Maths for AI

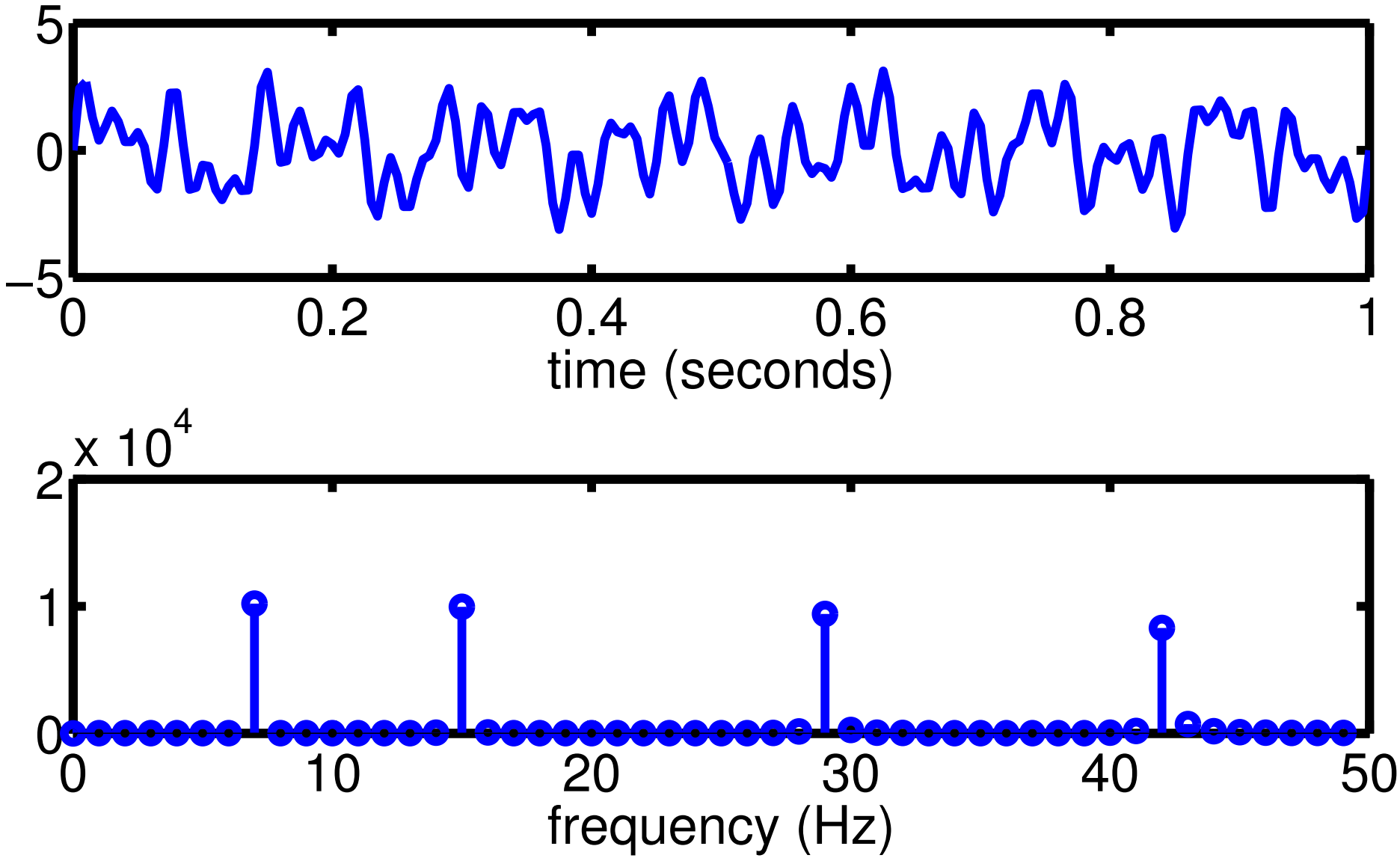
Lecture 1: Sampling

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

School of Mathematical Sciences and AIML,
University of Adelaide

March 15, 2022

An example: the Fourier transform



Discrete signals

Real signals (these days) are discrete (time and value)

- 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).

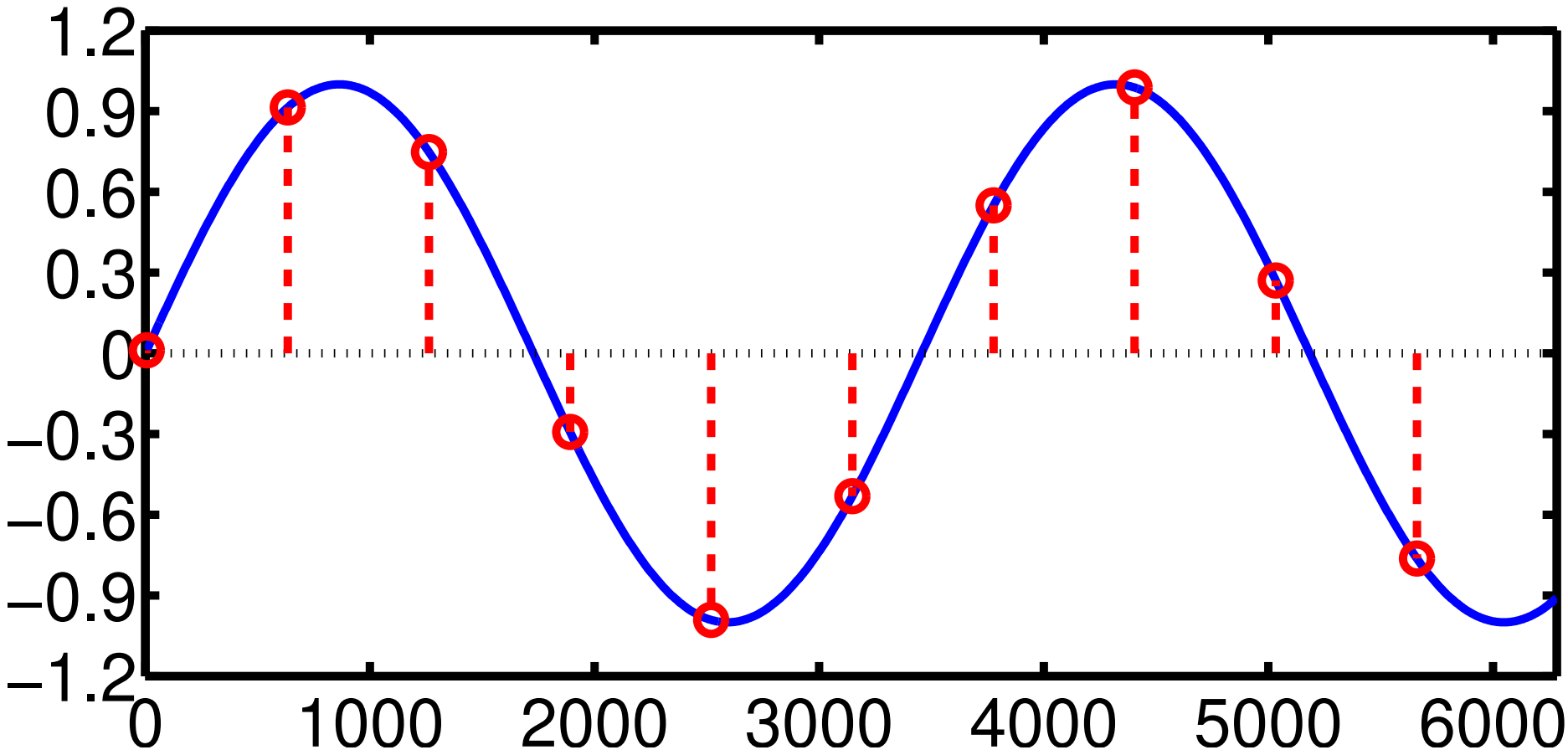
  "Cramming more components into integrated circuits", Gordon E. Moore, Electronics, Vol. 38, No. 8, April, 1965.

- 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.

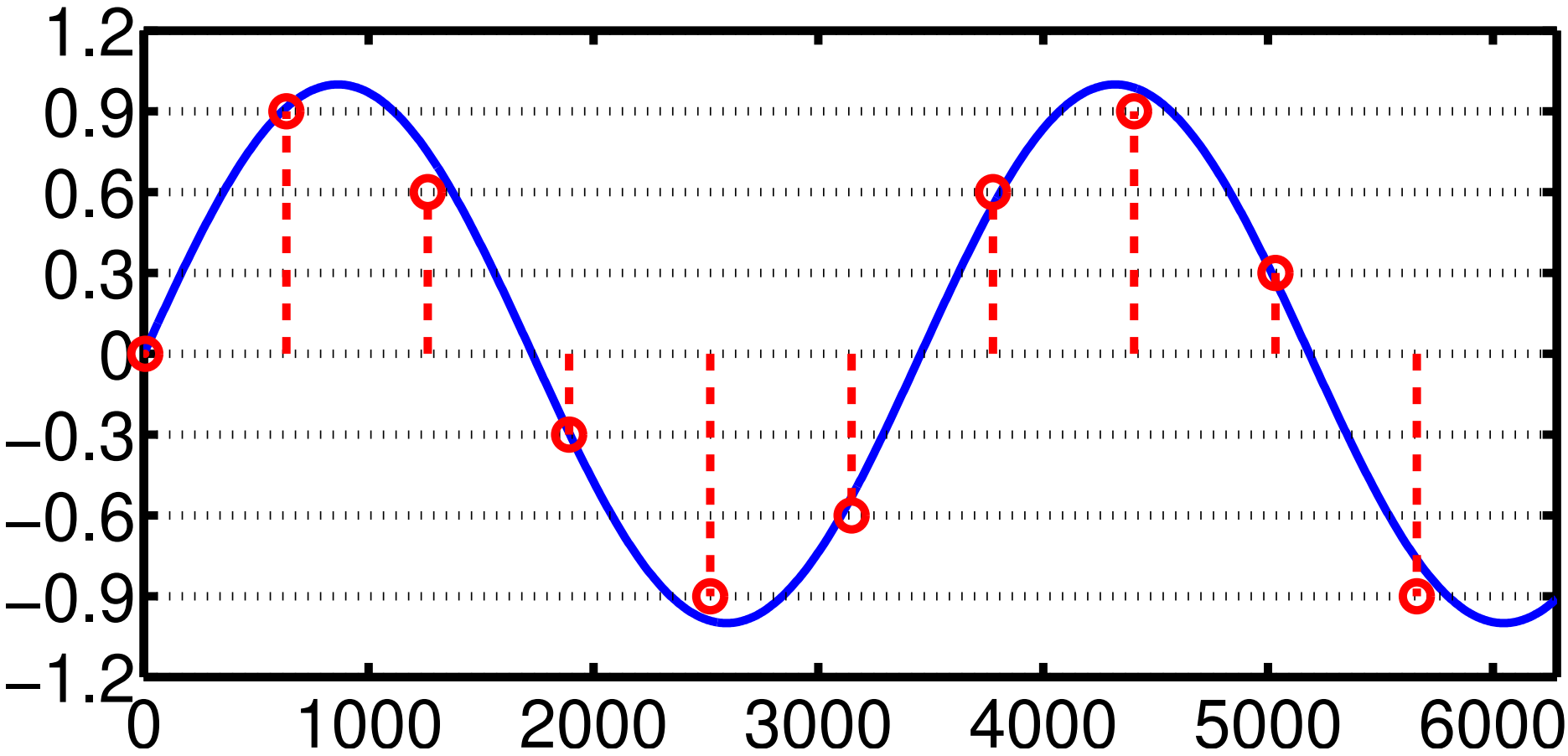
Discrete time

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



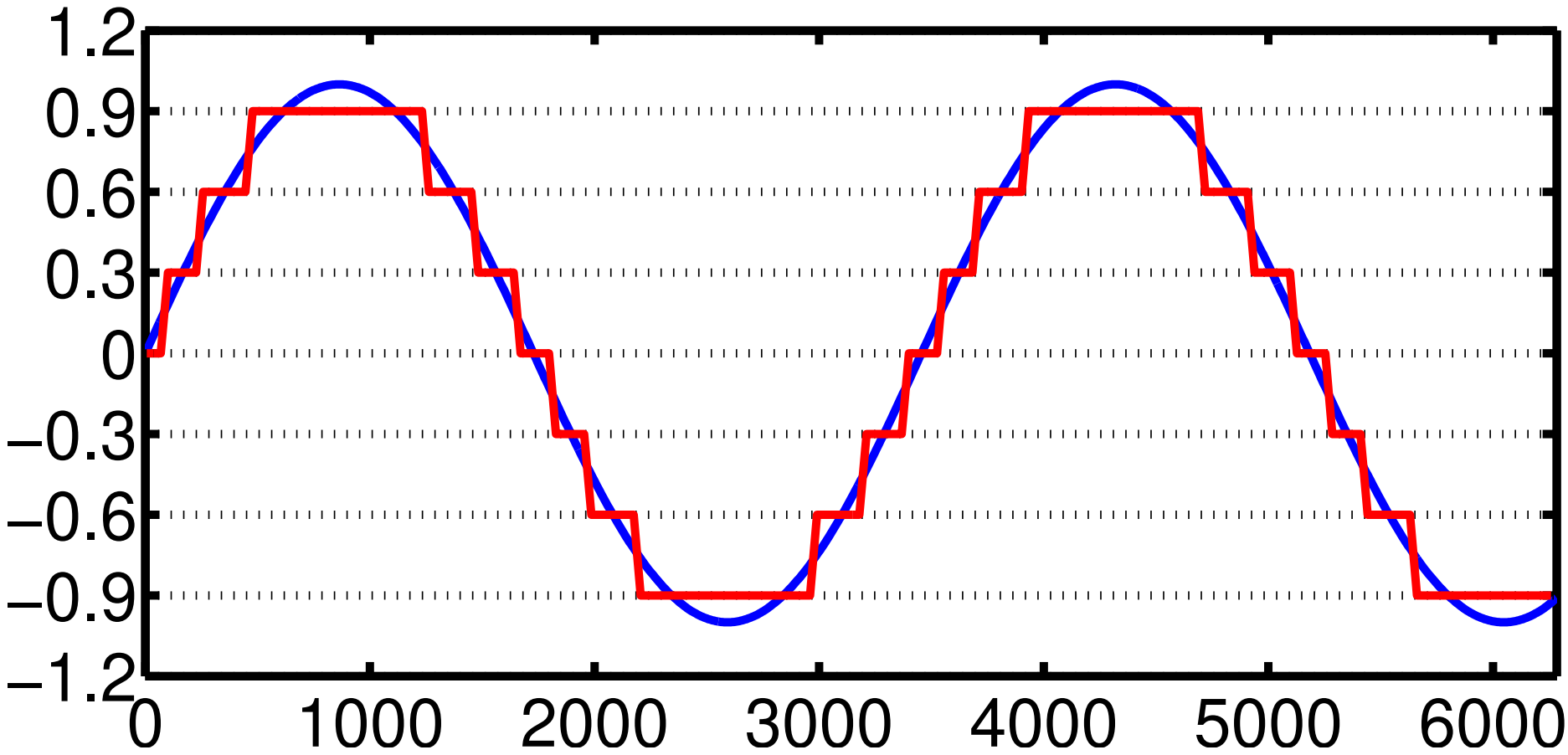
Quantization: discrete-value

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



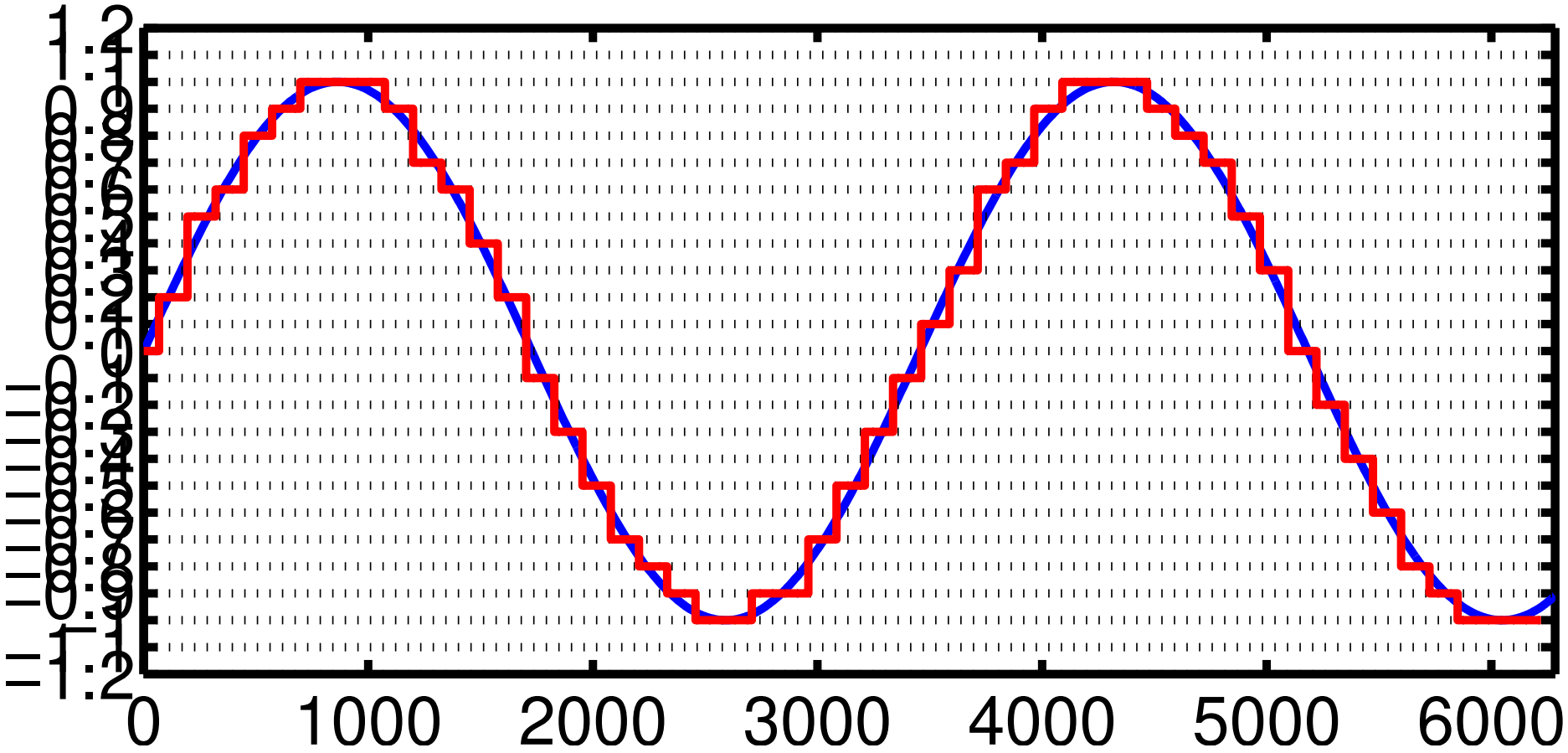
Approximation

- Faster sampling $\Rightarrow$ better approximation
- More details later



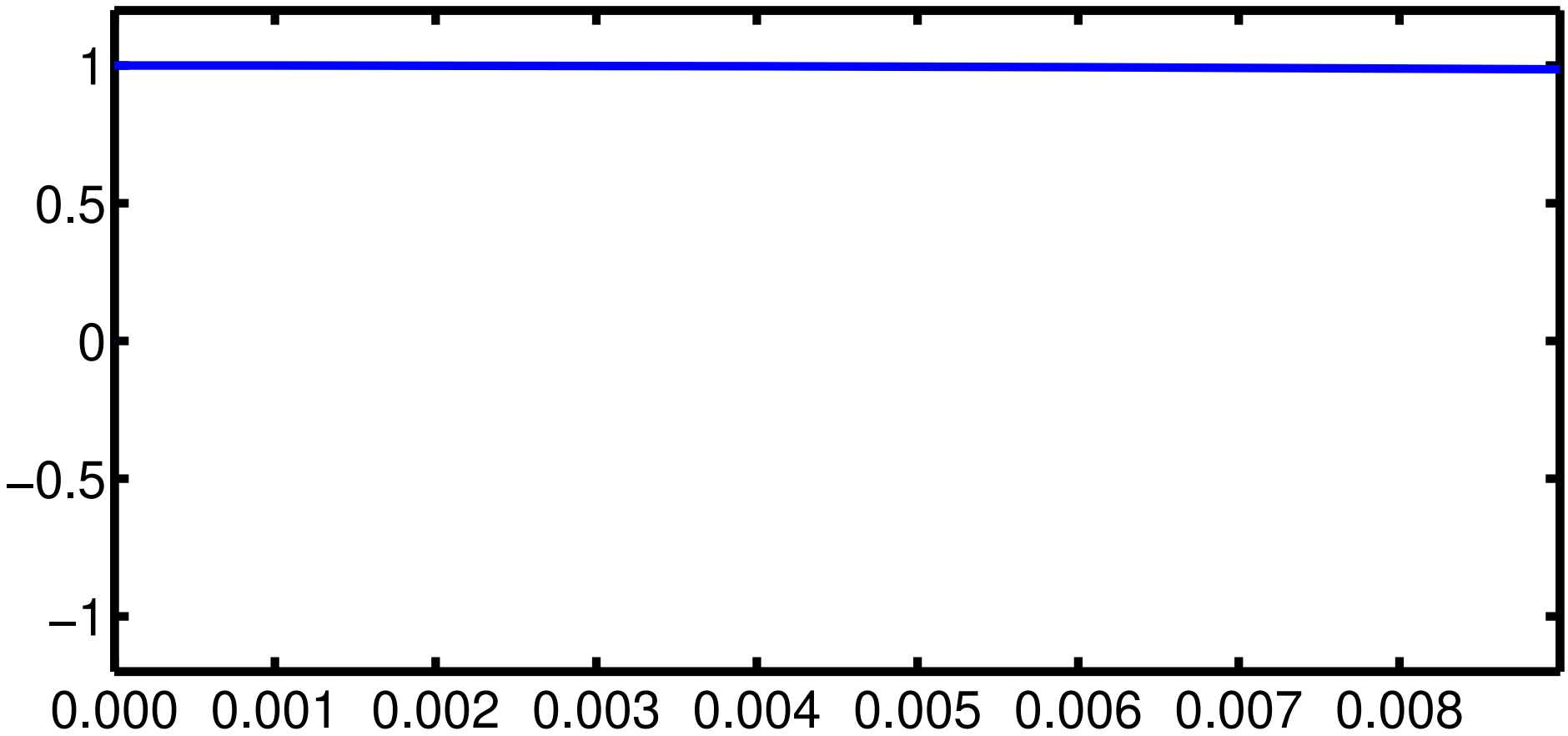
Approximation

- Finer quantization $\Rightarrow$ better approximation
- More details later



Approximation

- Longer data sets $\Rightarrow$ better approximation
- More details later



Sampling

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

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

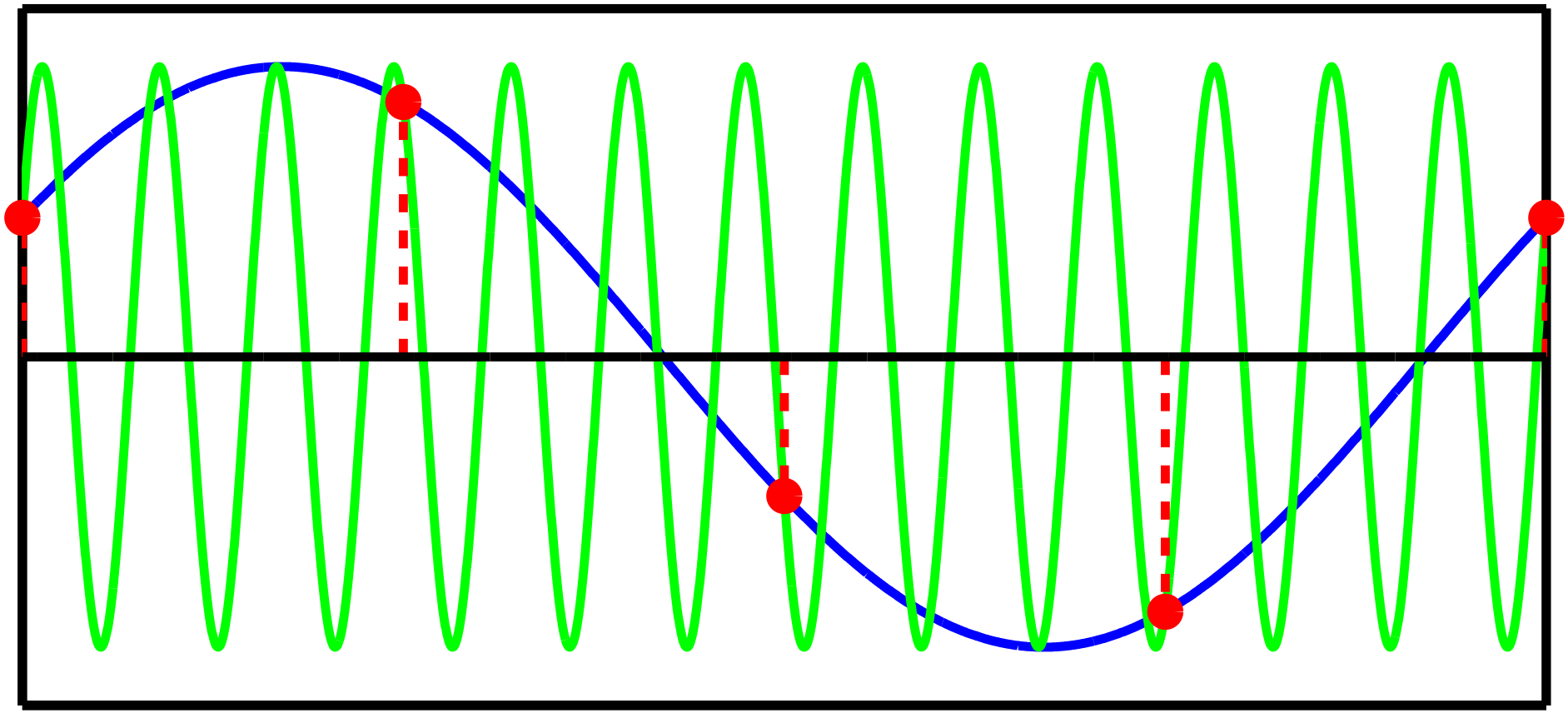
where t_s is the **sampling interval**

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

e.g., sampling frequency for CDs is 44.1 kHz

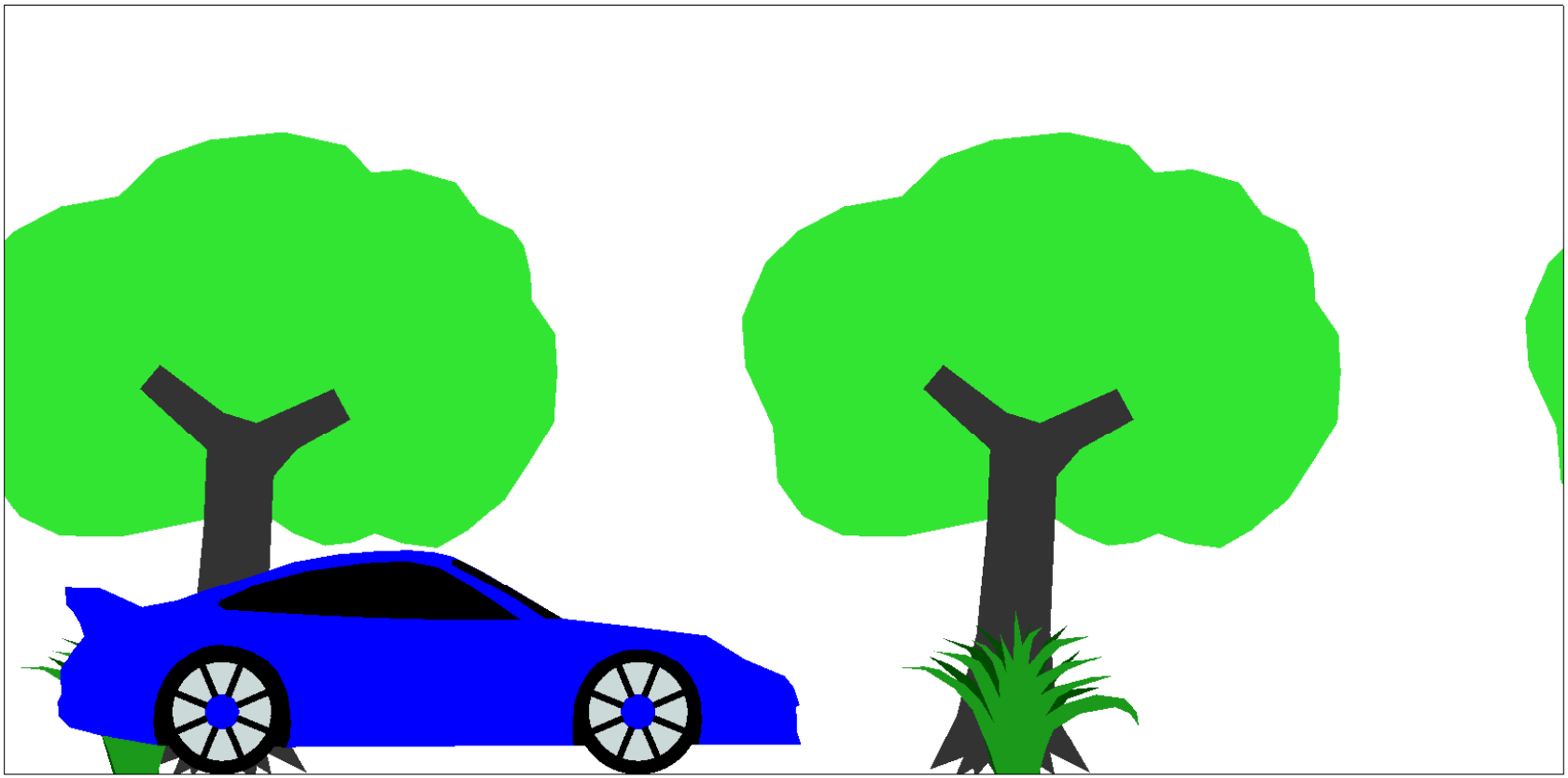
Aliasing

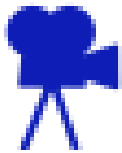
A critical issue for sampling is **aliasing**



An ∞ number of possibilities...

Another example of aliasing





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.*,

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

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

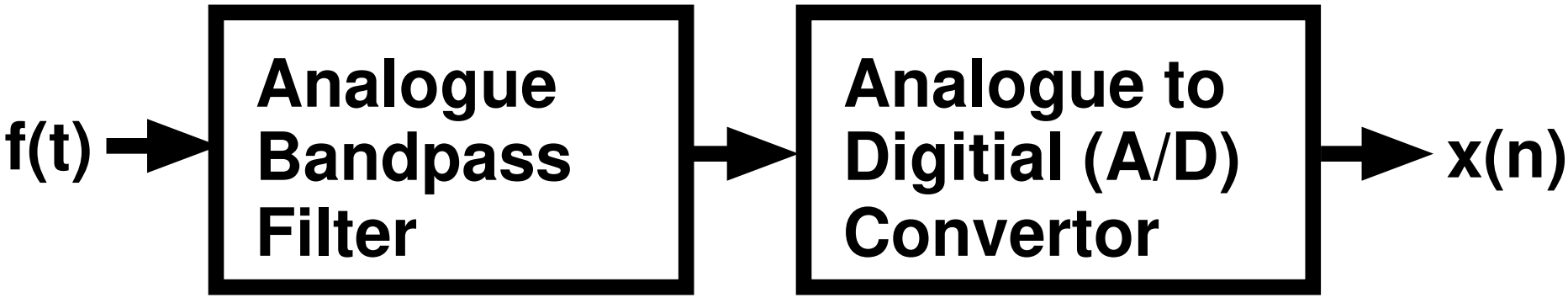
Nyquist sampling theorem

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.

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.

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,

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

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.
□

Shannon interpolation

Reconstruction of original signal from IFT

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

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

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

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.

Do It

Use Colab to

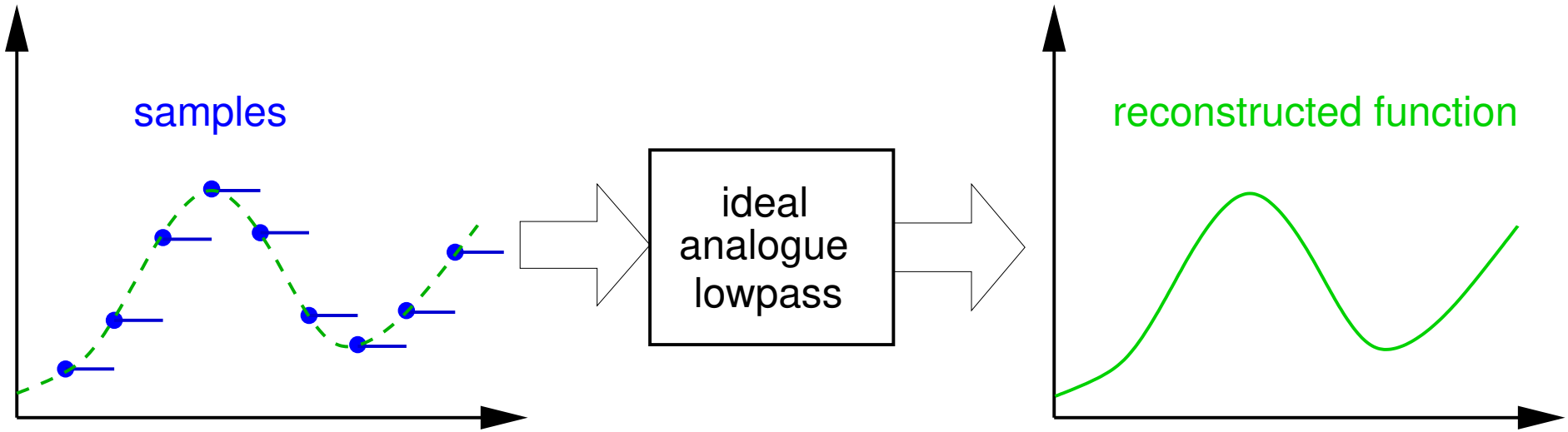
1. Create two sampled signals. Both will be sin waves, one with frequency 10 Hz, and the other with frequency 35 Hz. The sampling frequency should be 25 Hz. Plot the two on semilog axes.
2. Take a Discrete Fourier transform of the two, and examine the squared magnitude of the result (the power spectrum). What do you notice?
3. Use Shannon interpolation to reconstruct the signals, plotting them at a 10x finer time resolution than the original sampling.
4. Now try changing the sampling frequency to something higher than $2\times$ the second frequency.

Have a look at `shannon_interp.ipynb` to see how to do this.

Digital to Analogue converter

Interpretation

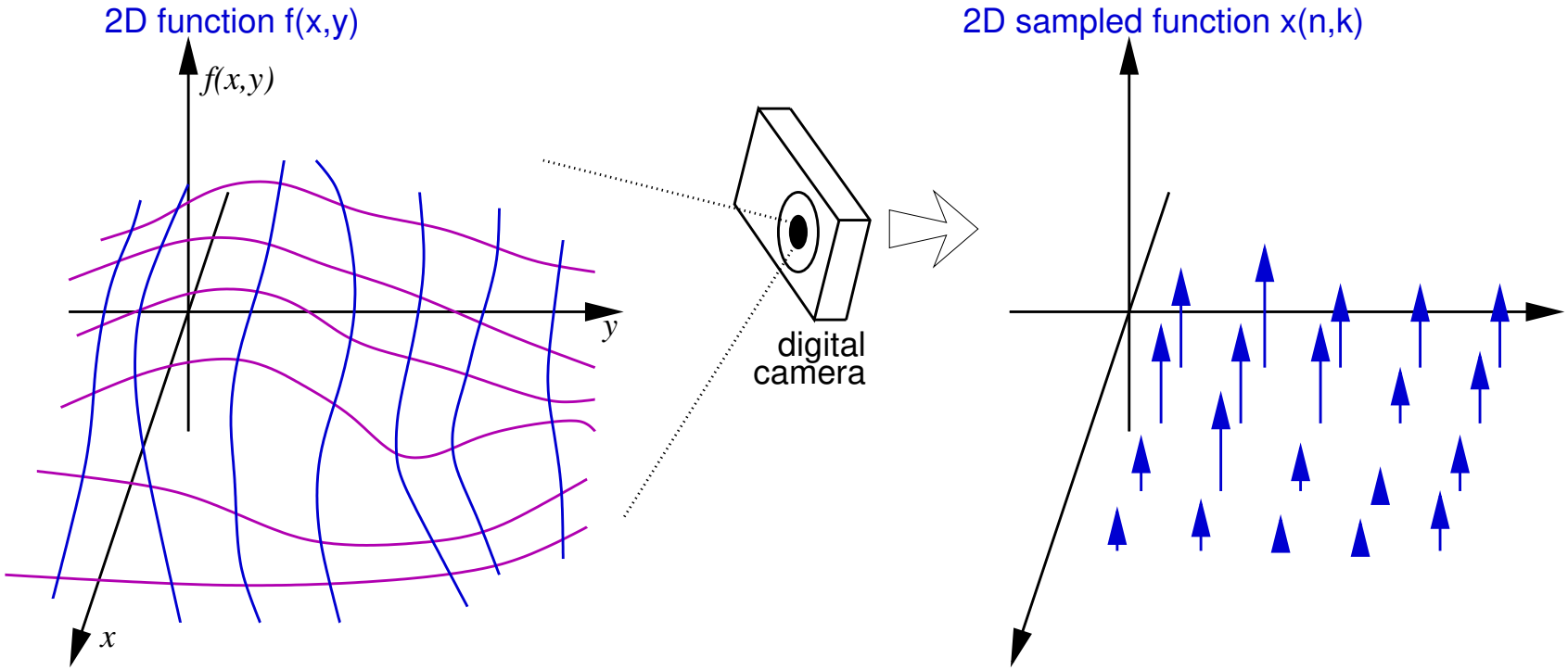
- convolution with sinc
- equivalent to ideal analogue low-pass filter



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

2D signals = images

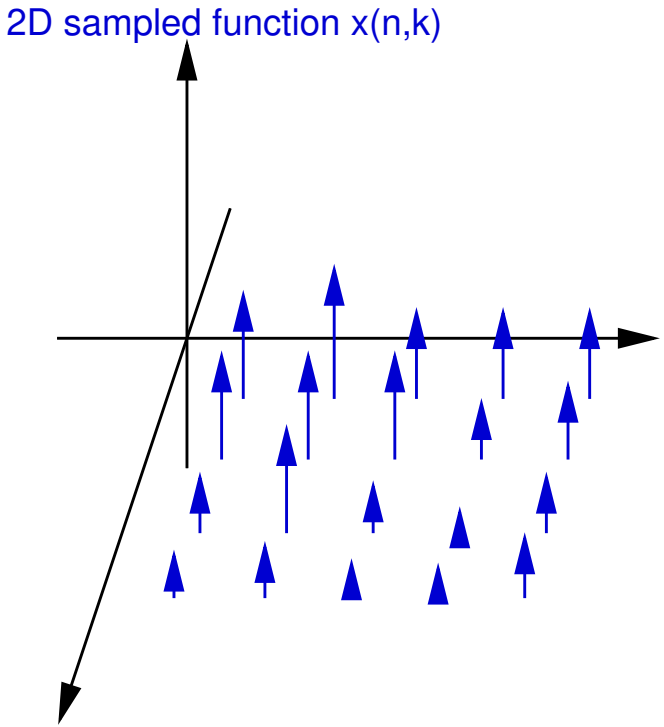
- 2D signal processing is used for images
- images are sampled signals in 2D

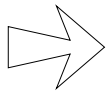


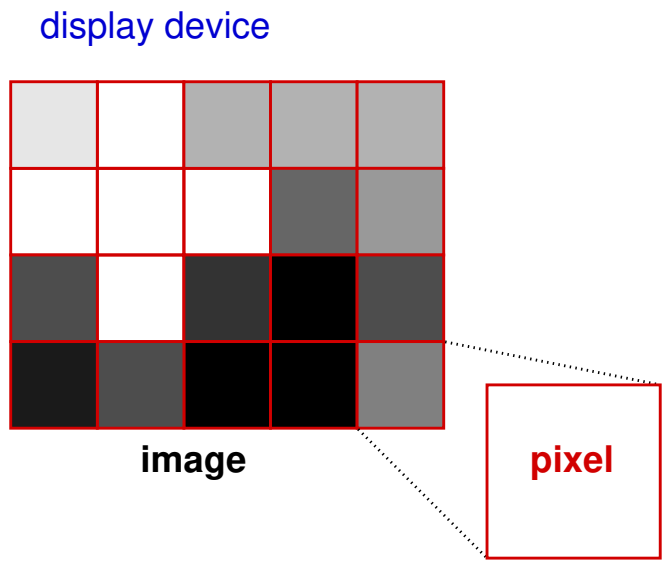
- same issues for sampling/quantization as we have for 1D signals

Displaying 2D signals

- 2D signal processing is used for images
- images are sampled signals in 2D







Aliasing in images

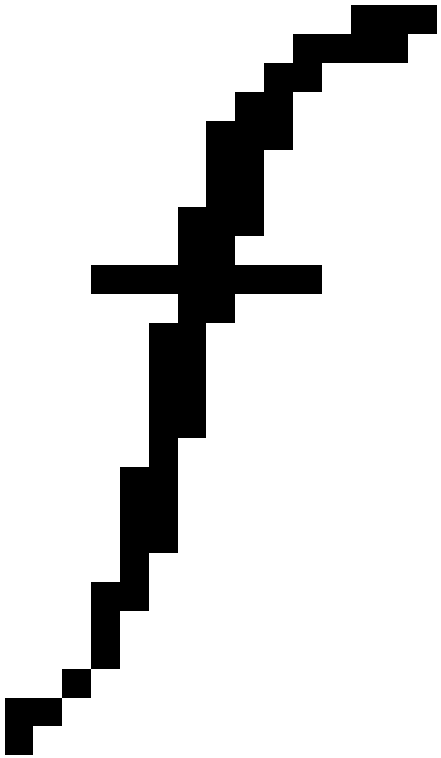
Aliasing in images is similar to that in time signals.

- an image is a sampled spatial field
- cameras average over a small angle for each pixel, so effectively low-pass the image field before sampling.
- CGI generation by sampling of underlying mathematical model.
- produces jagged edges in images "jaggies"
- Moire patterns

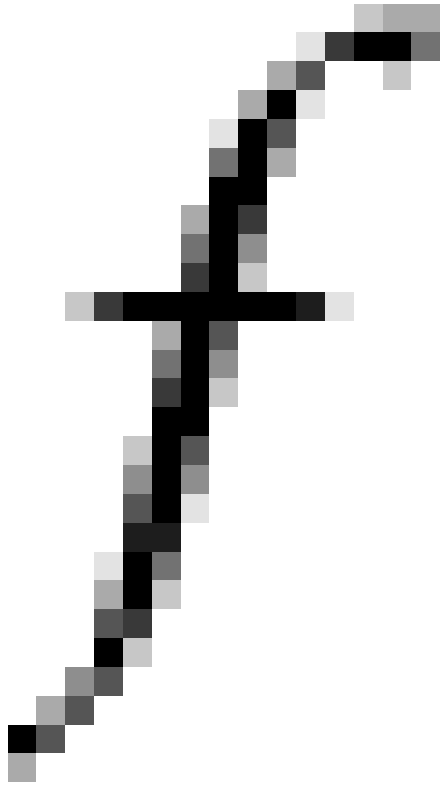
Solution is low-pass prefiltering of data (as before).

Jaggies and Anti-aliased fonts

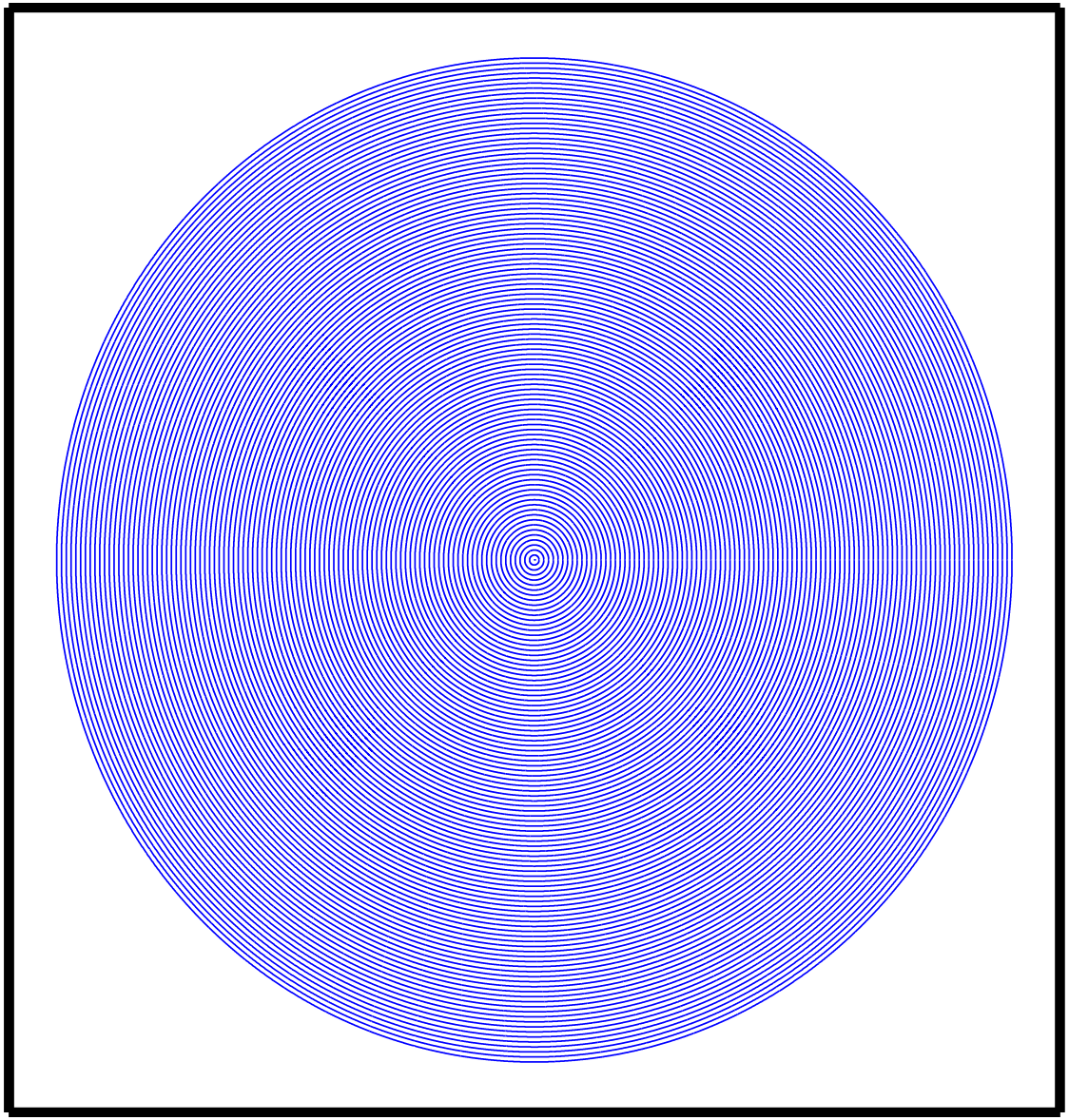
Aliased



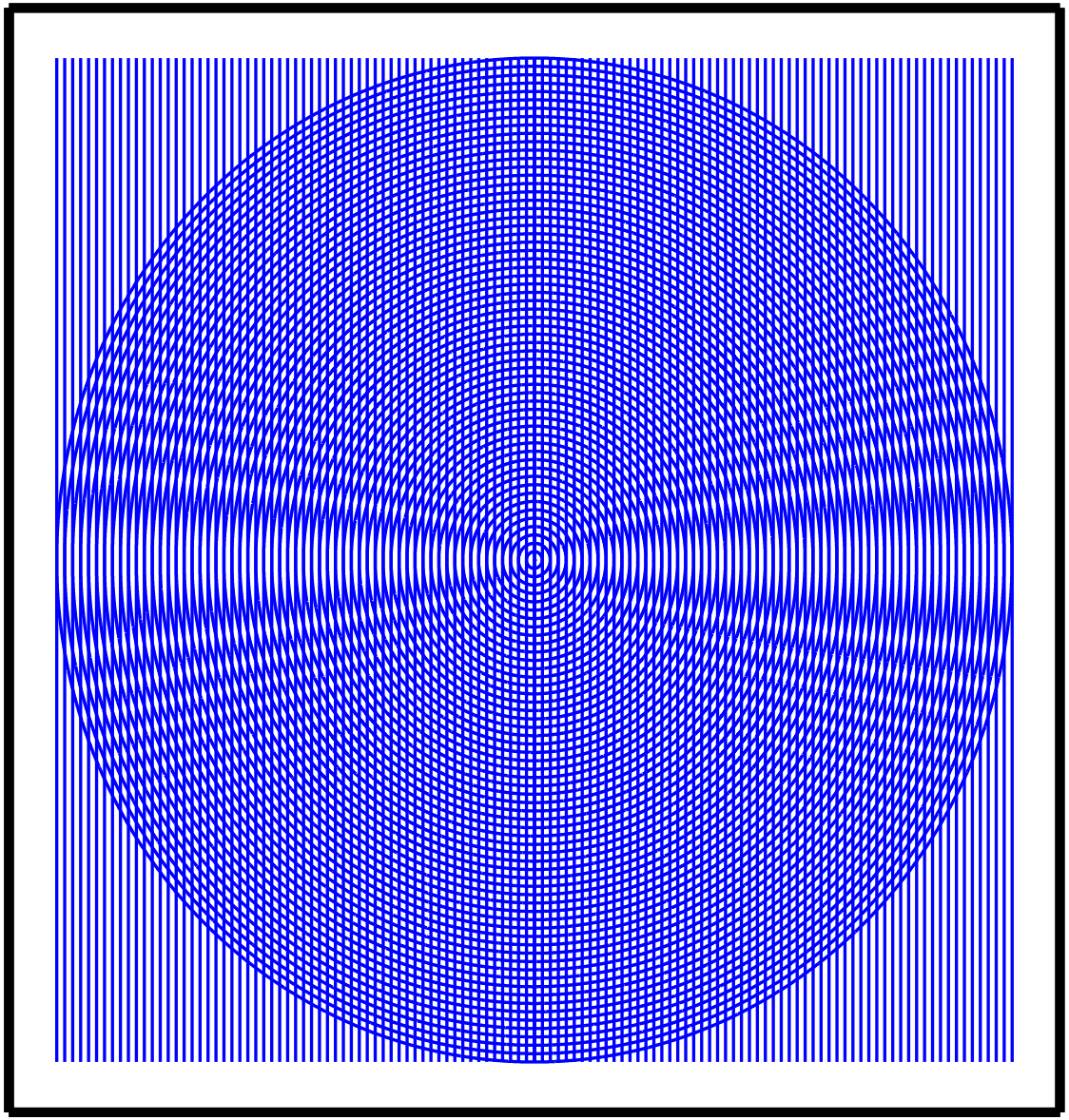
Anti-aliased



Moire patterns



Moire patterns



Anti-aliased CGI

Computer-Generated Images (CGI)

- one way to generate is **raytracing**
- generate a ray for each image pixel, and trace its path, including reflections, and refraction.
- computationally expensive (there are faster but less exact methods), so don't want to generate more than one ray per pixel
- but jagged edges move (somewhat randomly), generating marching ants.
- to get **good** results need to **oversample**, and average (*e.g.*, a low-pass operation)

Aliasing: crawling ants

Resampling applications: video

There are many variations of display for images

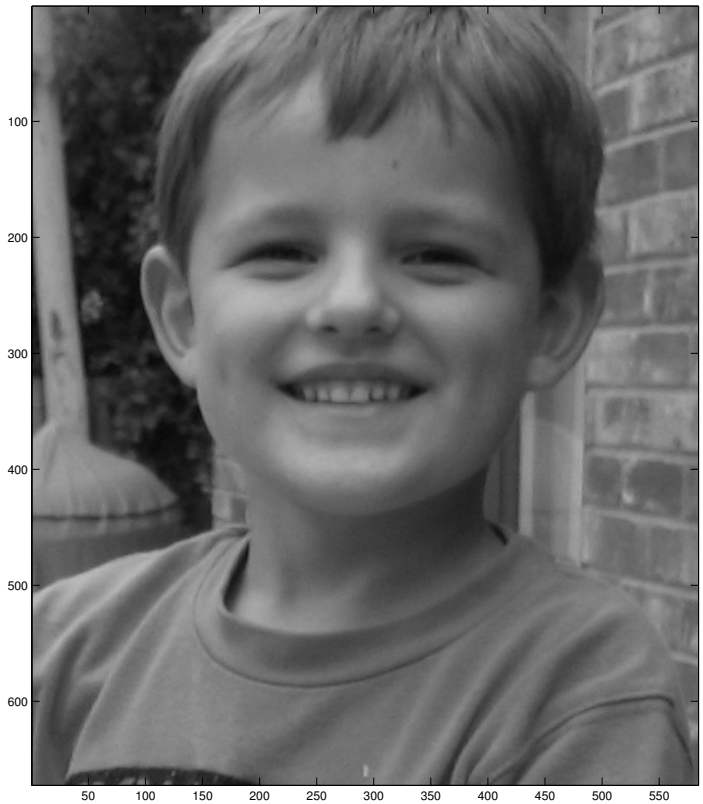
Display	**width $\times$ height**	**Depth**	**rate**
VGA monitor	640×480	4 bit	60 Hz
PAL TV	720×576 (625)	-	50 Hz
HDTV 1080i	1920×1080	8 bit	50 Hz
Plasma TV (typical)	1024×768	14 bit	variable
Film	$\sim 3000 \times \sim 2000$	~ 12 bit	24 Hz
Workstation	1920×1200	24 bit	60 Hz
B&W laser printer	6600×5500	1 bit	-

We need to be able to convert between these (*e.g.*, if your standard DVD player is hooked up to a high-def plasma screen).

Resampling applications: printing

Problem of resampling has been around for a long while

- Printers put ink on a page
  - think of this as 1 bit quantization (ink, or no ink)
  - so how can you do a picture, with only 1 bit?



becomes

Newsprint

Newspapers came up with solutions many years ago

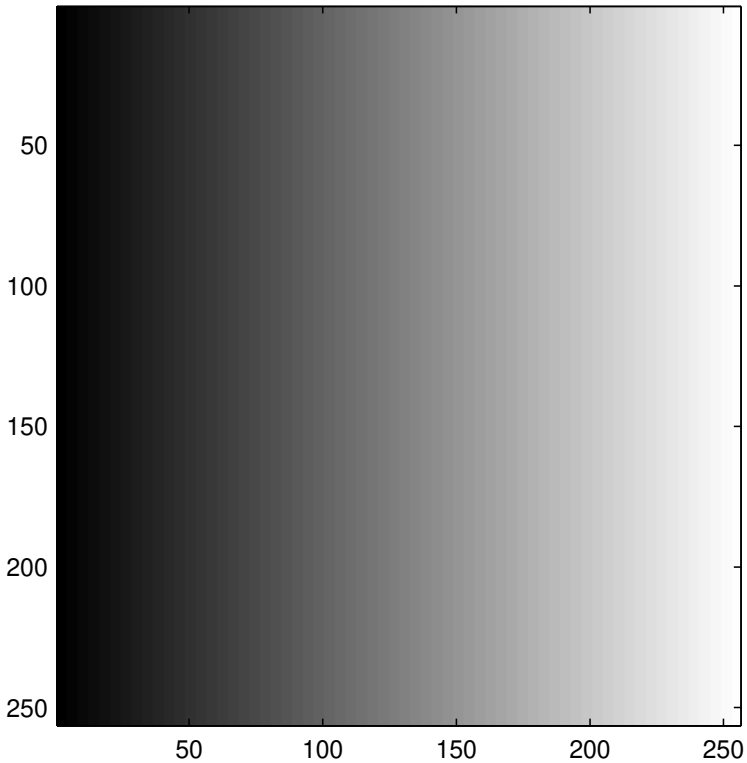
from the Australian, Dec 11th, 2005

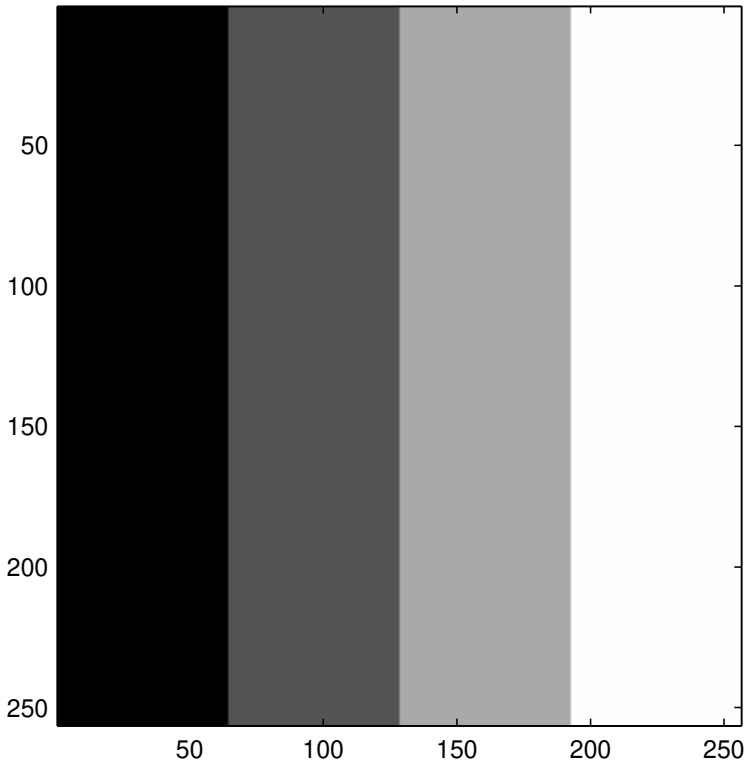
Resampling applications: printing

- many printers have only 1 bit quantization
  - *e.g.*, B&W laser printers
- ink jets (typically) have 3 or 4 color inks
  - but you can't mix them, so need to put together somehow?
- printers typically have better spatial resolution than a monitor
  - *e.g.*, 1200 dpi (dots per inch)
  - compare to a monitor with perhaps 72 dpi
- so we tradeoff sampling vs quantization
  - it works because our eyes (or other sensors) effectively do a low pass before sampling

Half-toning

- problem is that we have limited quantization



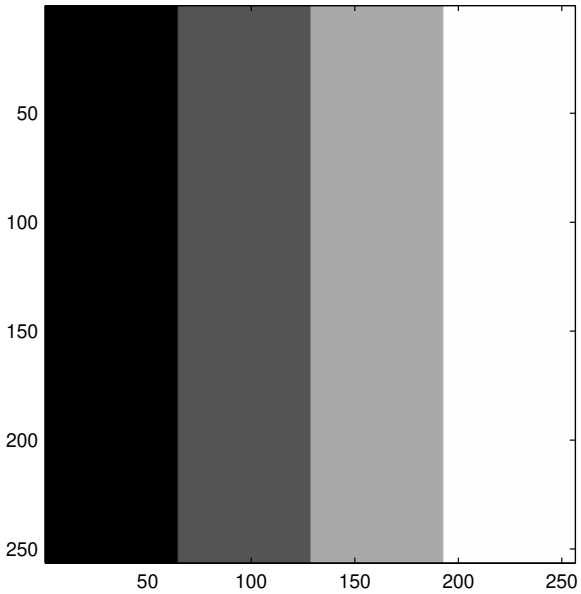


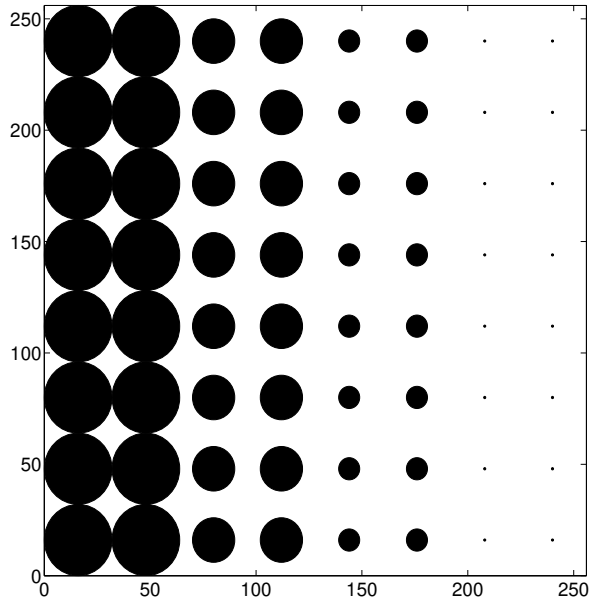
- exchange spatial resolution for quantization
- processing is called half-toning (for printing)

Halftoning

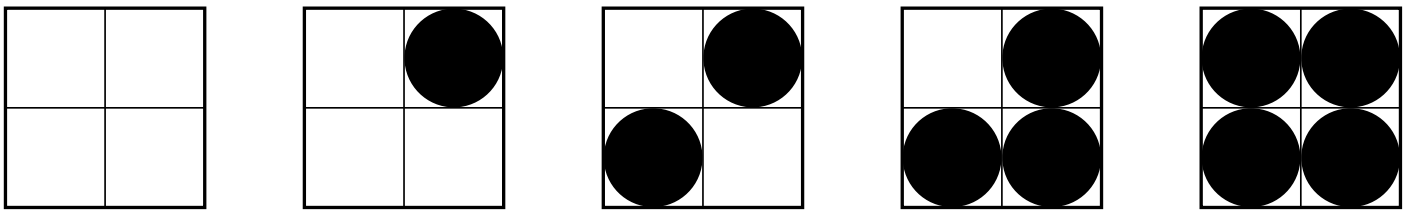
Processing is called half-toning (for printing)

- use dots of varying size





- use patterns of dots



level 1 2 3 4 5

Dithering

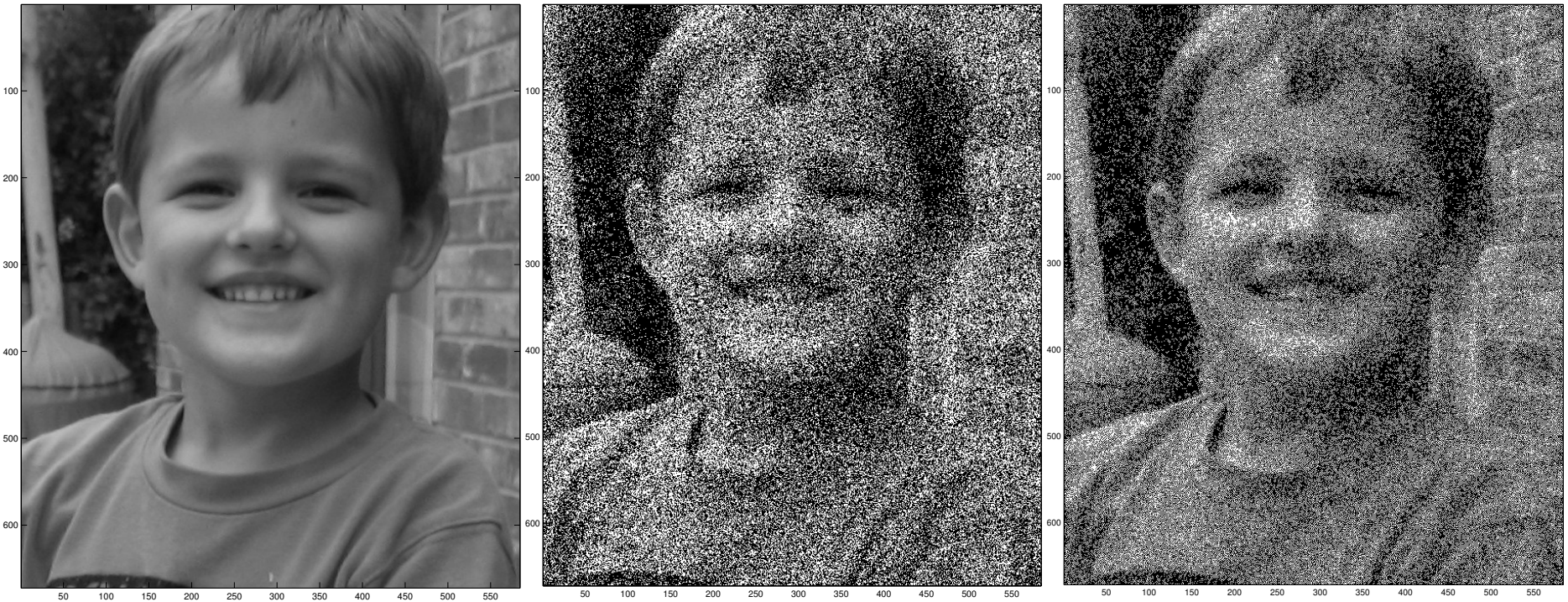
Problem with simple approaches above:

- introduce some patterns into the image
- these might be perceptable
- want to add some randomization
- example:

$$P(x, y) = \text{round}\left[I(x, y) + \text{rand}(x, y)\right]$$

- Example applet
  http://www.markschulze.net/halftone/

Examples of Dithering

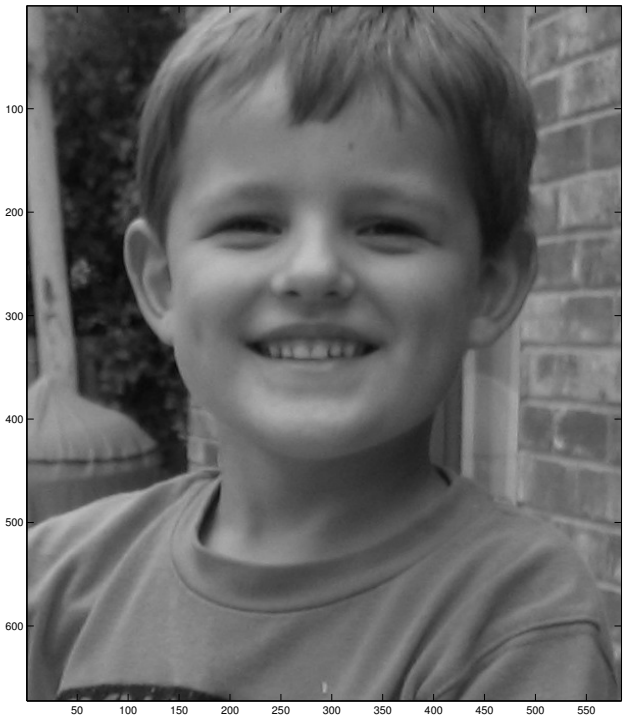


1. original image
2. dithered by adding a random value
3. Floyd-Steinberg error diffusion dithering

Extreme example: ascii art

- Sample the image down to grayscale with less than 8-bit precision, and then assign a character for each value.
- Ideally character is related to grayscale
- also shape can be used to help define lines.

Extreme example: ascii art



$\Rightarrow$

```
=IZm#mmWWQWQQQW&>+-:::==vvSnXYY1uXXXSX1isilvvSX1><ii|:=|vv
|vZ#mBWWWWWWWWY~:===||indvvnovooonu1o1iv32owoXuI><|i=+=%v}
oomWWWWWWBWQB(<suqX1vvn*+<omqm#XmXXqZXoXqqZ#Bmmoc||||=ii+=
({$WWWWWWWW#nommZ1|li|:::<mWmmm#X21XZo#mBWWWWWWQmoo2>=il|s
;+3WWWWWWWW&m#WBl|||+++||vXTYYY1lIvvXXX#mmWWWWWBBm2I|vvvvv
:=]QWWQWWmWm#BWel||+|||||||+|||iiIvn2XXZ#BWWWWWWWW1nSSovnn
;;<QWQWQWWW#Wm#ei=:;==||||||||ivvIvIII1X#mWWWQWWWEin1***+|
>:<$WWQWWmmmWBZ+=;|ivXXZZXv|ivXmZ##ZXwwwX##WWQWWWk><==>=|l
c:=dWWQWQW1v$#>;<vwm#mWBmZ+:-<X#mWWWWmmm####WWQWWZvviiiiiI
C:;3QWWQWEX#SS;:--~+"!!"+:: .=iI*YYYYYYYSX##WWWWWmovvv<nns
h=>)WmWWQQ2evd>::.....:_===_=iiuna>;:;==ivo##mWm###1nvsi2no
Q=1=WWWWWWm%il====i%uoX1IX#mZmm#XXXuauoqX###m##XX+==<vvvvs
W>+=3QWWWWWWQa;;|vIvSXmasaawXmmmqwwXZXXZ#U###2XXe=:==|i||ll
Xc;=]WWWWWWWWQp|||iI1YVVX11v13XX#UVS1nXZZ##YXXv>=><%ii>+++
?^=+IWBBWBWWQWWoilil|=:===|illvvviilvXZ##m#=n1i=<=|||i=+++
:====*?TV$QWQWWWsivvi;::--=-=-++=inomX#mmB#=vi|:)ss>|isiIi
;=====|=ivd$WWQ#liiInna>||||iiaawm##mWWWmm#=v%|:)S21vXXXZo
sssvaouwwqmm#WWW>{ollvnXXqqqqmm#mWmWWWWBmWE|2o|:=|IIi*l*Yv
nnauXqd###ZVTT*l|}n%ivnnXZ##mmmBWBWWWWBBW#XoX1|;=innovvvv|
XX1Y!""~..iu2}===aB||v2XXX##X##mmBmm###Z#ZZSoonsiiII1vvvi|
+=:.;;==uZY|=:;:<#(-=<1IllIISZ####X#Z#Z#X2onv1li|||||i=<iv
i|==a|nX2<||====<nas>>|i|svuqqqXXZZXXSooo1vno1llvaonli||I1
+:vZ1oY+iv>==++||ilv2XUU###XXXSXXXX21no1vuZ2lnomZSvii|vlll
aomXva=vl+=|+|=|||IIvnnnoooXXXX2onlvnoIumS1lomZYlisvIvsvnS
Sm#mZiv|=+|=++=|+|ilIvvvoXXSnnvn1ivoIvwZ1ivd#XonnoXS2113#v
XZmZ|||=+||+||=+||ivnoXXXnnvvvvl|v1|vZSulvdZZYsuX2nvvilnZh
dZ#oawwwwpwwwwwwwumqXX1IlIvvI|||%i<w#nZoXqmSvoXX11ili|vXX#
```