linewidth', 3);
set(gca, 'linewidth', 3, 'fontsize', 18);
set(gca, 'xlim', [-K*N/2 K*N/2], 'ytick', []);
```

Upsampling tricks

Trick of the day: low-pass before upsampling.

- notionally, the filtering occurs after upsampling
- If filtering in the time domain however, $K-1/K$ proportion of multiplies in the filter are by zero.
- can ignore these, but this is the same as low-pass before upsampling.

Let's revisit this later (after discussing filtering in more detail).

Downsampling can be accomplished similarly, and combined we can perform resampling.

Upsampling applications: audio

Oversampling CD or DVD players

- digital components are cheap
- analogue components are more expensive
- Digital to Analogue Conversion (DAC) is required in CD player
- want to make this as cheap as possible (for a given quality)

The trick

- upsample in the digital domain (where it is cheap)
- when we convert to analogue, we can use a simpler, cheaper analogue filter, to get the same results

Notes:
`http://stereophile.com/asweseeit/344/`