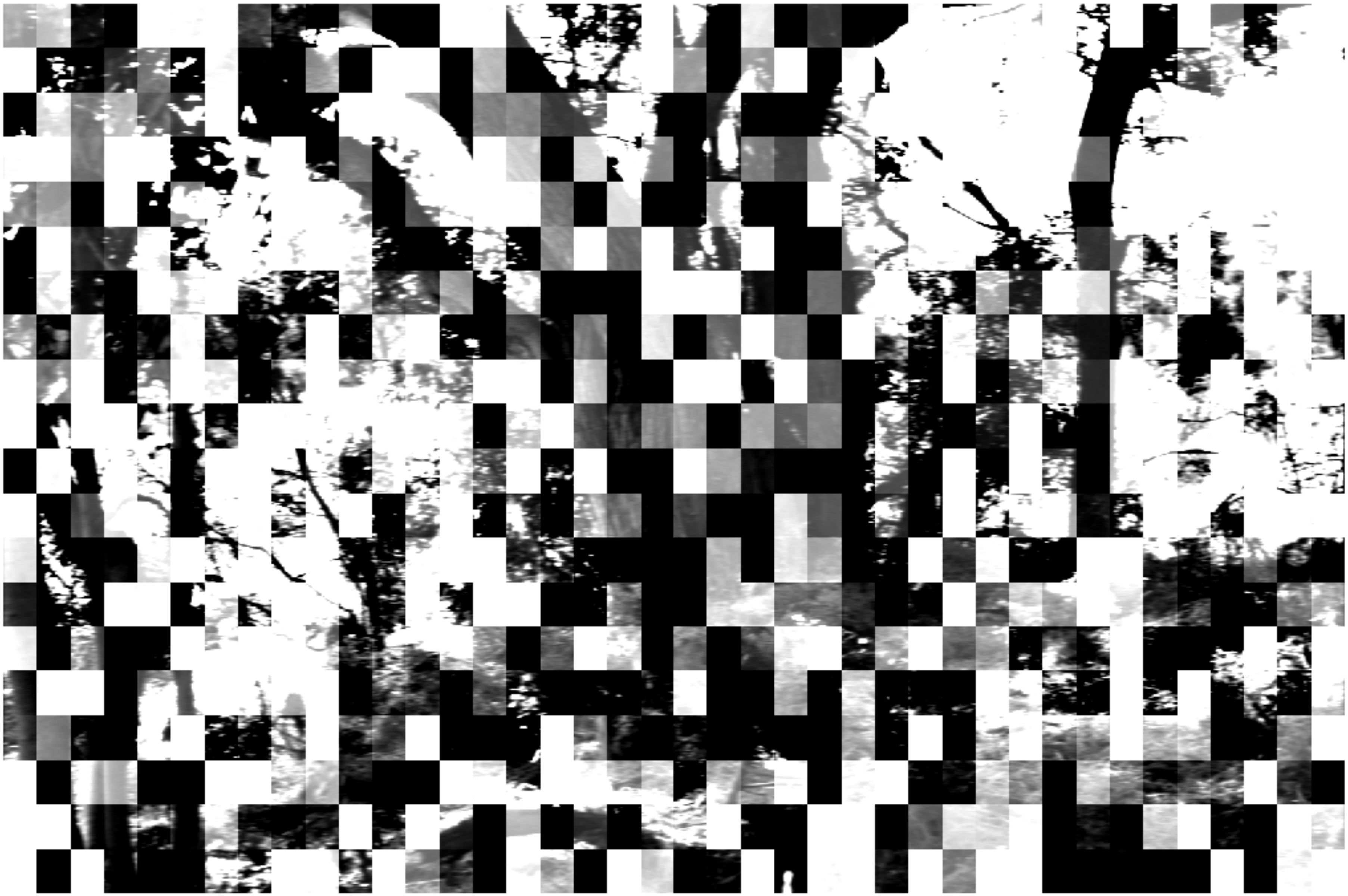


Untitled, 2016

Tkacik Natural Image database convolved with Gaussian and Hadamard multiplied by block random matrices in MATLAB. In memory of Ellsworth Kelly.

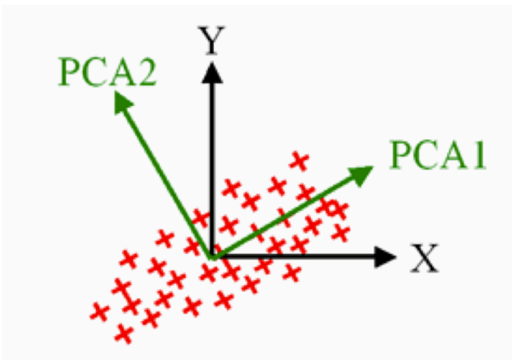
Lecture 4: Fourier Transforms, Filtering, and Convolutions

February 2nd, 2017

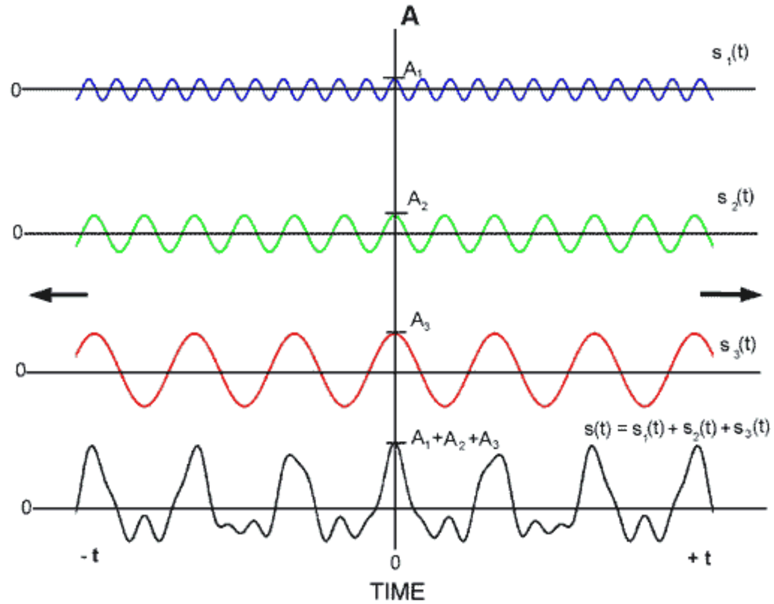
Lane McIntosh & Kiah Hardcastle

Math Tools for Neuroscience

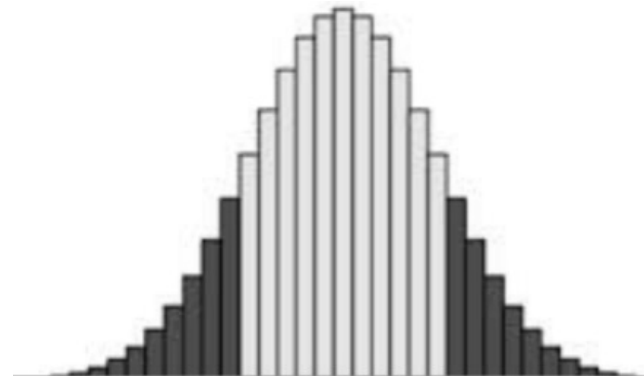
Topics we will cover this quarter



dimensionality reduction



Fourier transforms, convolutions, and filtering out noise

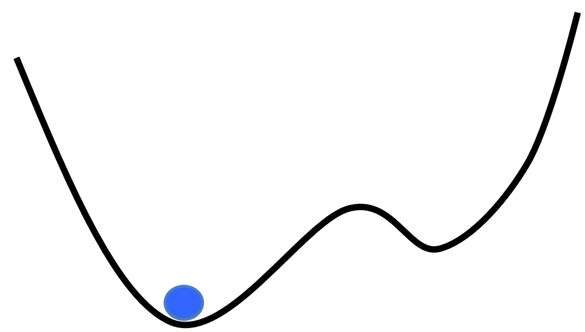


statistics, Bayesian probability, and information theory

$$C_m \frac{dV}{dt} = -g_L(V - E_L) + g_L \Delta_T e^{\frac{V-V_T}{\Delta_T}} - u + I$$

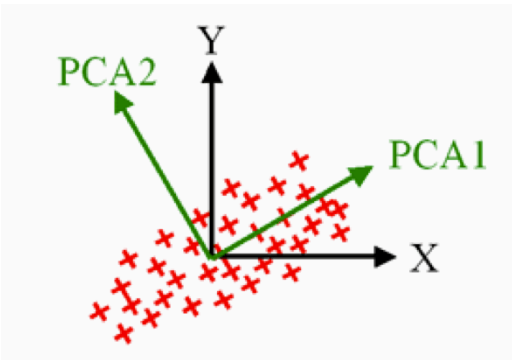
$$\tau_w \frac{du}{dt} = a(V - E_L) - u$$

differential equations and modeling

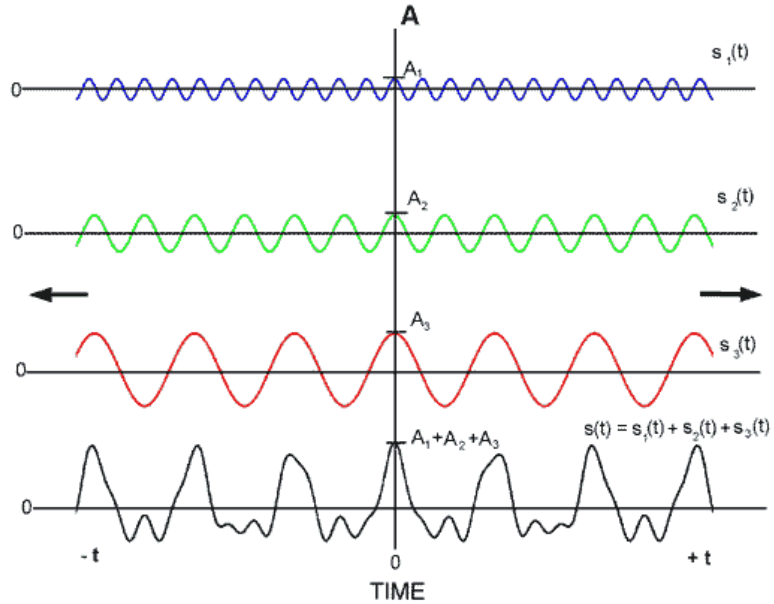


optimization

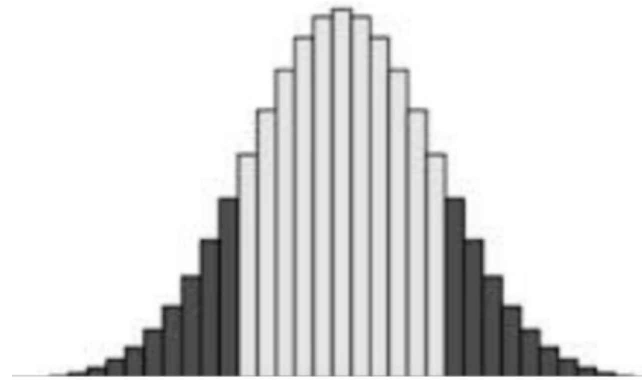
Topics we will cover this quarter



dimensionality reduction



Fourier transforms, convolutions, and filtering out noise

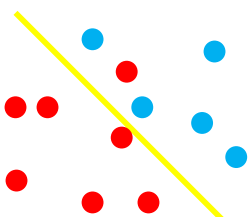


statistics, Bayesian probability, and information theory

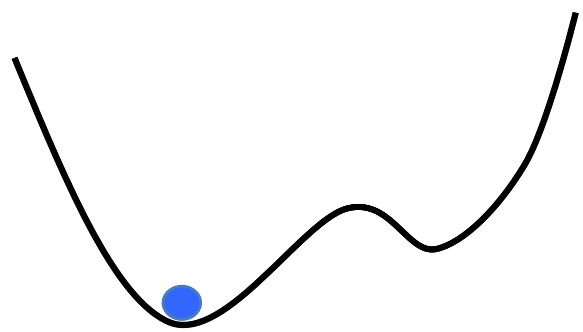
$$C_m \frac{dV}{dt} = -g_L(V - E_L) + g_L \Delta_T e^{\frac{V - V_T}{\Delta_T}} - u + I$$

$$\tau_w \frac{du}{dt} = a(V - E_L) - u$$

differential equations and modeling

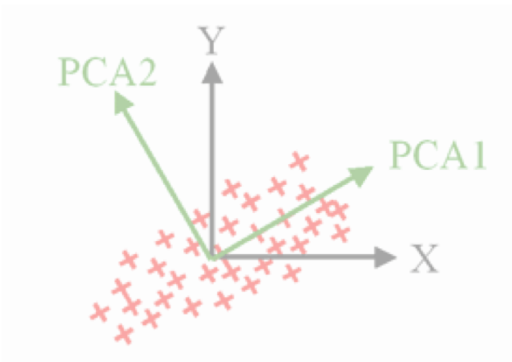


statistical models

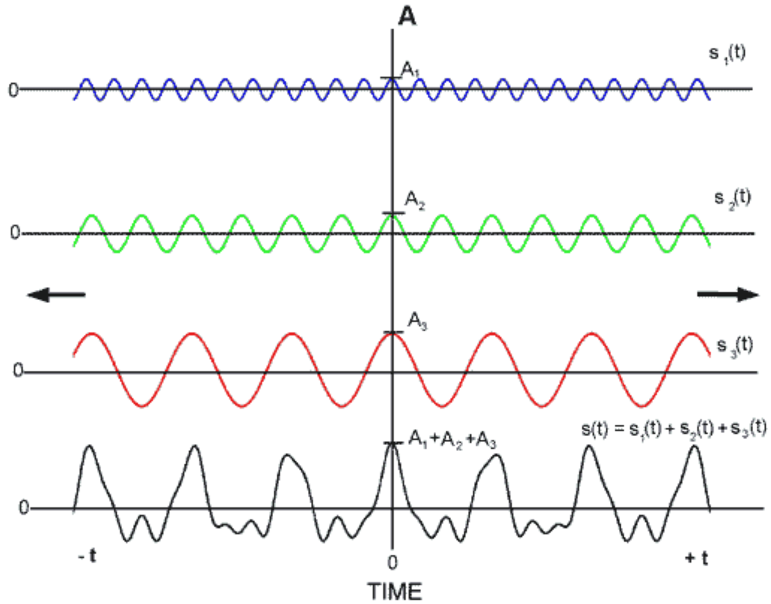


optimization

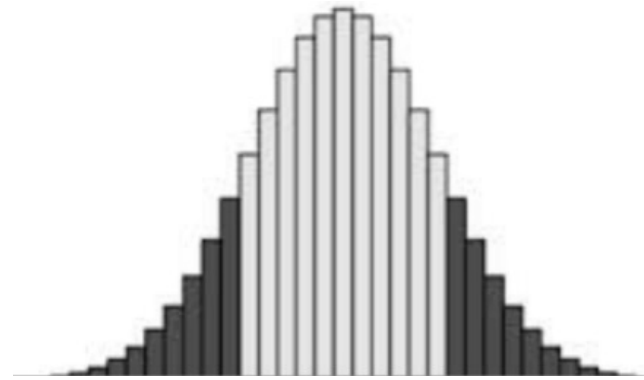
Topics we will cover this quarter



dimensionality reduction



Fourier transforms, convolutions, and filtering out noise

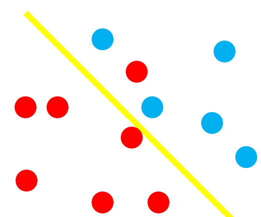


statistics, Bayesian probability, and information theory

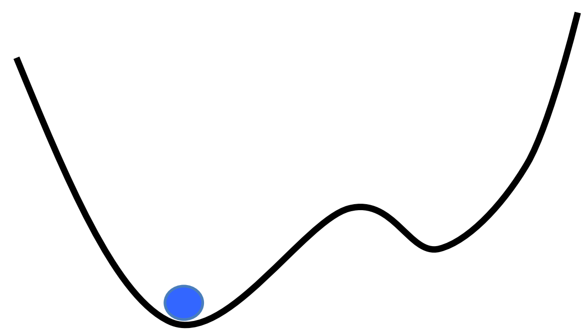
$$C_m \frac{dV}{dt} = -g_L(V - E_L) + g_L \Delta_T e^{\frac{V-V_T}{\Delta_T}} - u + I$$

$$\tau_w \frac{du}{dt} = a(V - E_L) - u$$

differential equations and modeling



statistical models



optimization

Today’s lecture

When will this be useful?

Filtering in time or space

- What is convolution?
- How much should I filter?

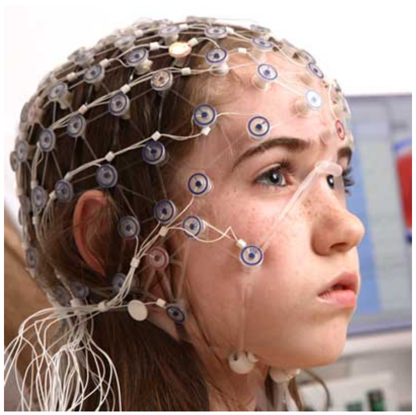
Filtering in frequency

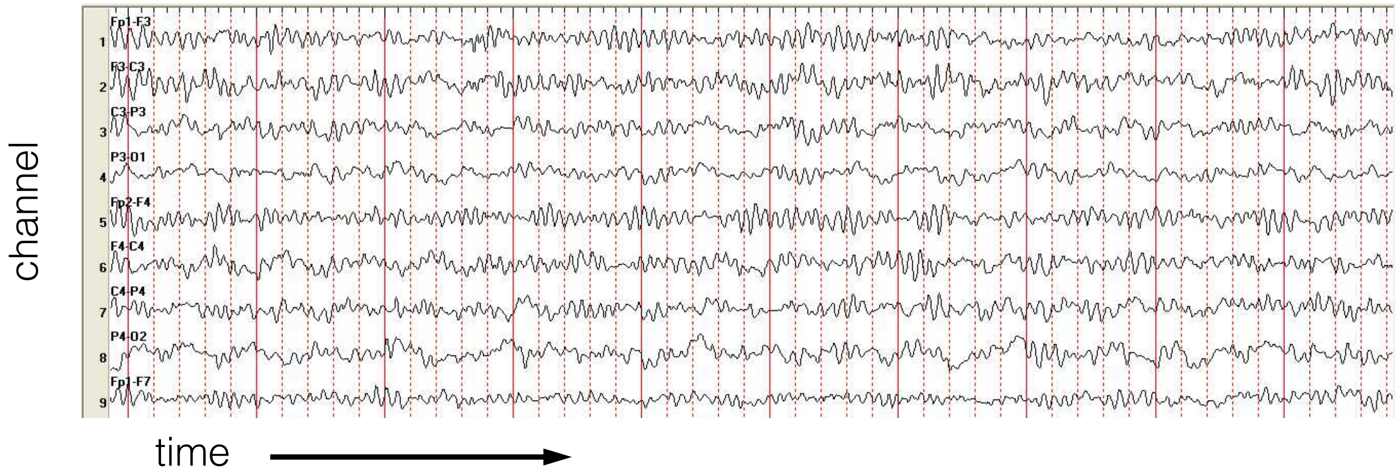
- Fourier series
- What is a Fourier transform?
- Designing filters in frequency domain

Experimental examples of filtering

Experimental examples of filtering

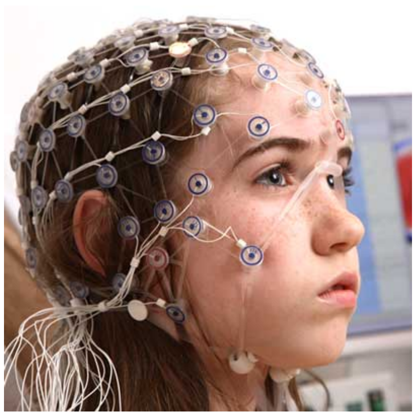
Experimental noise



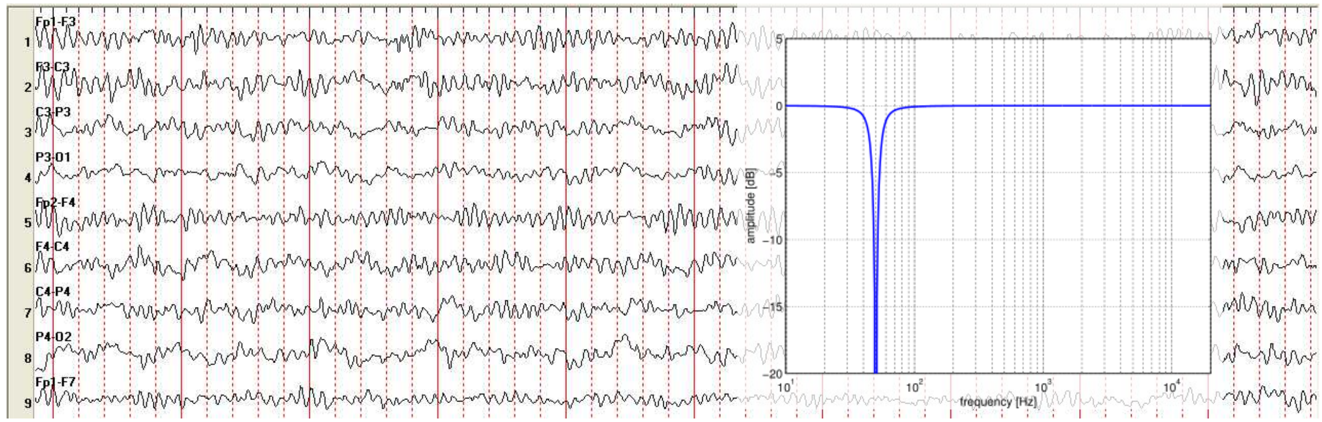


Experimental examples of filtering

Experimental noise

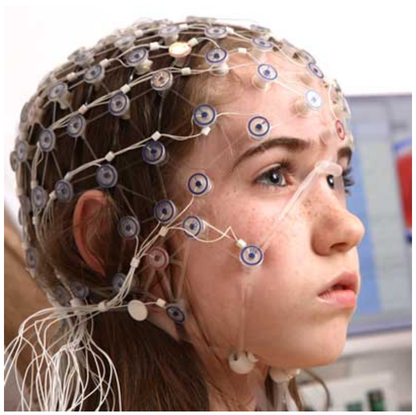


channel



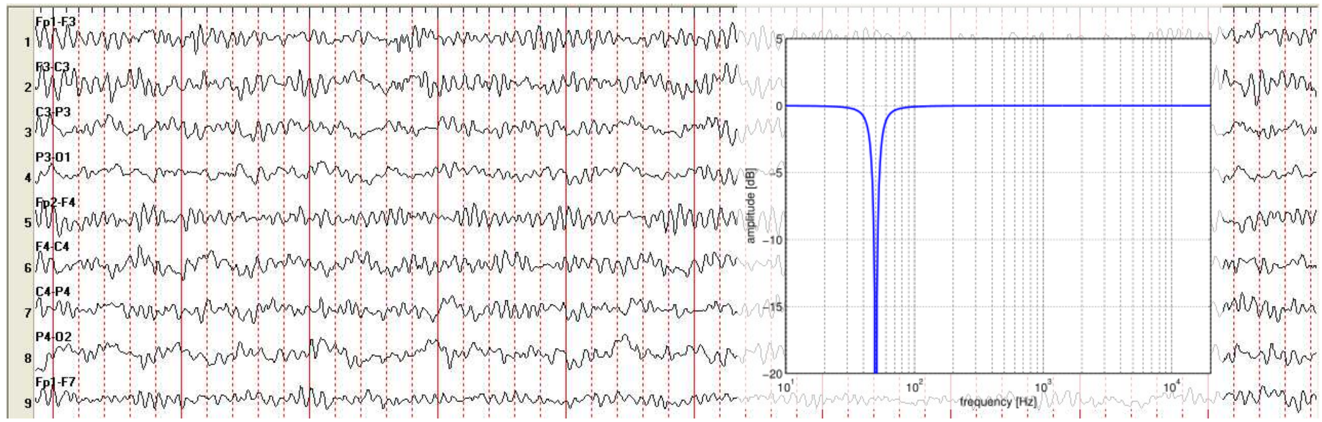
time

Experimental examples of filtering

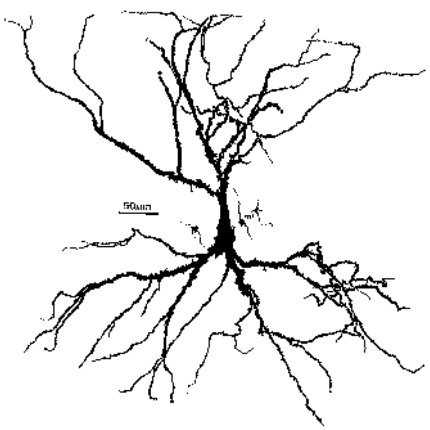


Experimental noise

channel

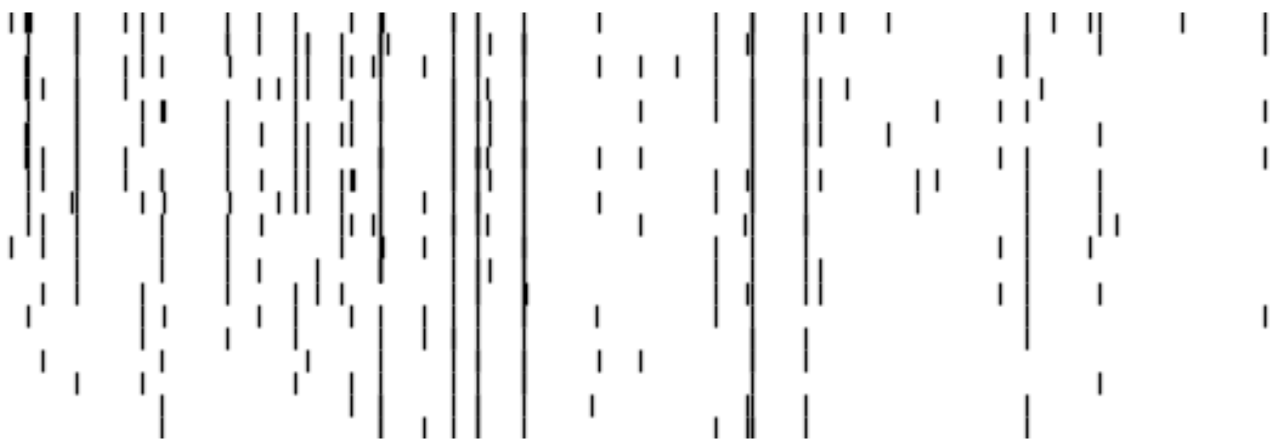


time



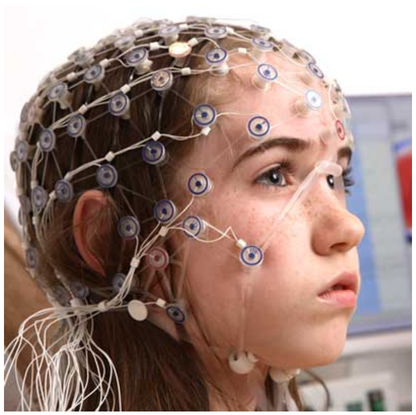
Biological noise

trial



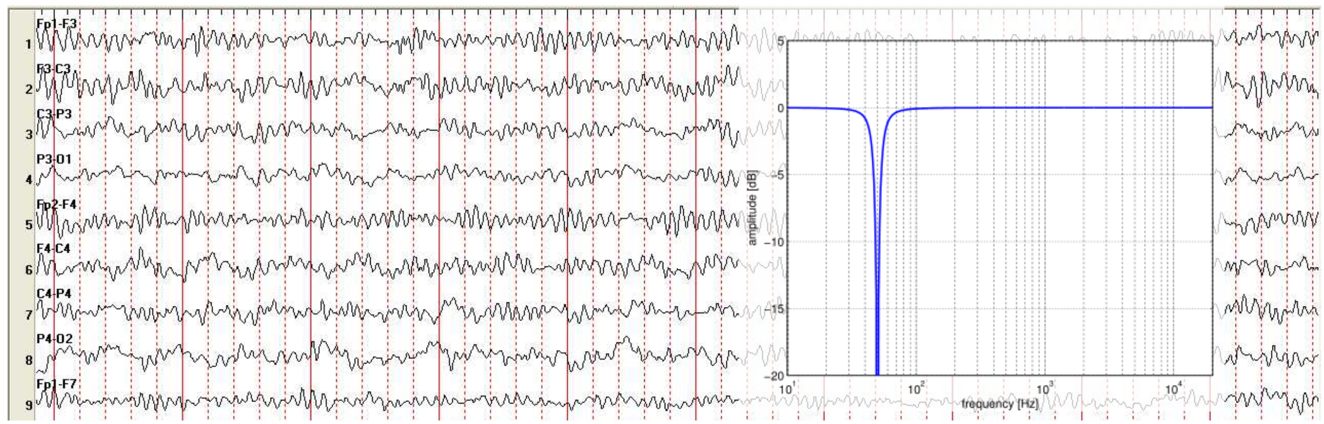
time

Experimental examples of filtering

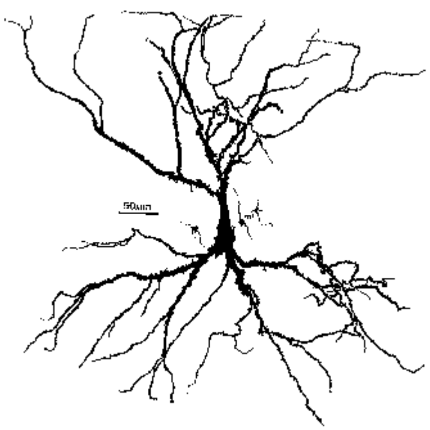


Experimental noise

channel

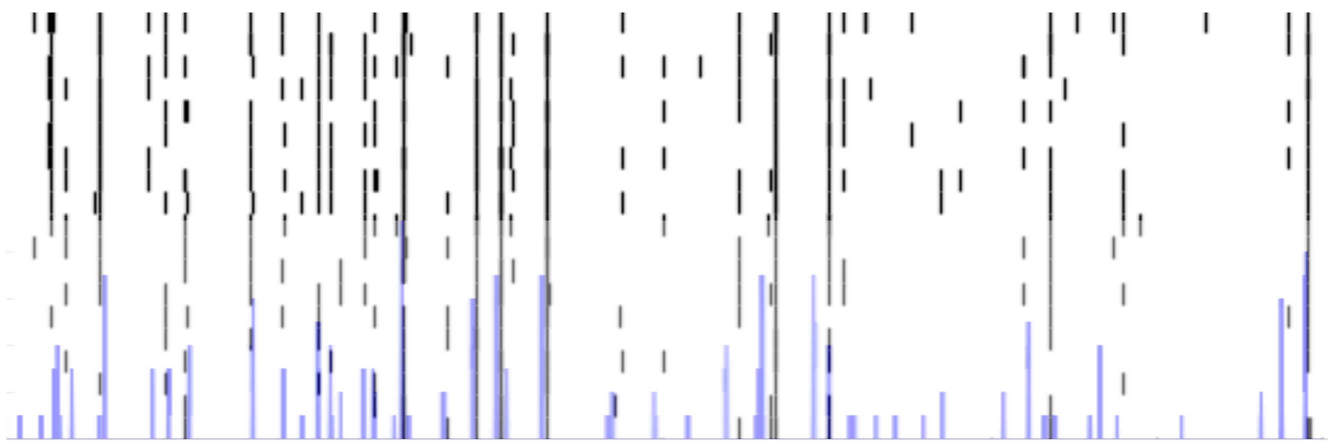


time →



Biological noise

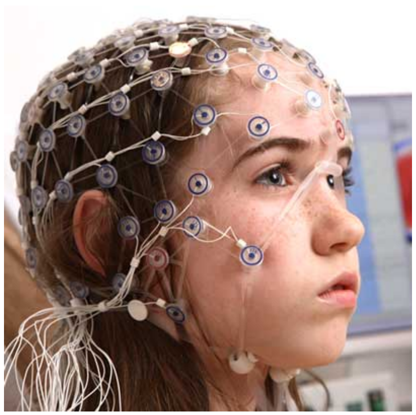
average firing rate



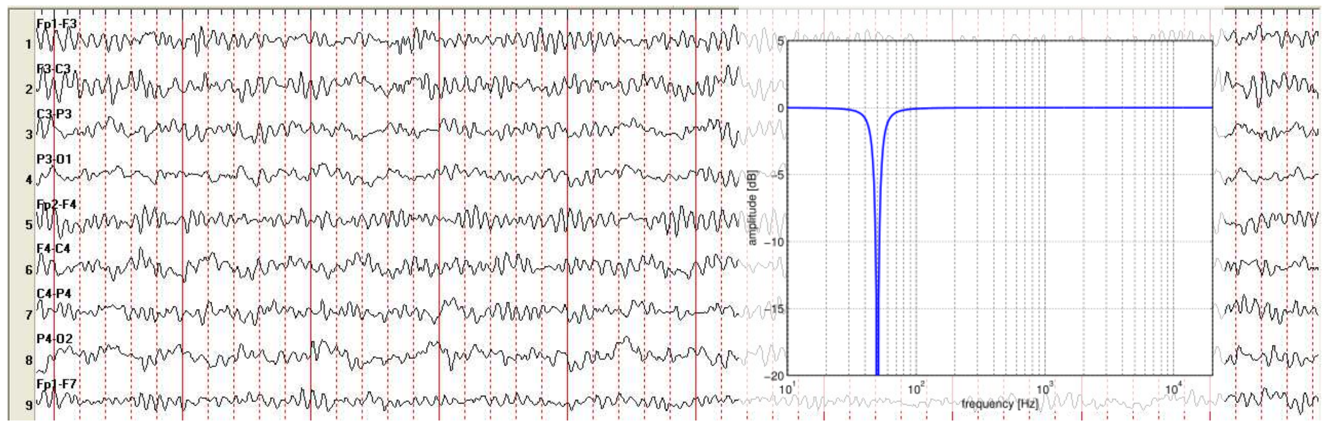
time →

Experimental examples of filtering

Experimental noise

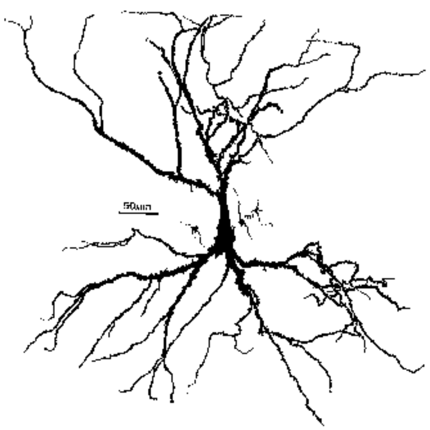


channel

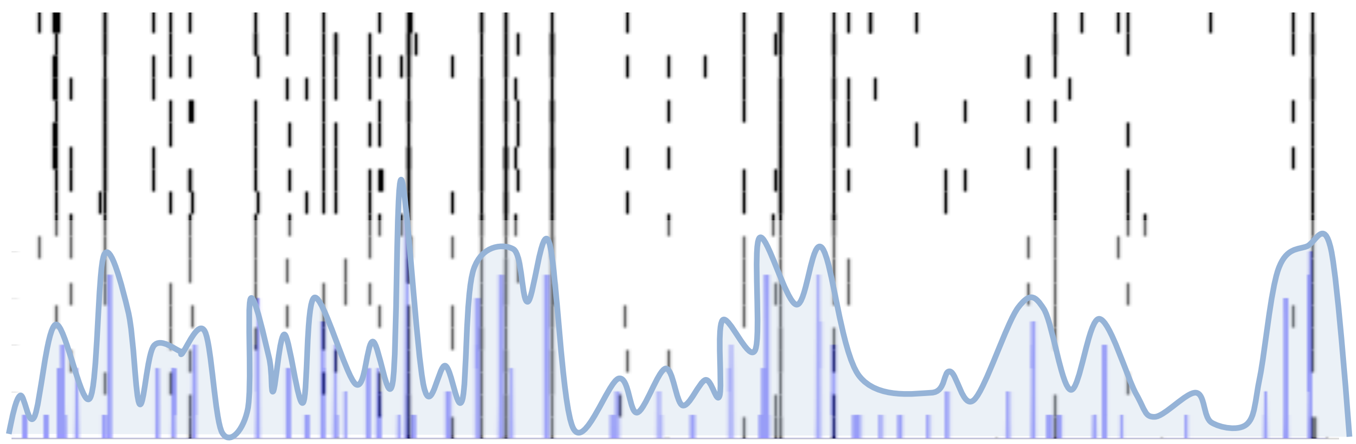


time

Biological noise



average firing rate



time

Today's lecture

When will this be useful?

Filtering in time or space

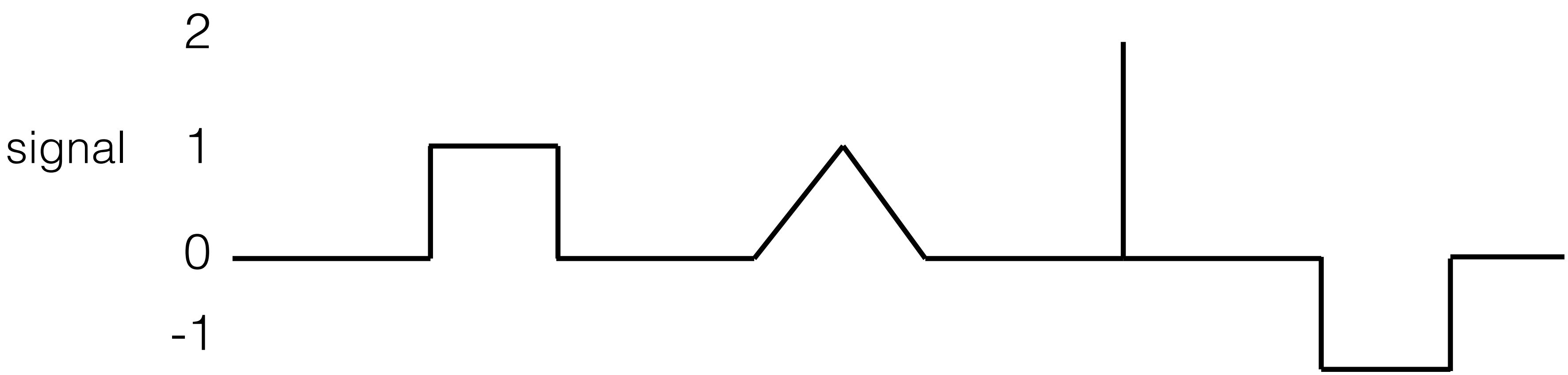
- What is convolution?
- How much should I filter?

Filtering in frequency

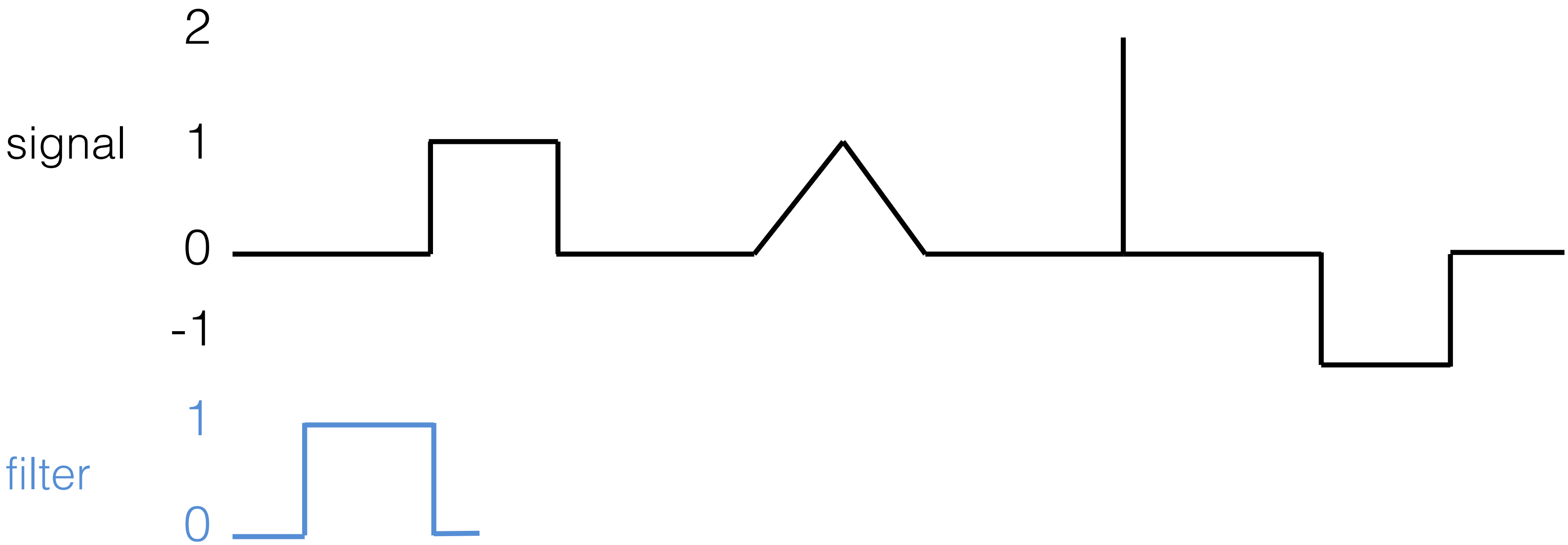
- Fourier series
- What is a Fourier transform?
- Designing filters in frequency domain

Convolution

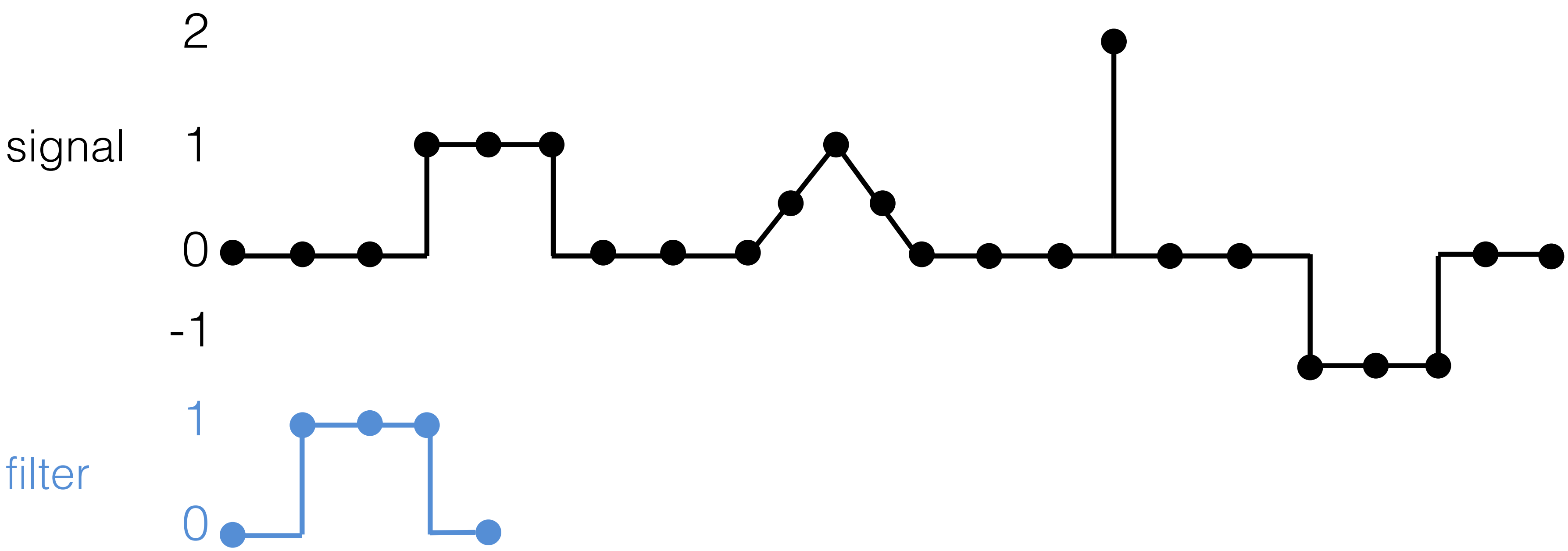
Convolution



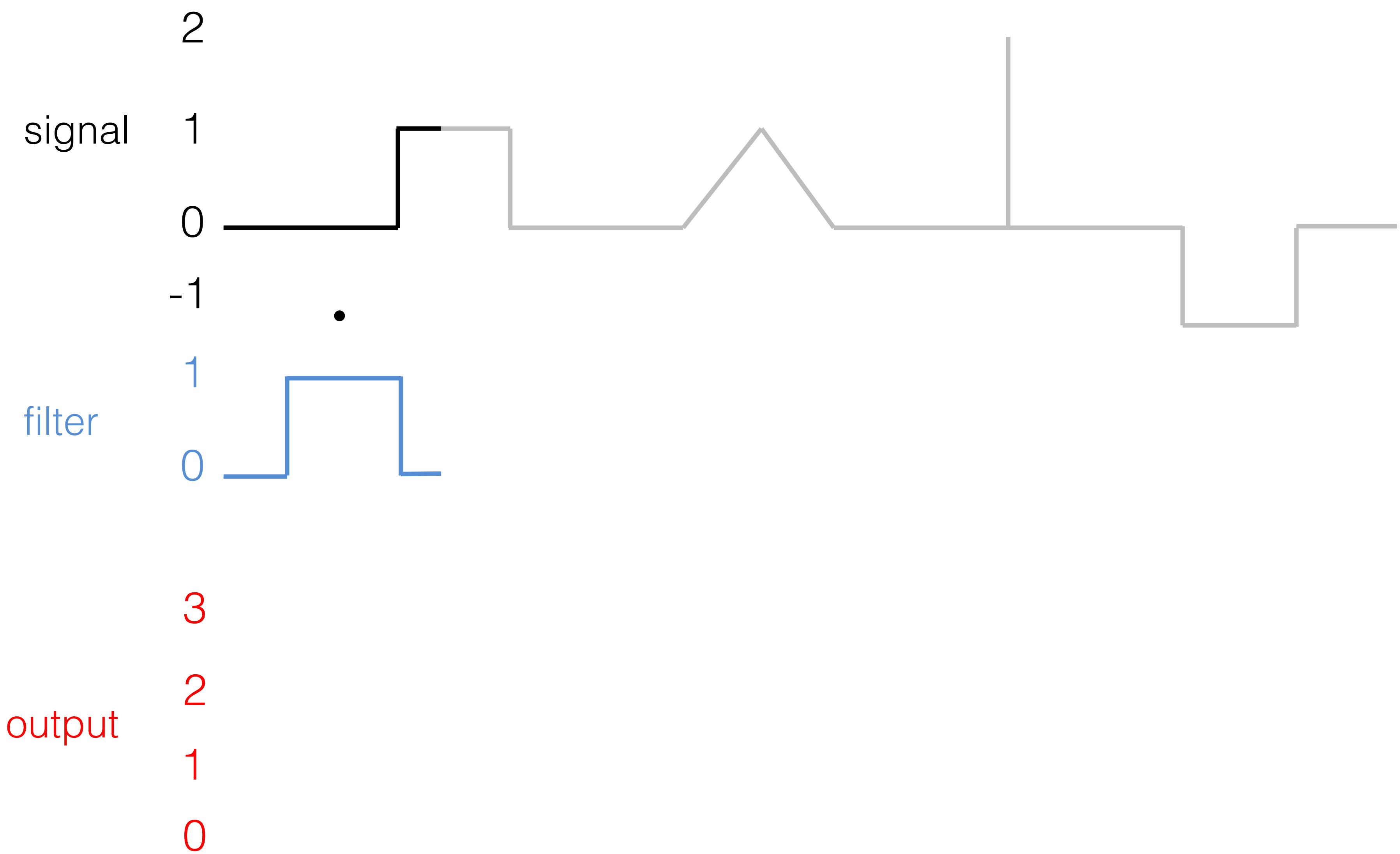
Convolution



Convolution

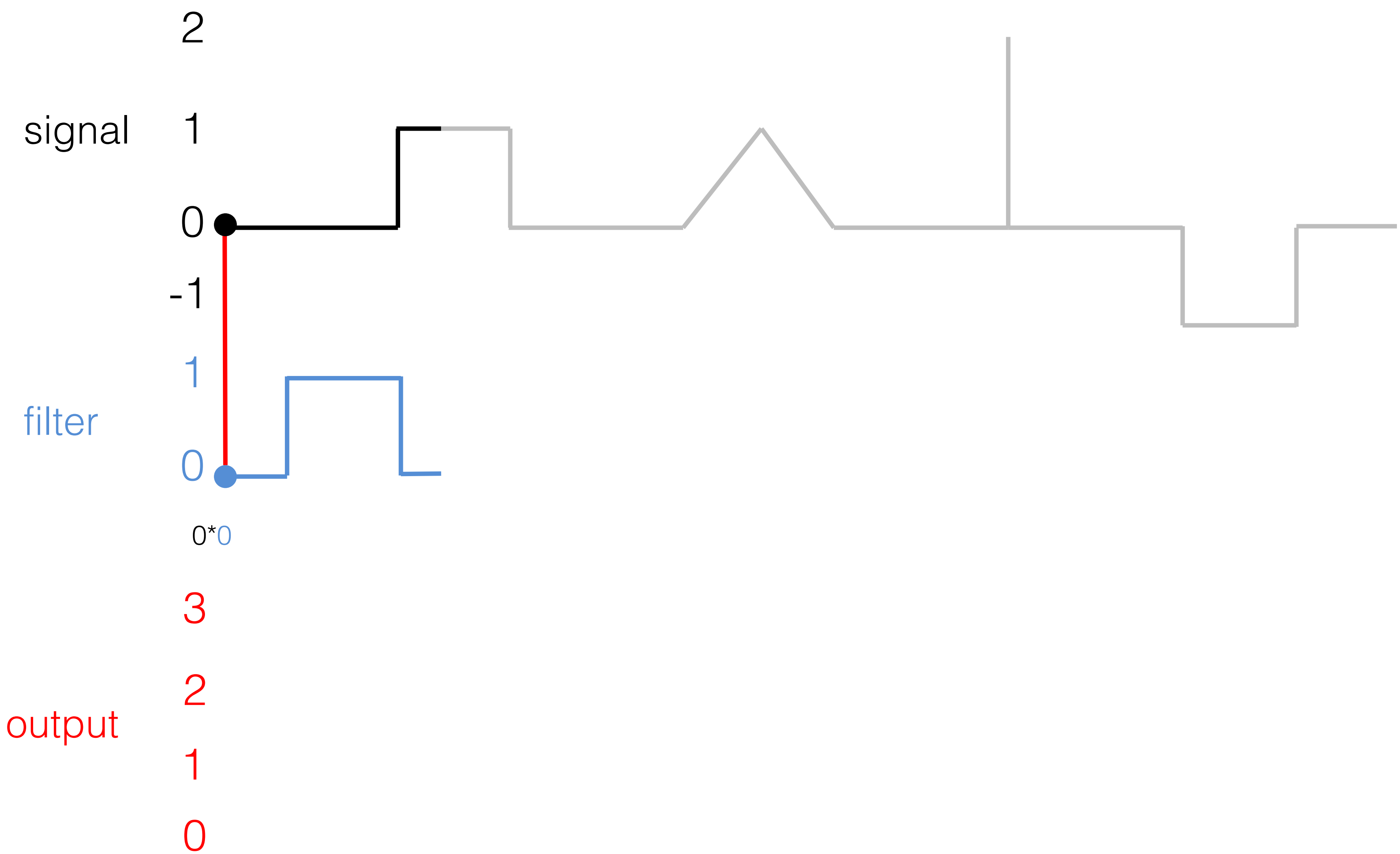


Convolution



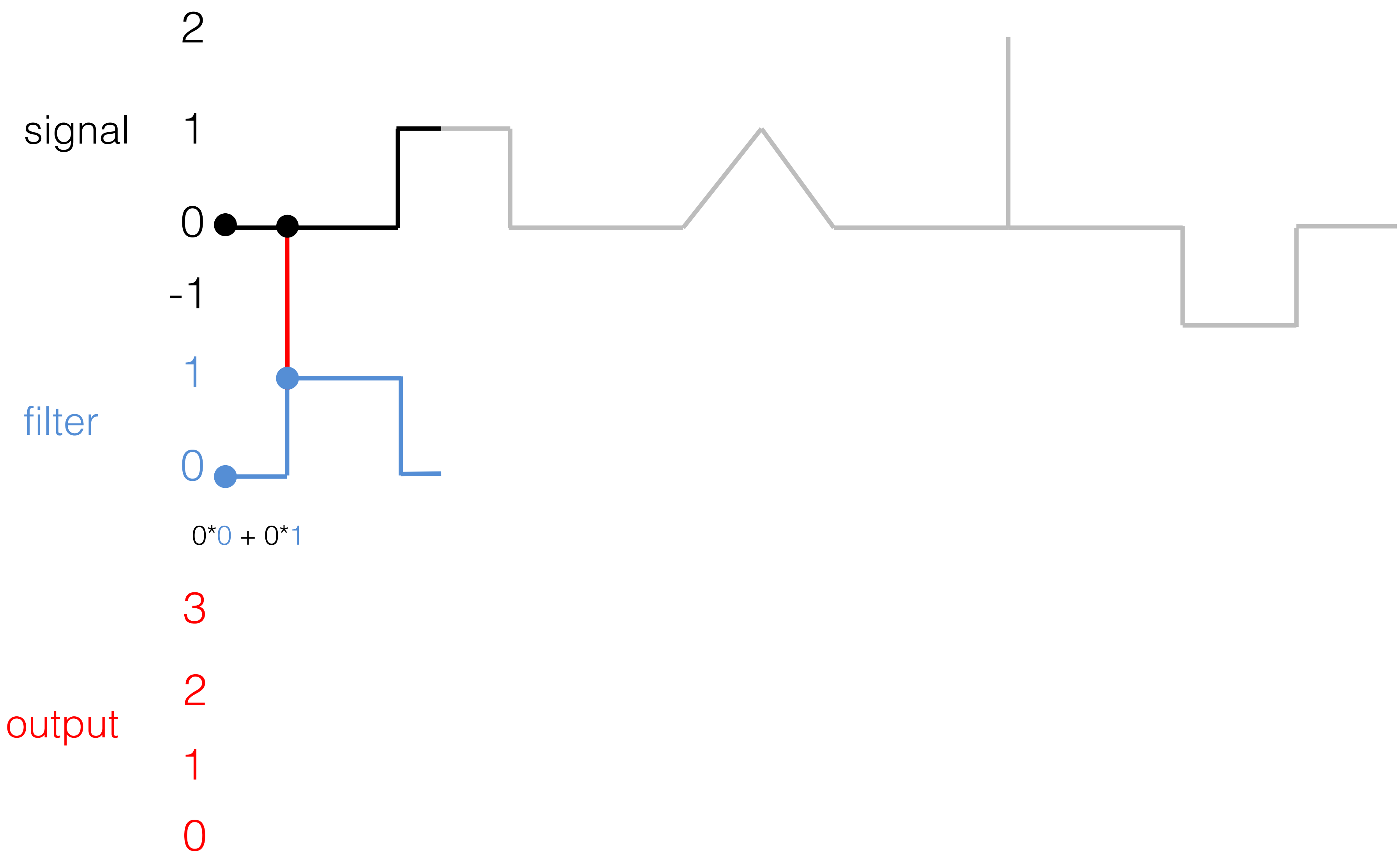
MATLAB: `conv(signal, filter, ‘valid’)`

Convolution



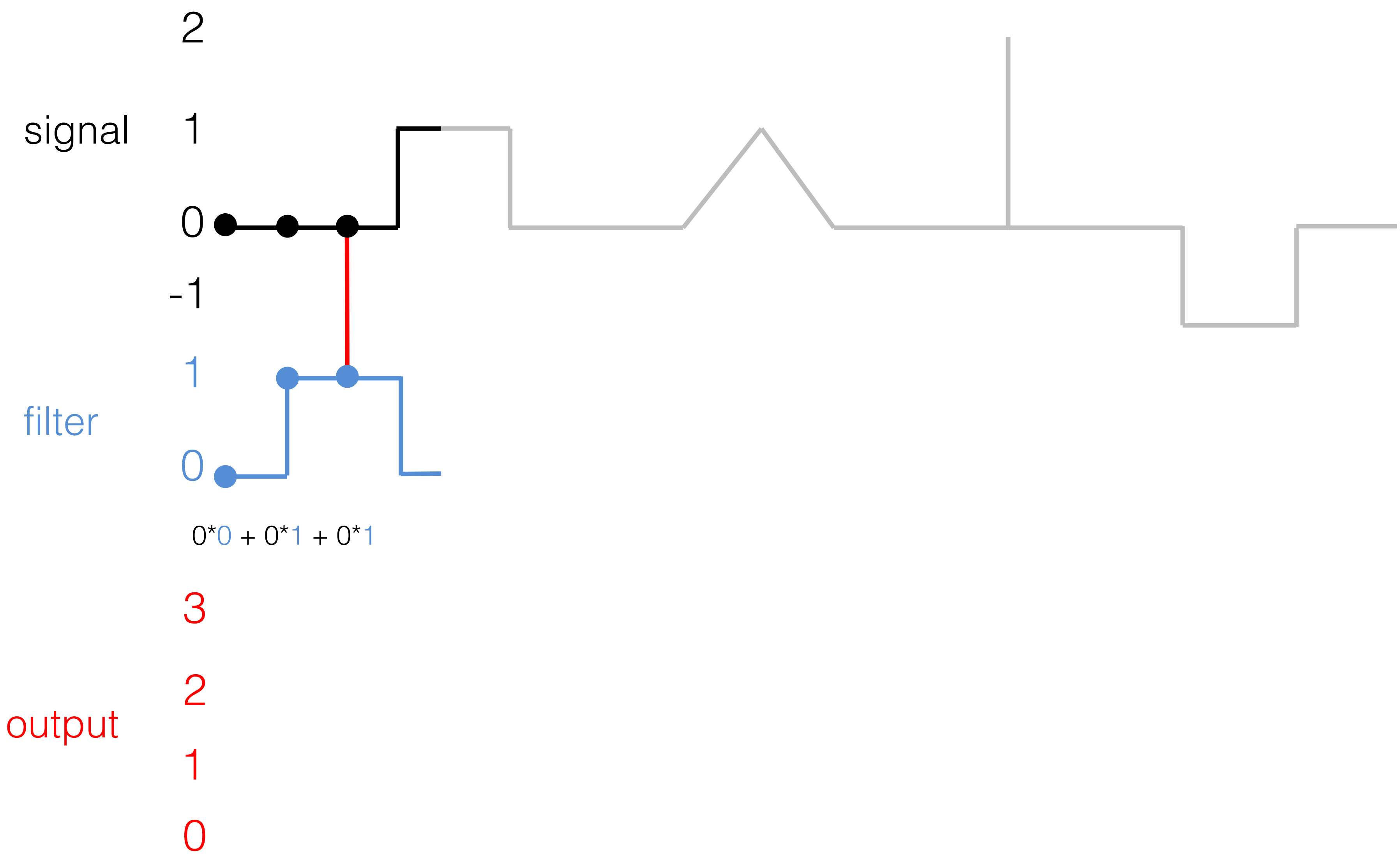
MATLAB: `conv(signal, filter, 'valid')`

Convolution



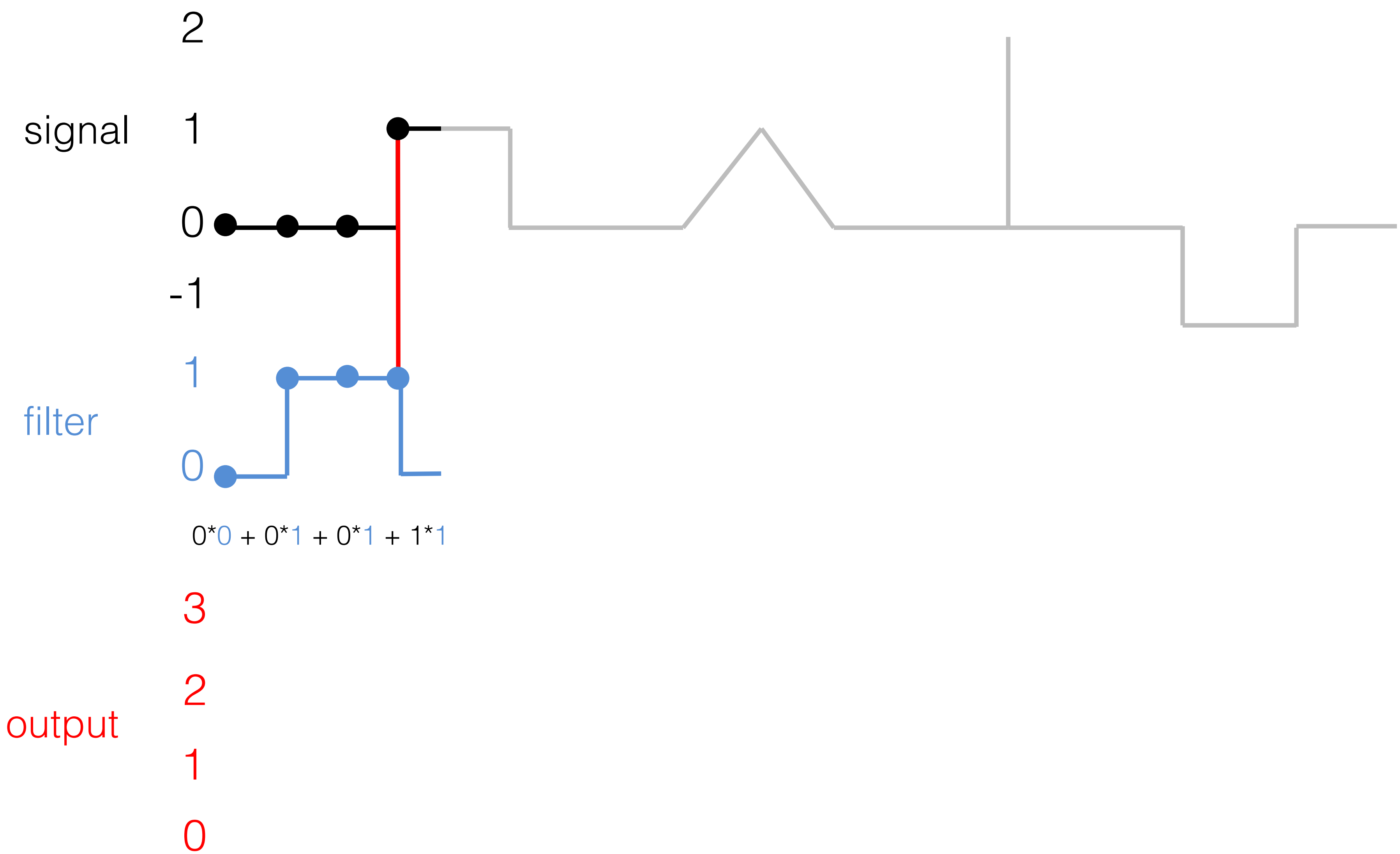
MATLAB: `conv(signal, filter, 'valid')`

Convolution



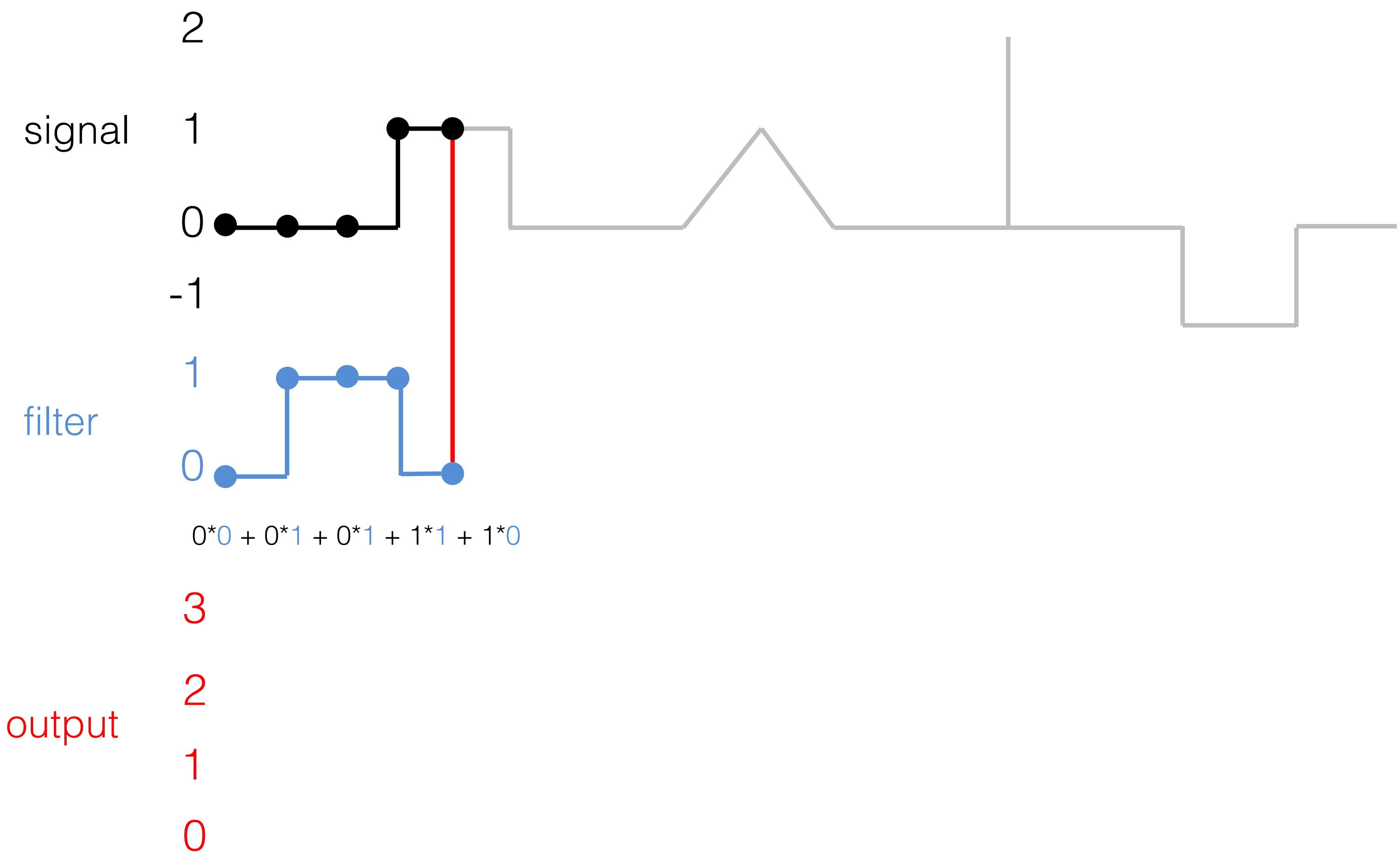
MATLAB: `conv(signal, filter, 'valid')`

Convolution



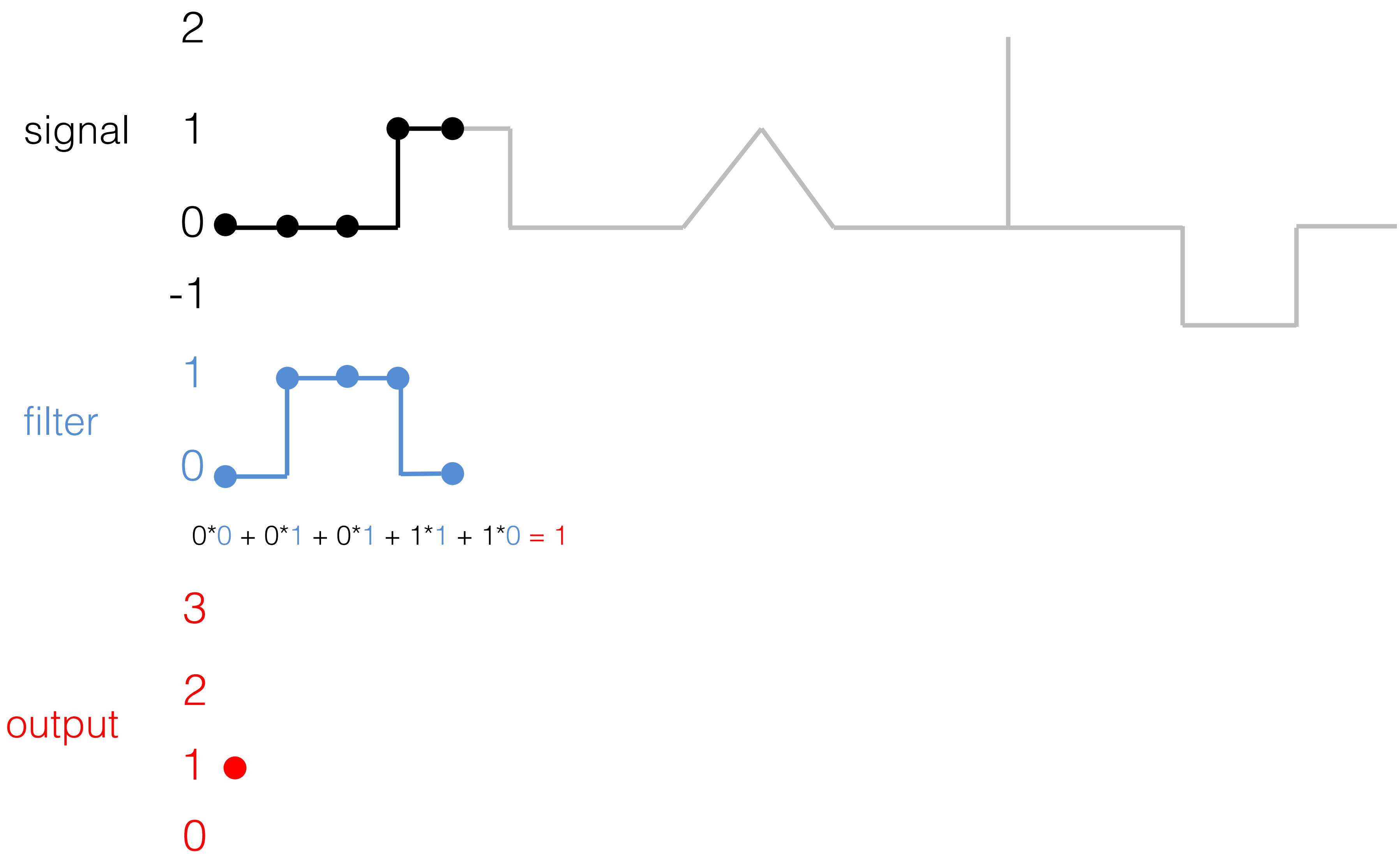
MATLAB: `conv(signal, filter, 'valid')`

Convolution



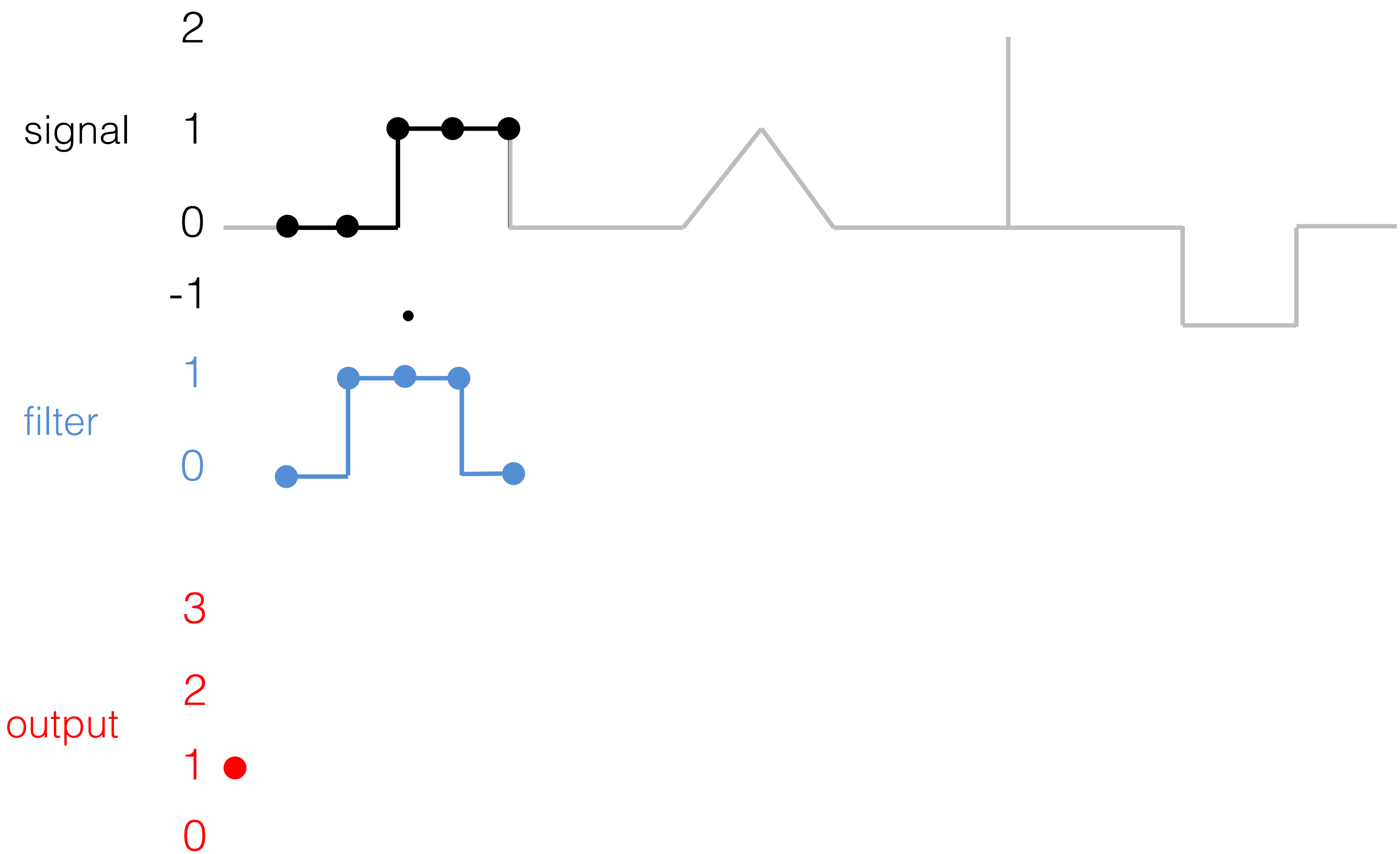
MATLAB: `conv(signal, filter, 'valid')`

Convolution



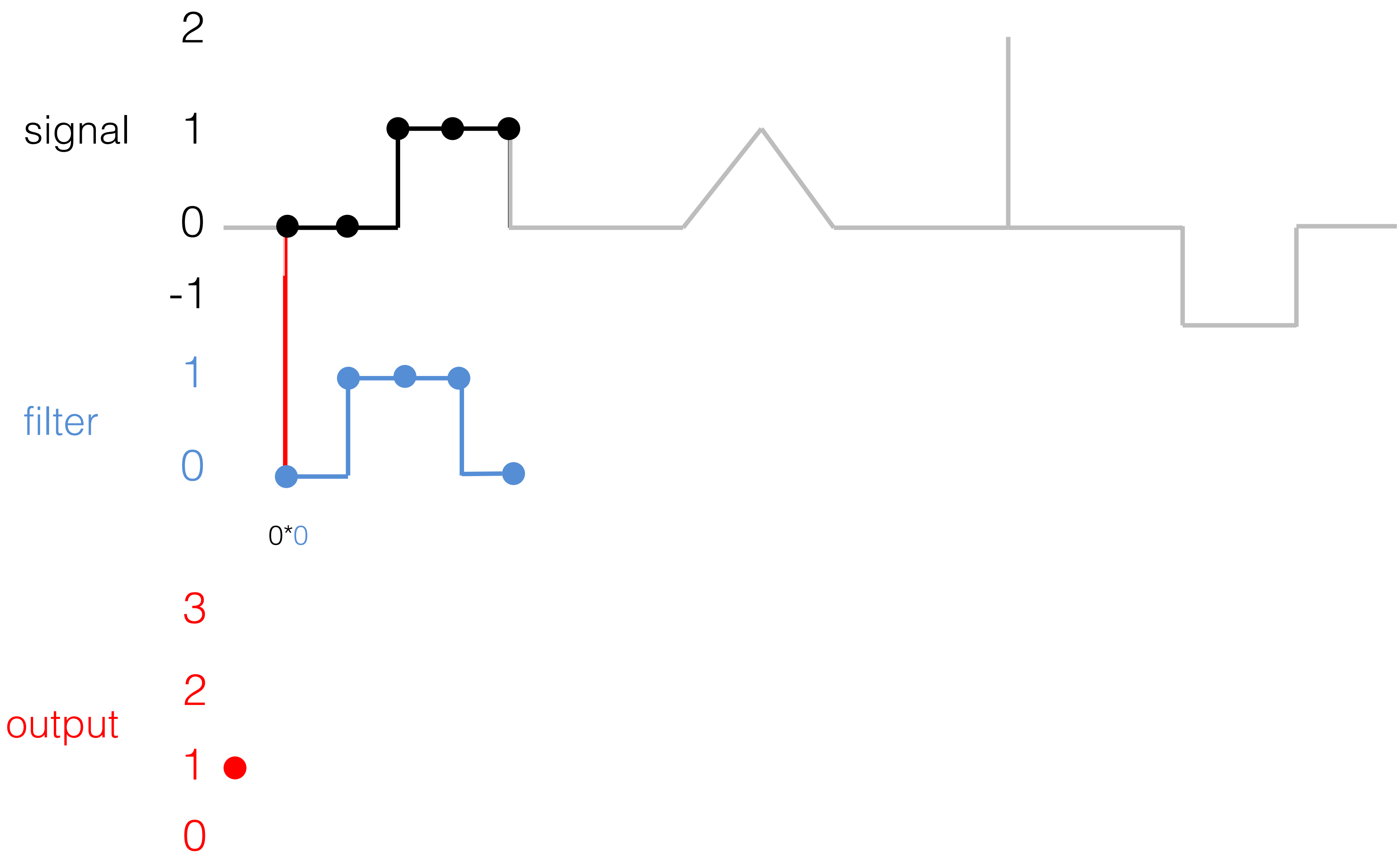
MATLAB: `conv(signal, filter, ‘valid’)`

Convolution



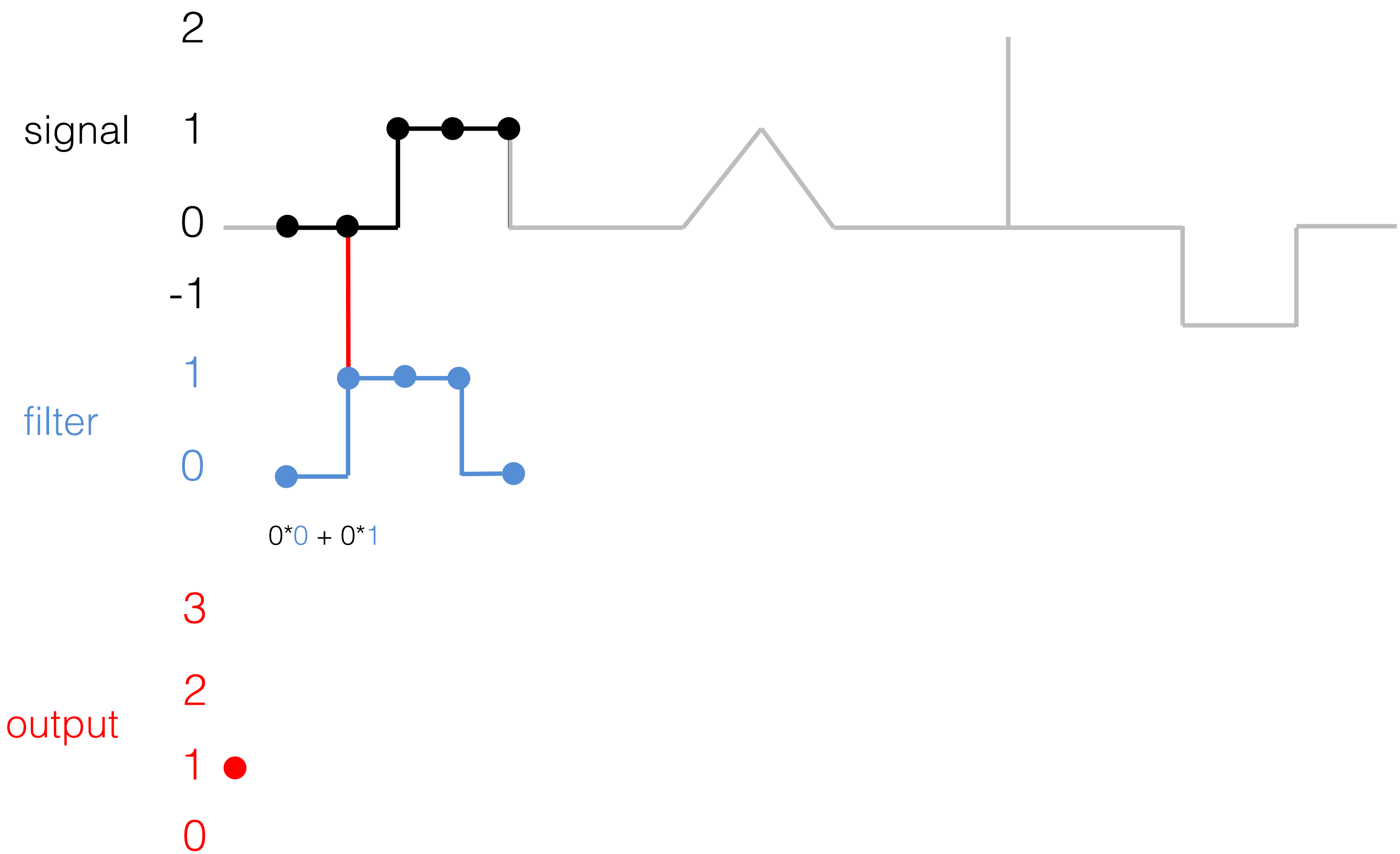
MATLAB: `conv(signal, filter, 'valid')`

Convolution



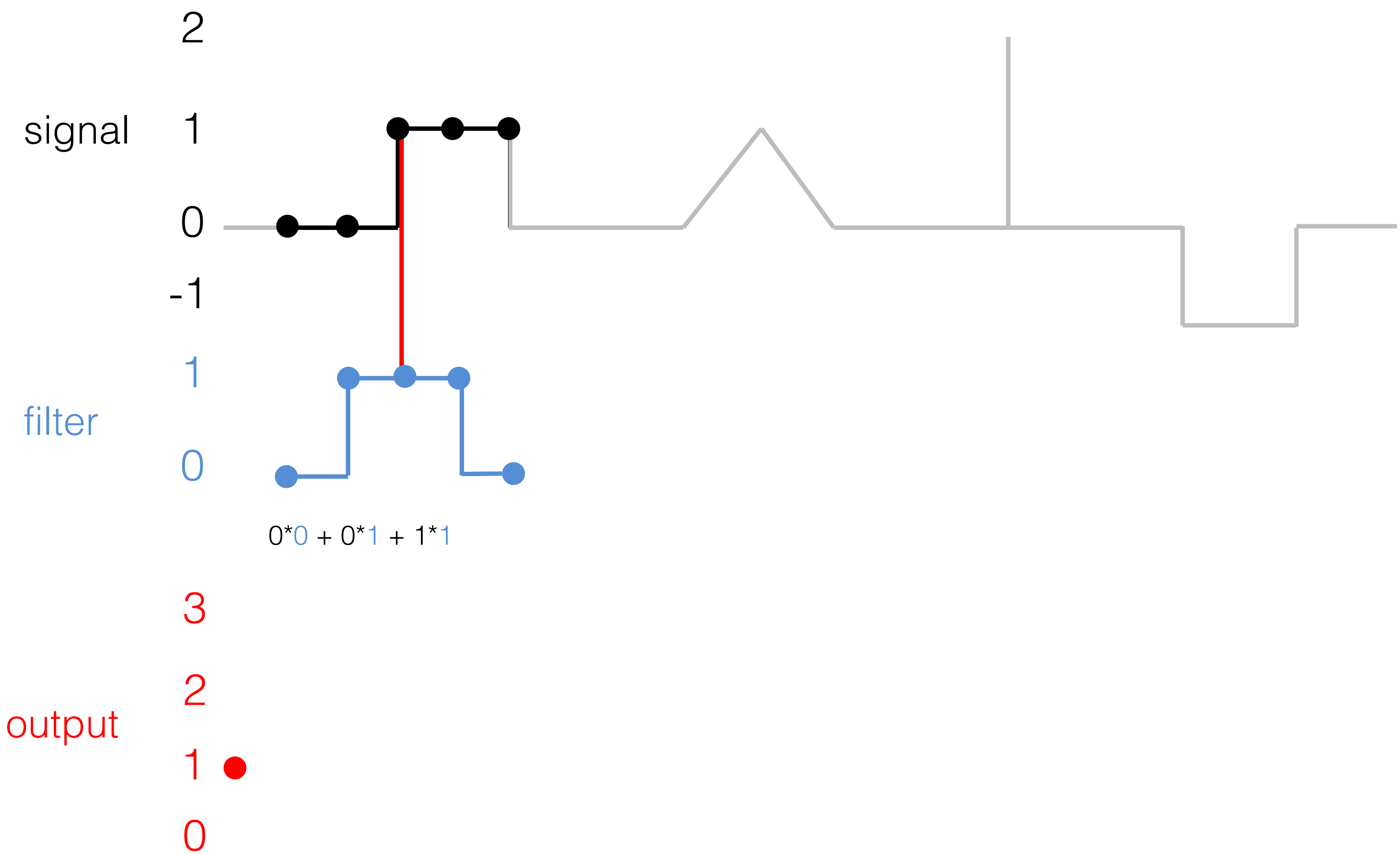
MATLAB: `conv(signal, filter, ‘valid’)`

Convolution



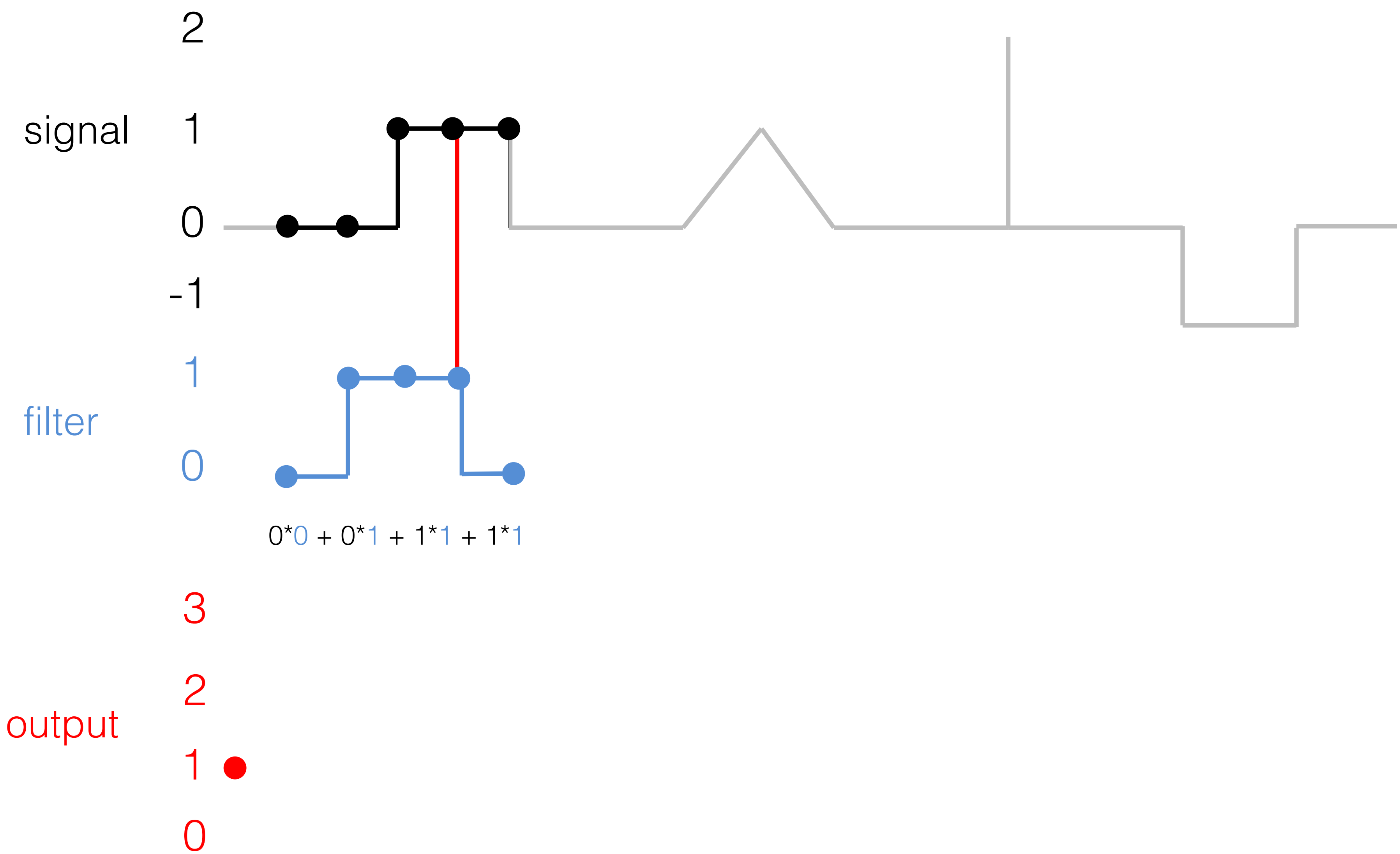
MATLAB: `conv(signal, filter, 'valid')`

Convolution



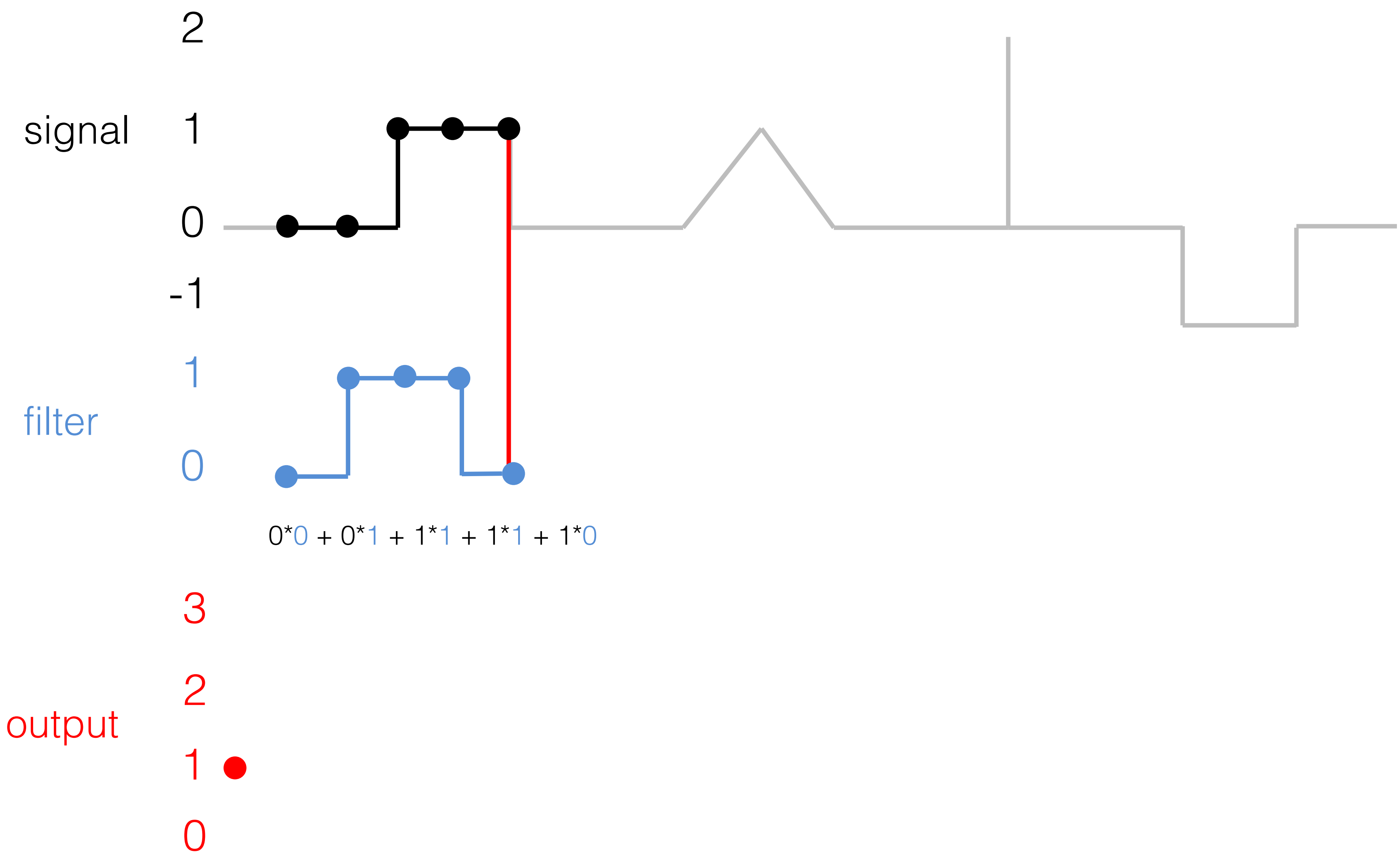
MATLAB: `conv(signal, filter, 'valid')`

Convolution



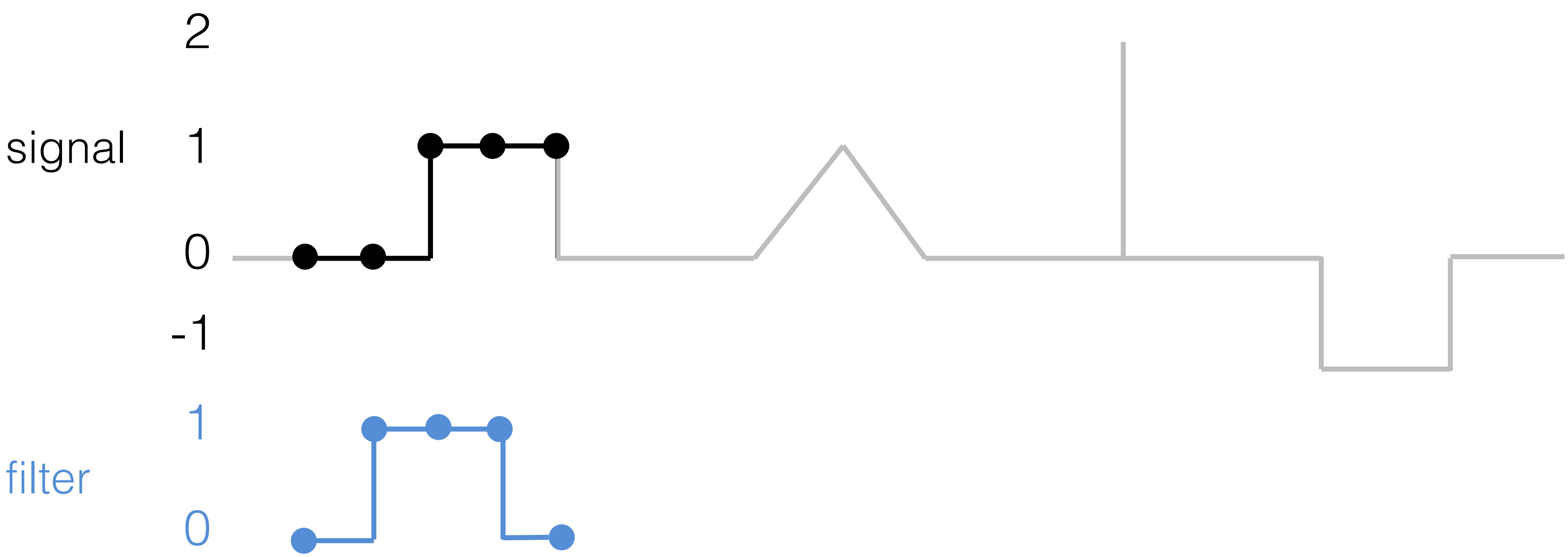
MATLAB: `conv(signal, filter, ‘valid’)`

Convolution

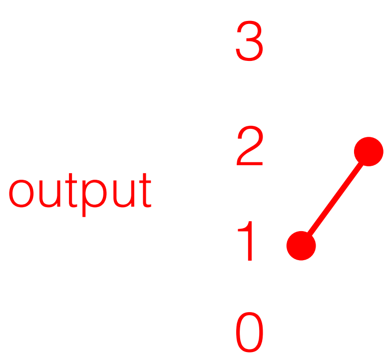


MATLAB: `conv(signal, filter, 'valid')`

Convolution

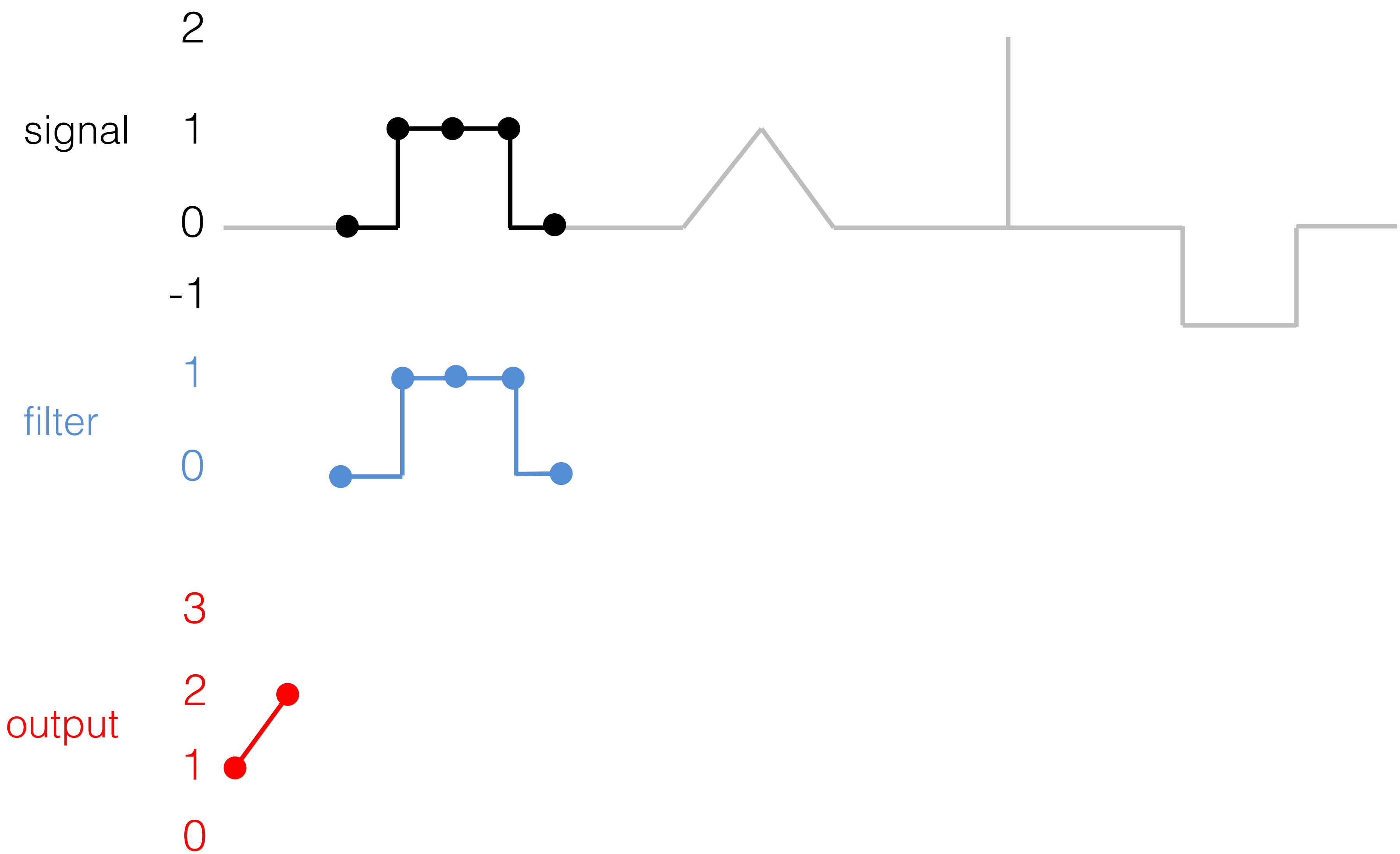


0*0 + 0*1 + 1*1 + 1*1 + 1*0 = 2



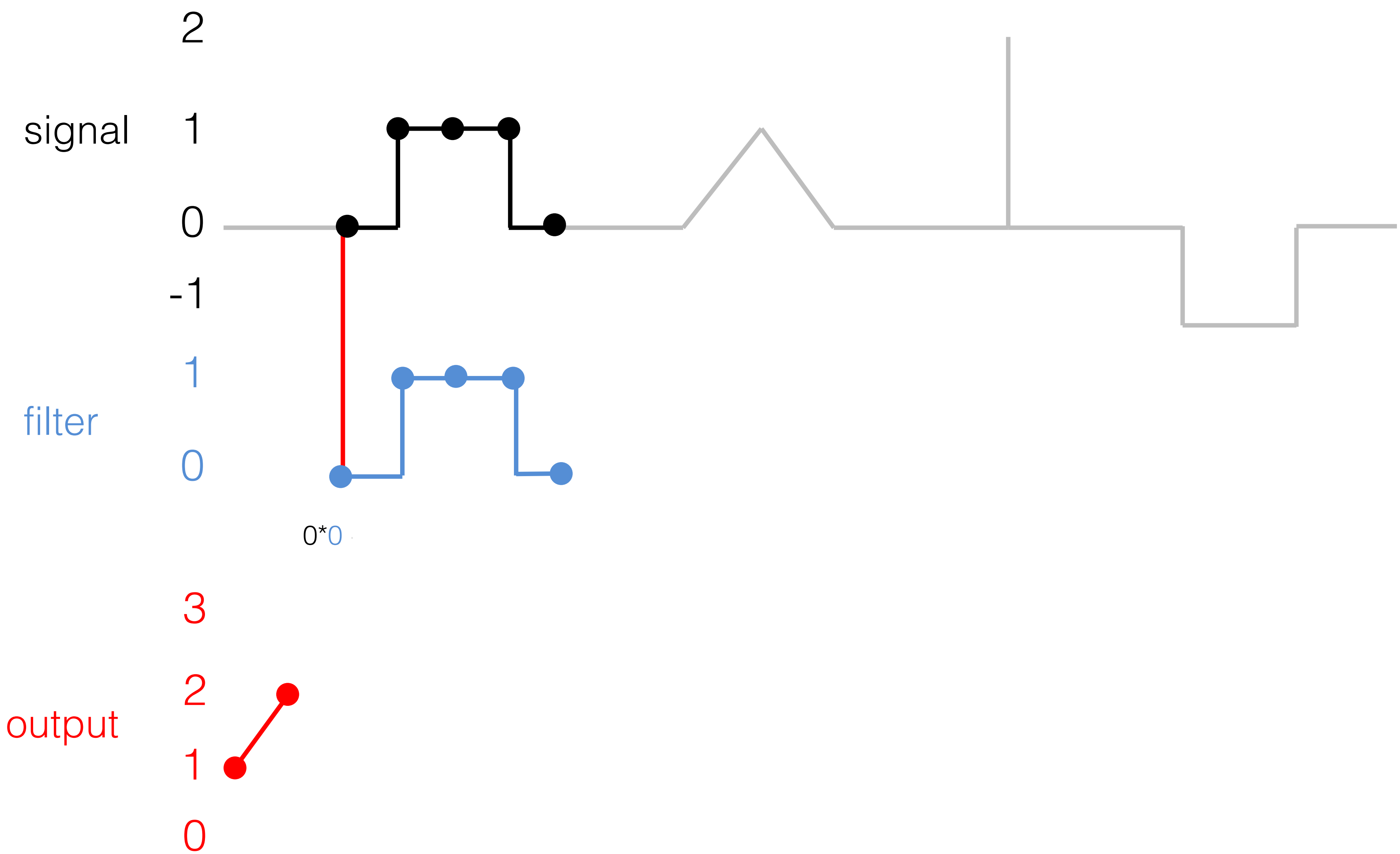
MATLAB: `conv(signal, filter, ‘valid’)`

Convolution



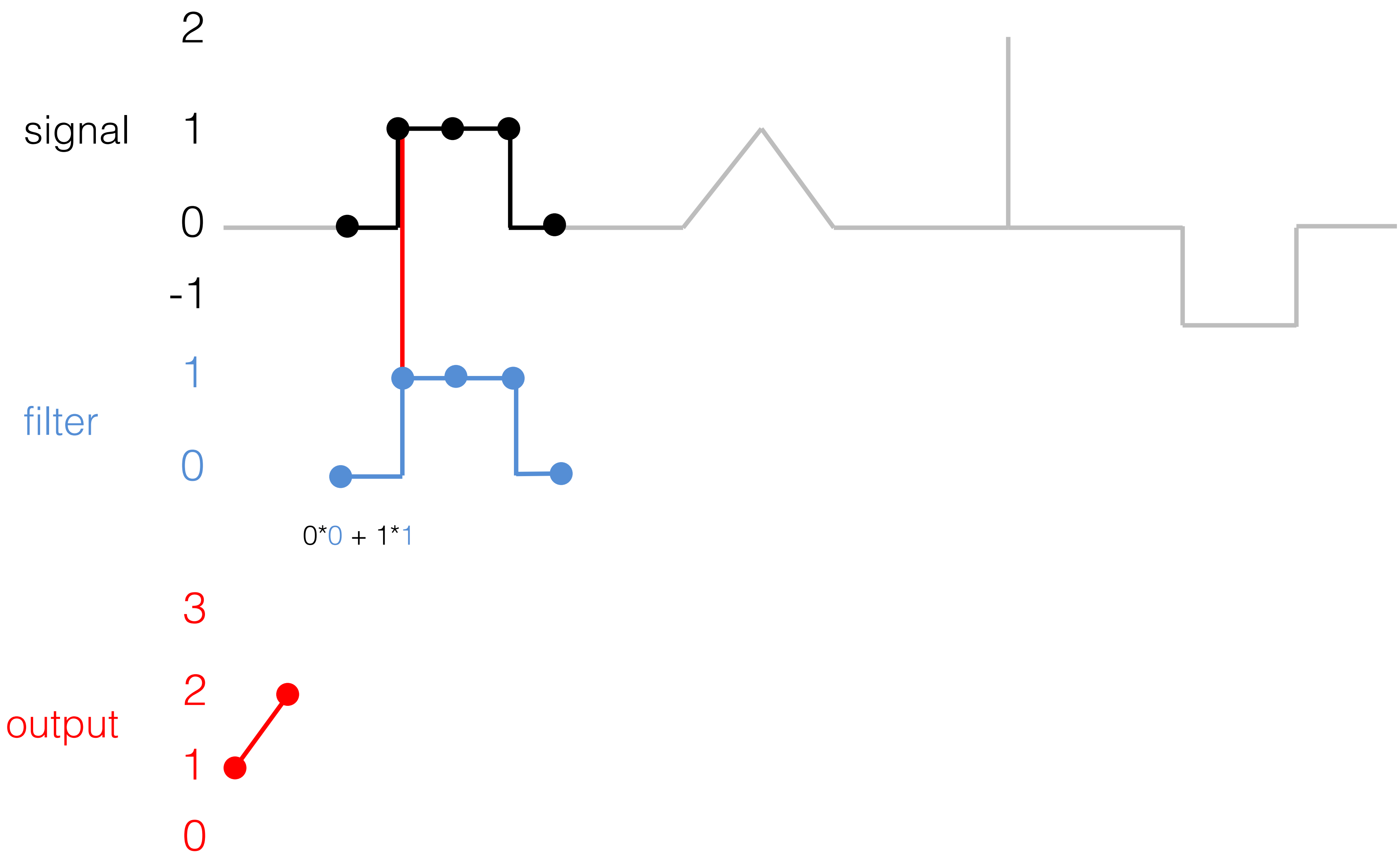
MATLAB: `conv(signal, filter, 'valid')`

Convolution



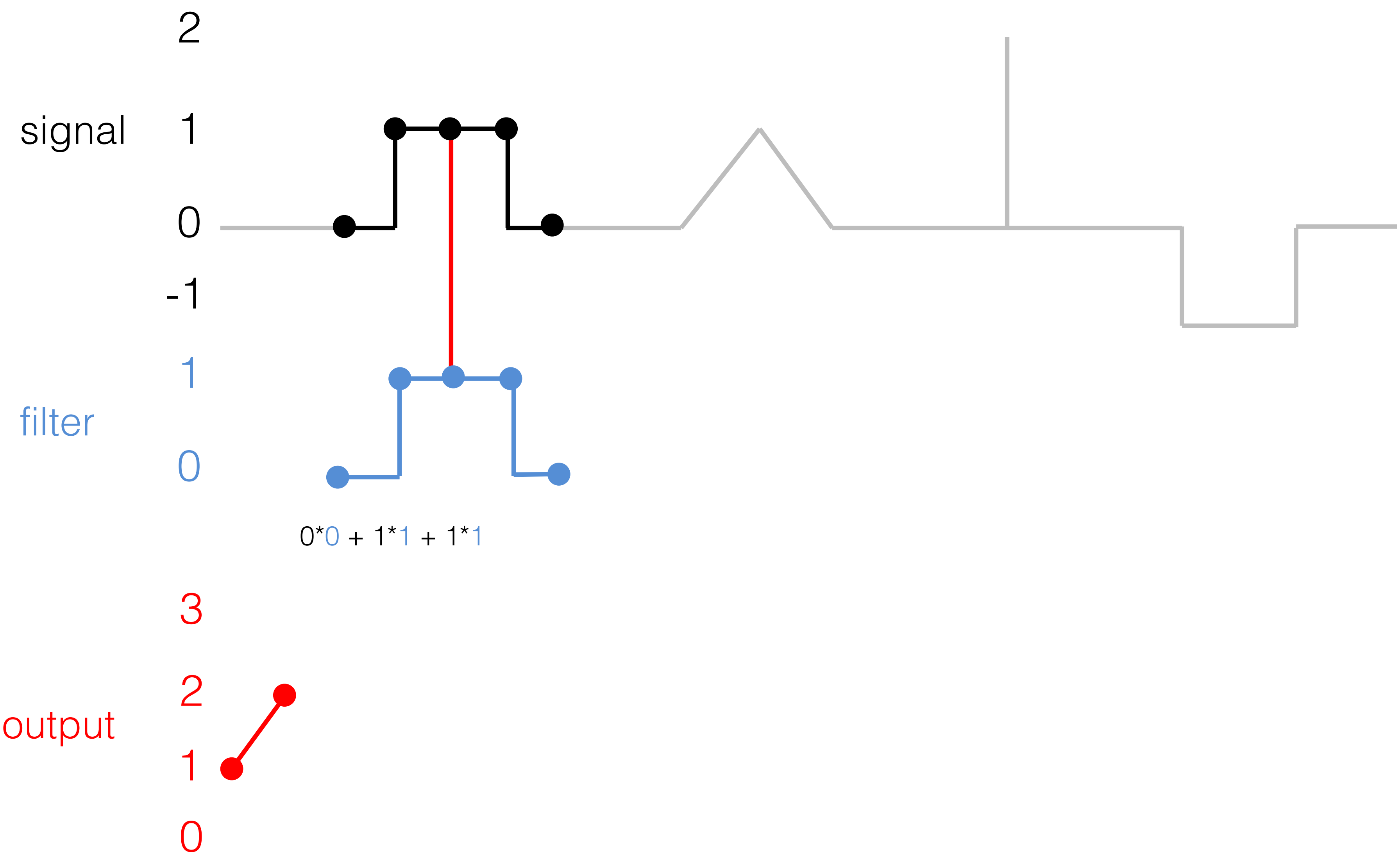
MATLAB: `conv(signal, filter, 'valid')`

Convolution



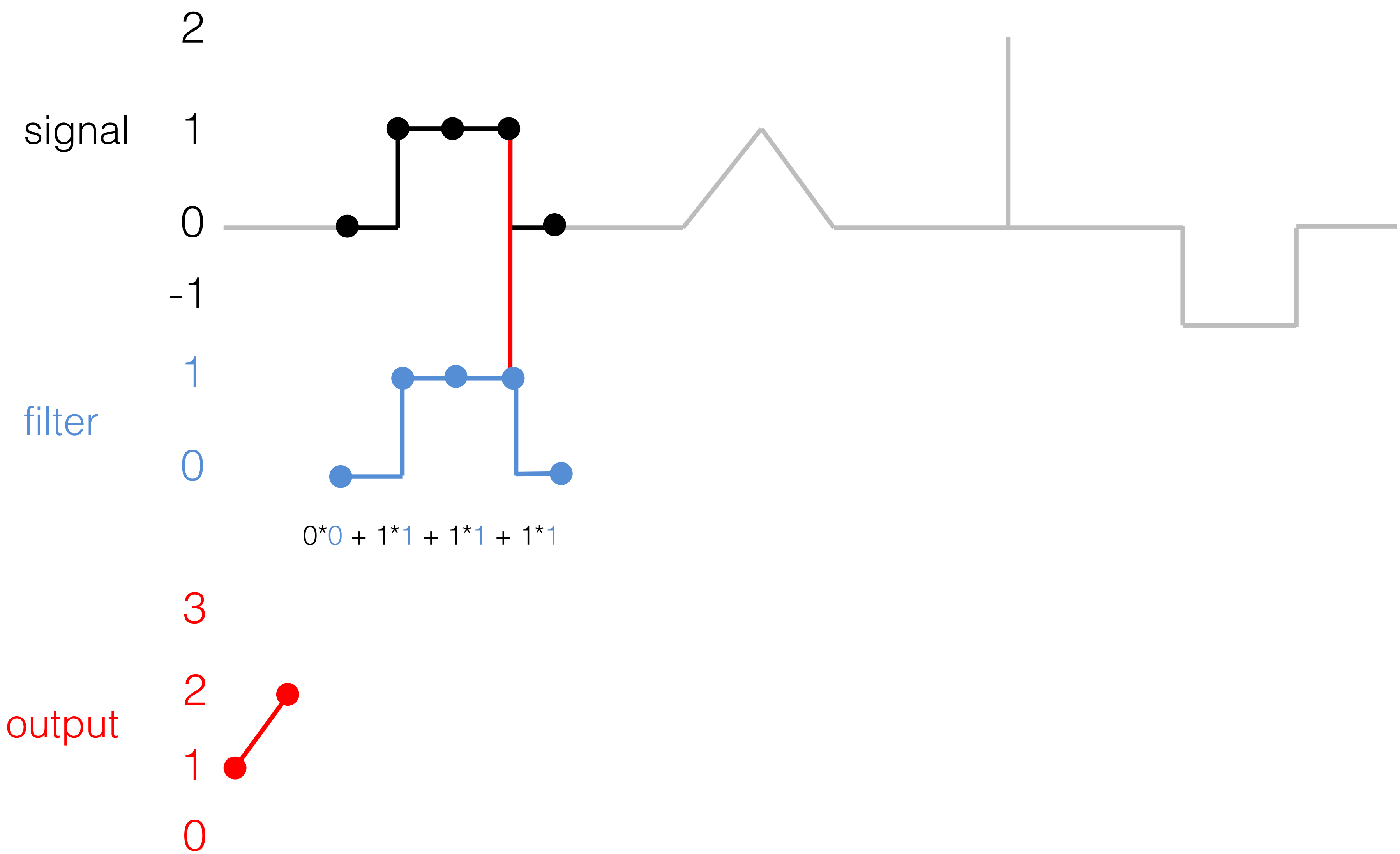
MATLAB: `conv(signal, filter, 'valid')`

Convolution



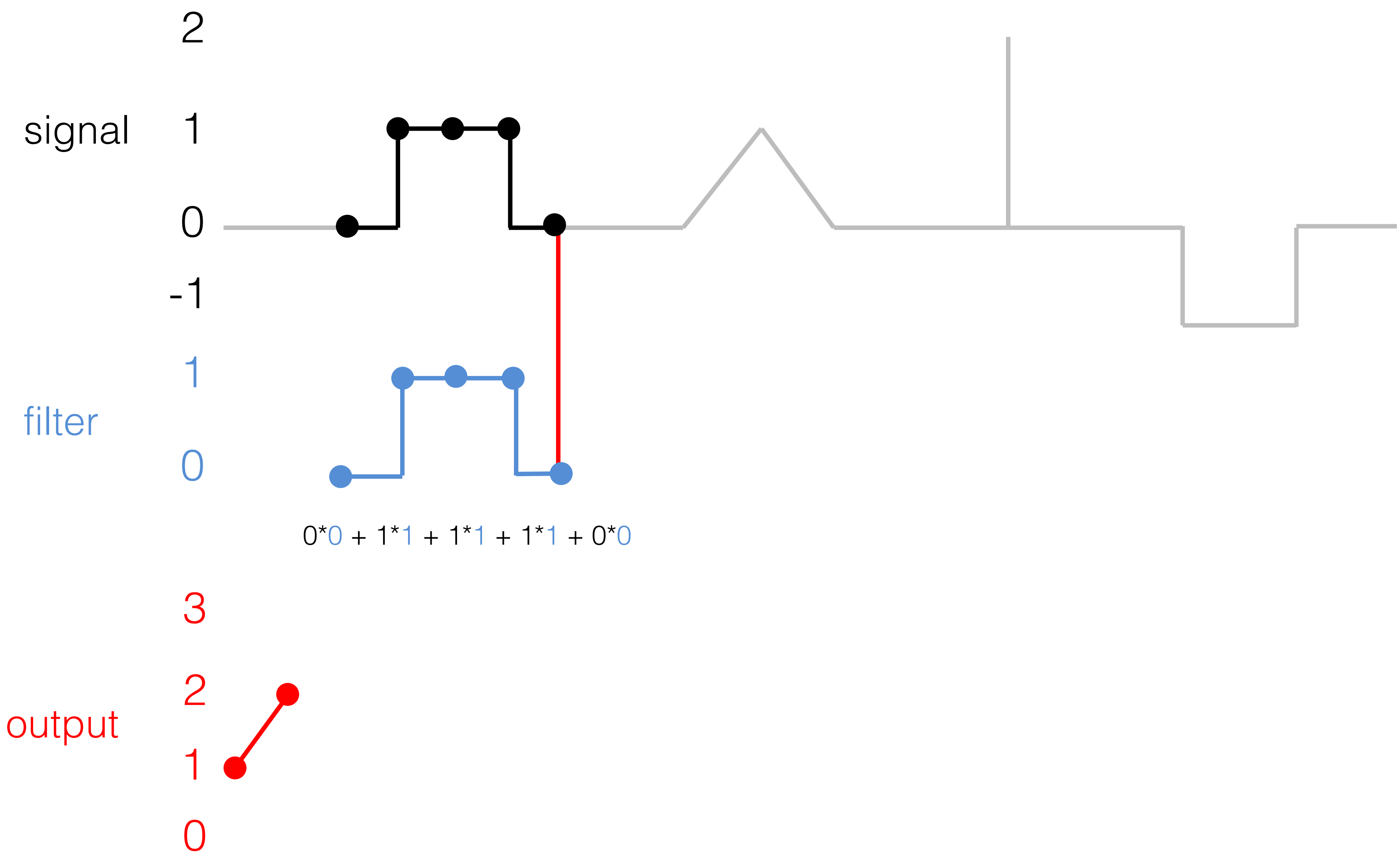
MATLAB: `conv(signal, filter, 'valid')`

Convolution



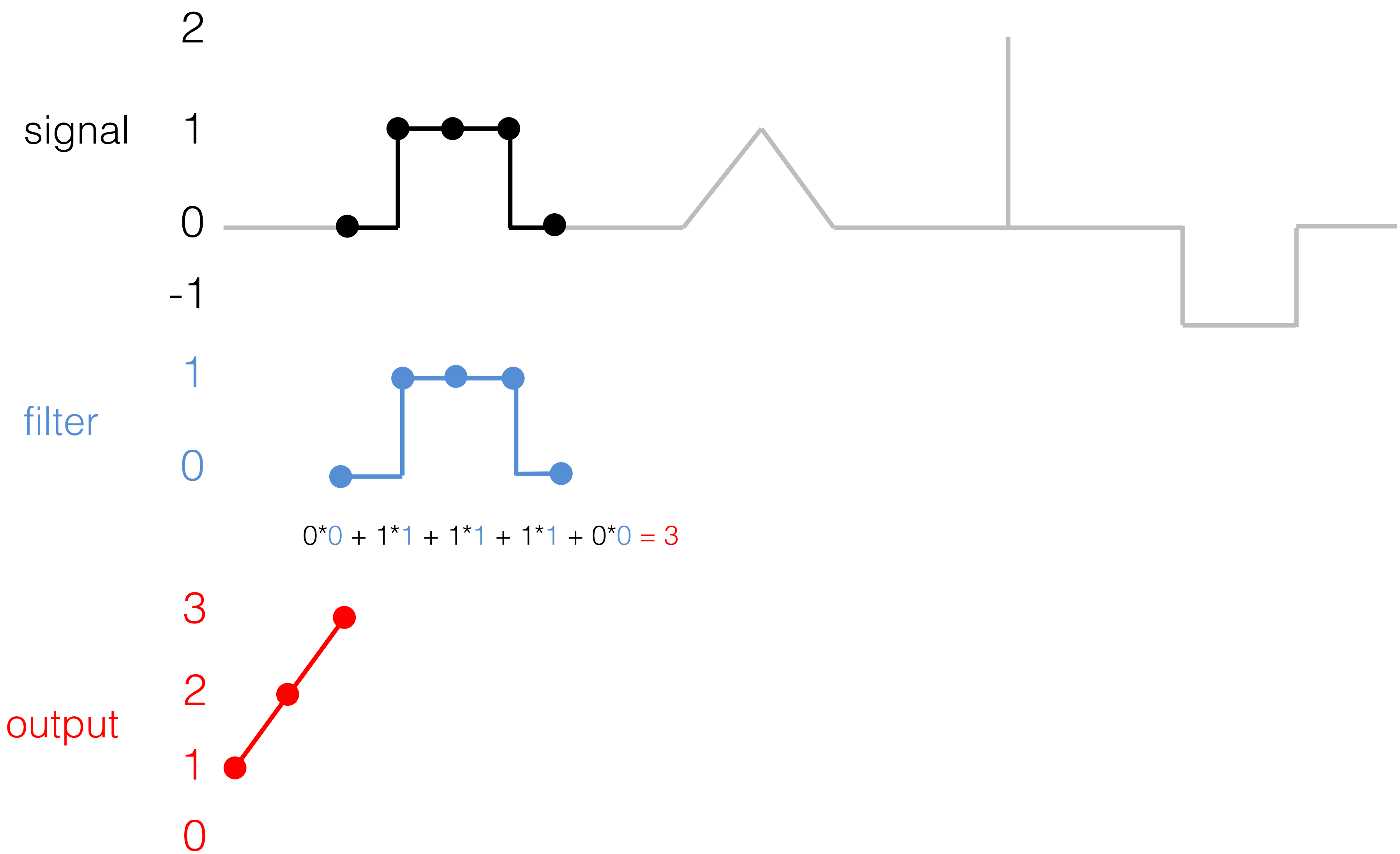
MATLAB: `conv(signal, filter, ‘valid’)`

Convolution



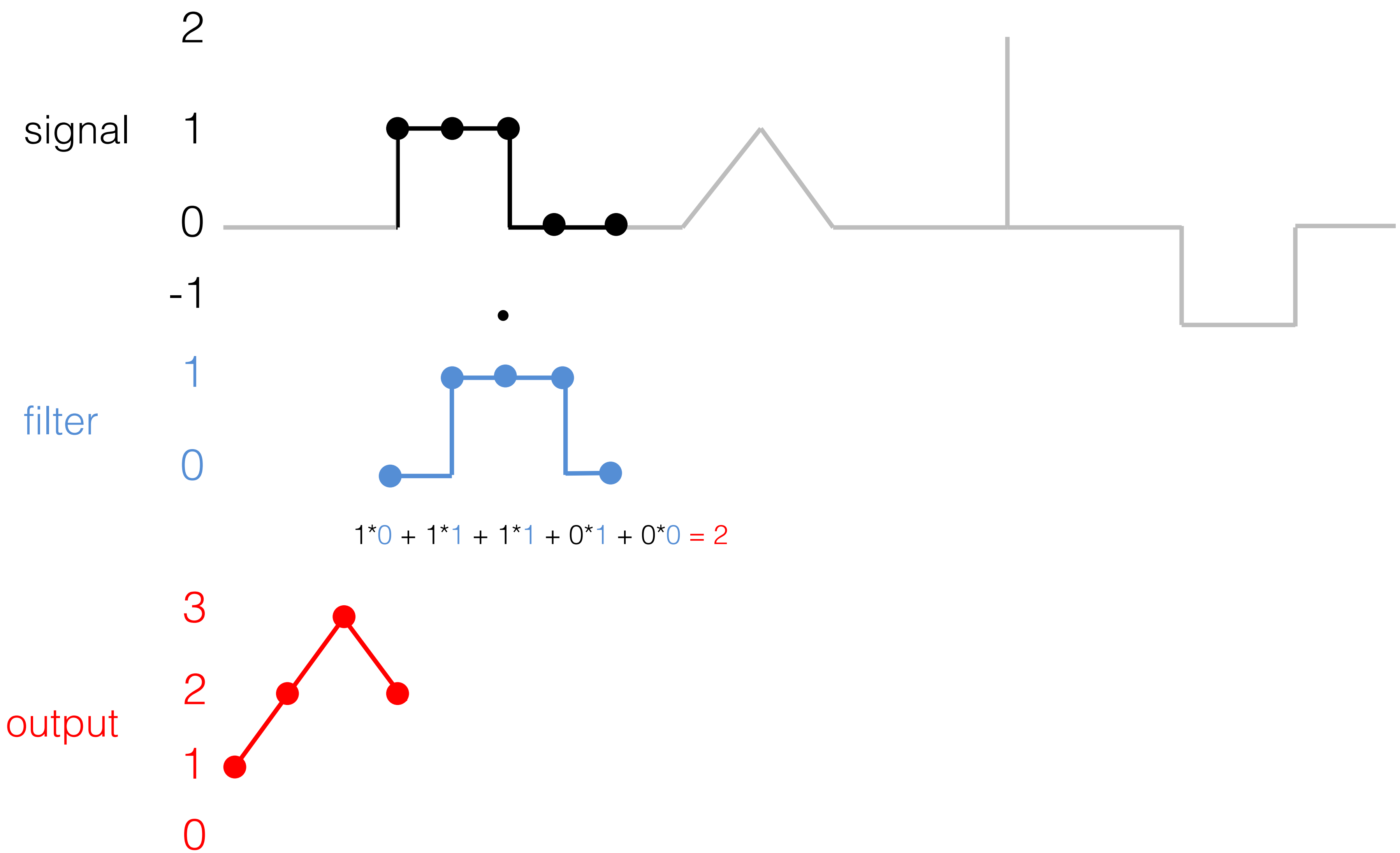
MATLAB: `conv(signal, filter, ‘valid’)`

Convolution



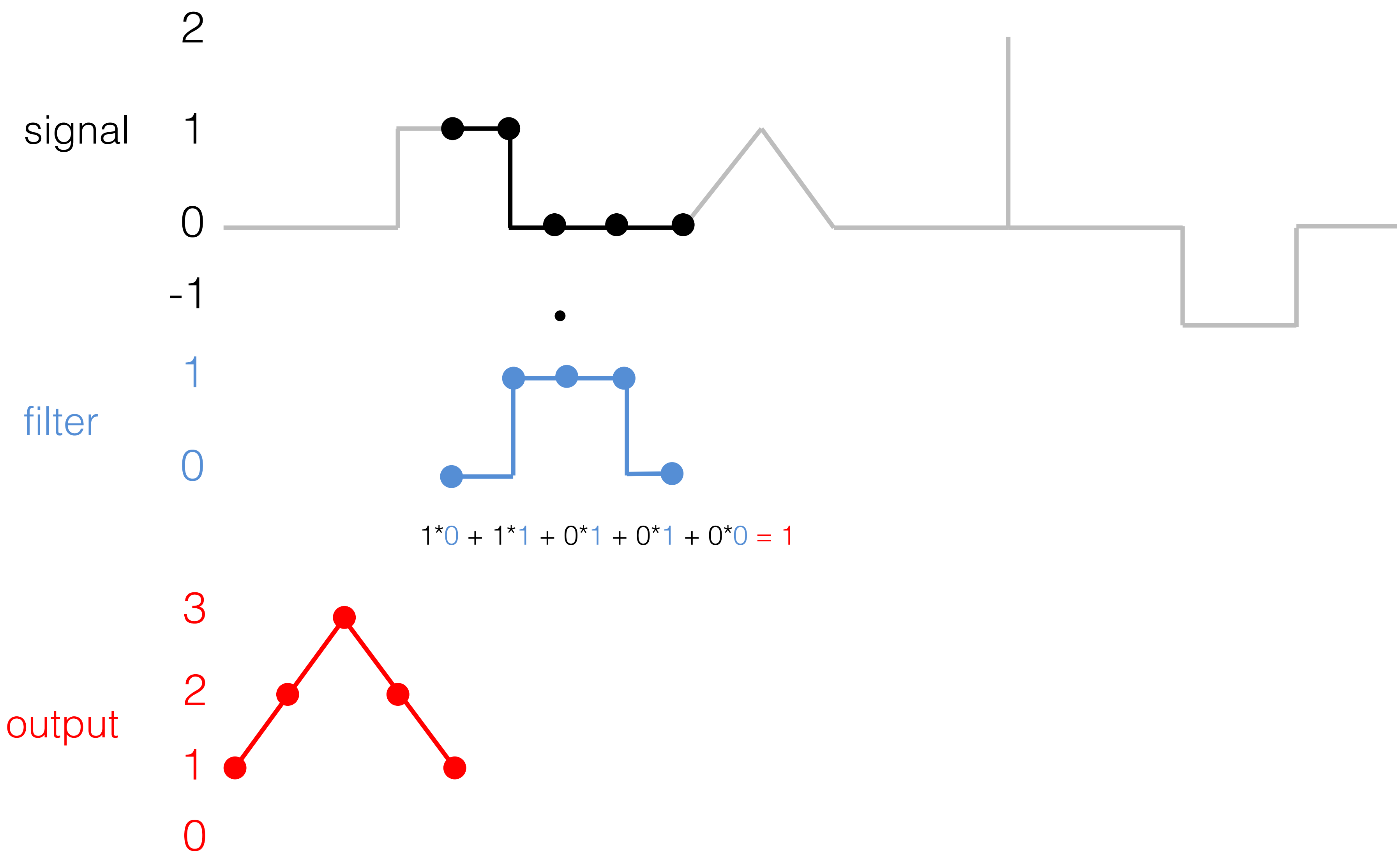
MATLAB: `conv(signal, filter, ‘valid’)`

Convolution



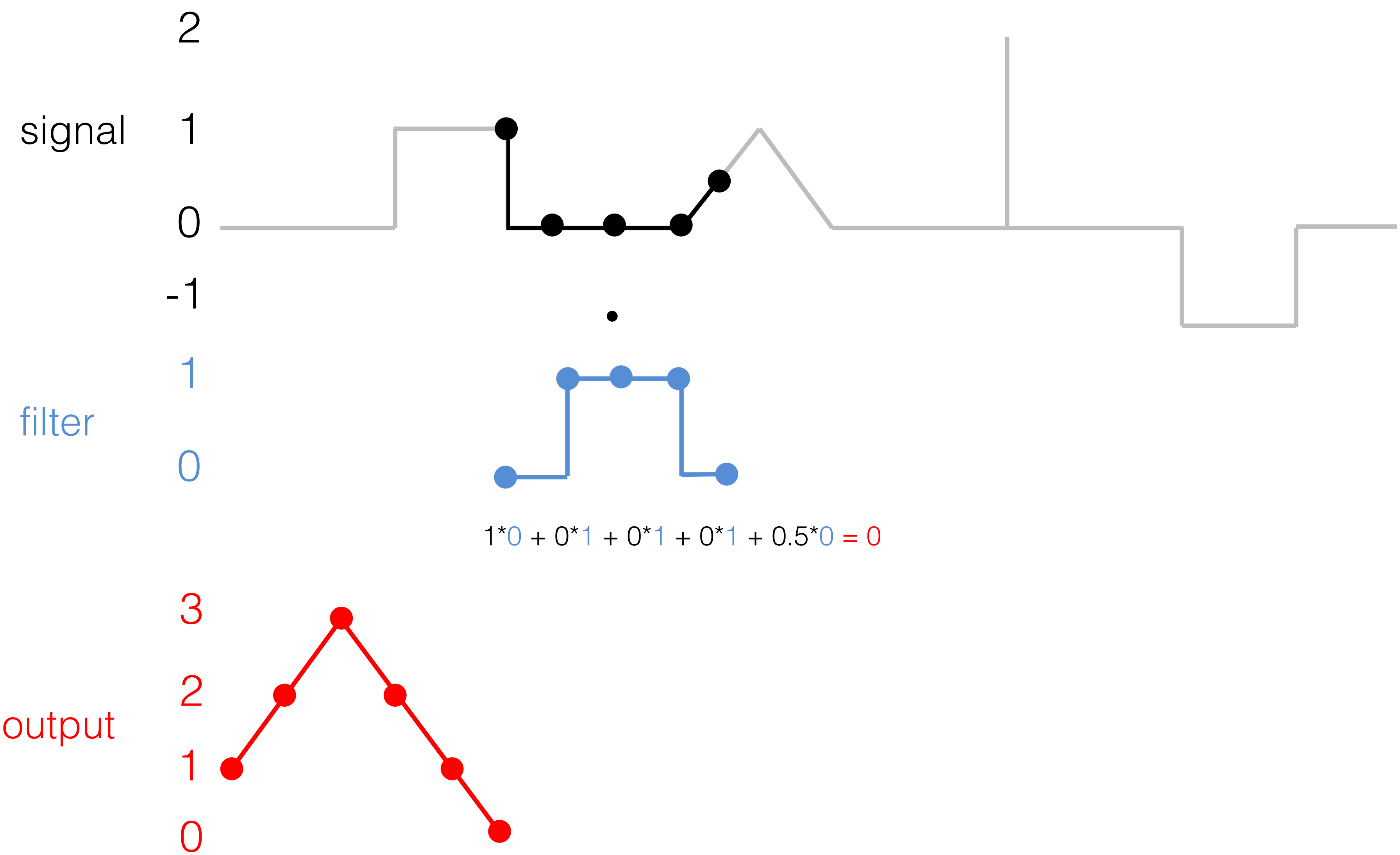
MATLAB: `conv(signal, filter, 'valid')`

Convolution



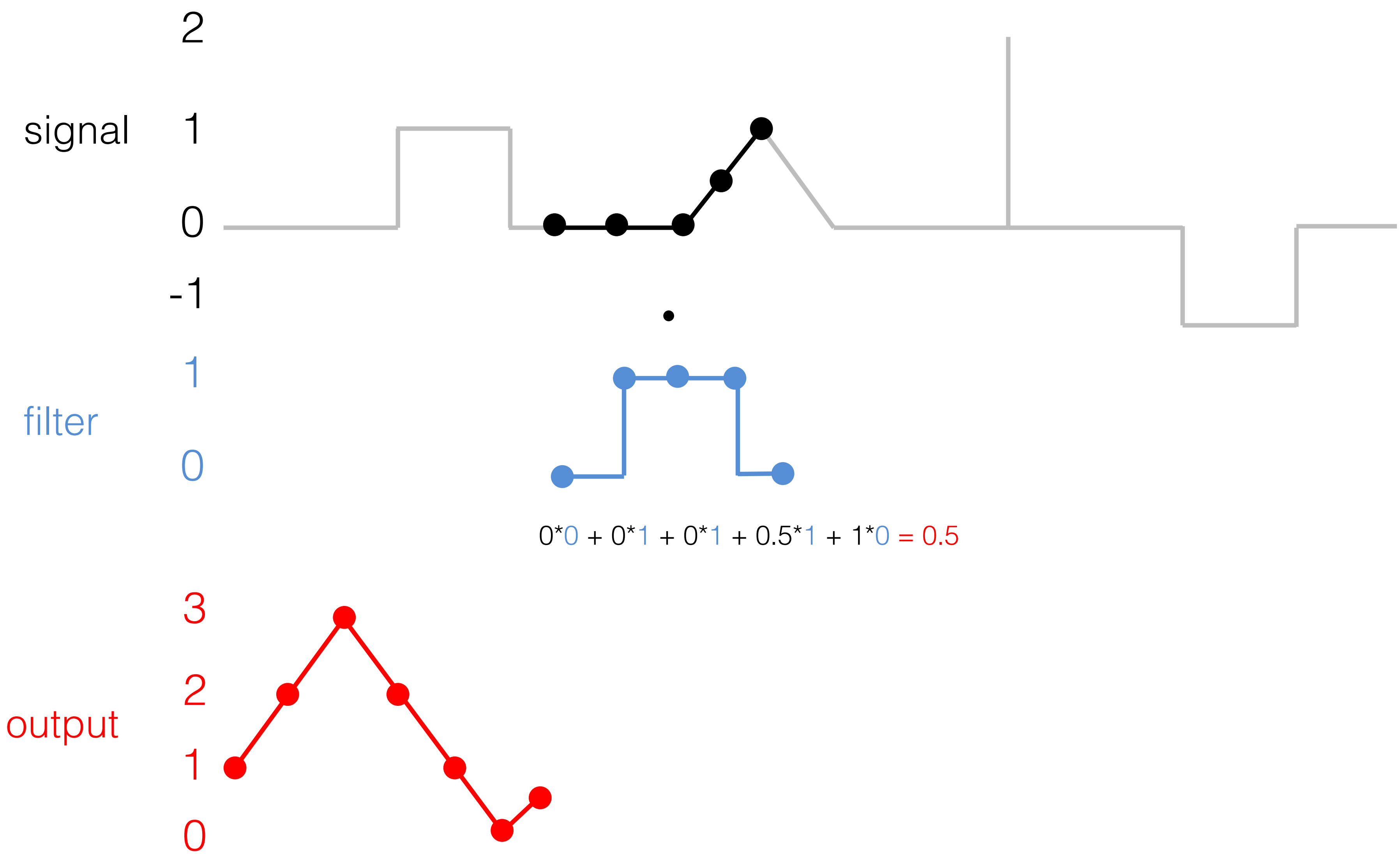
MATLAB: `conv(signal, filter, ‘valid’)`

Convolution



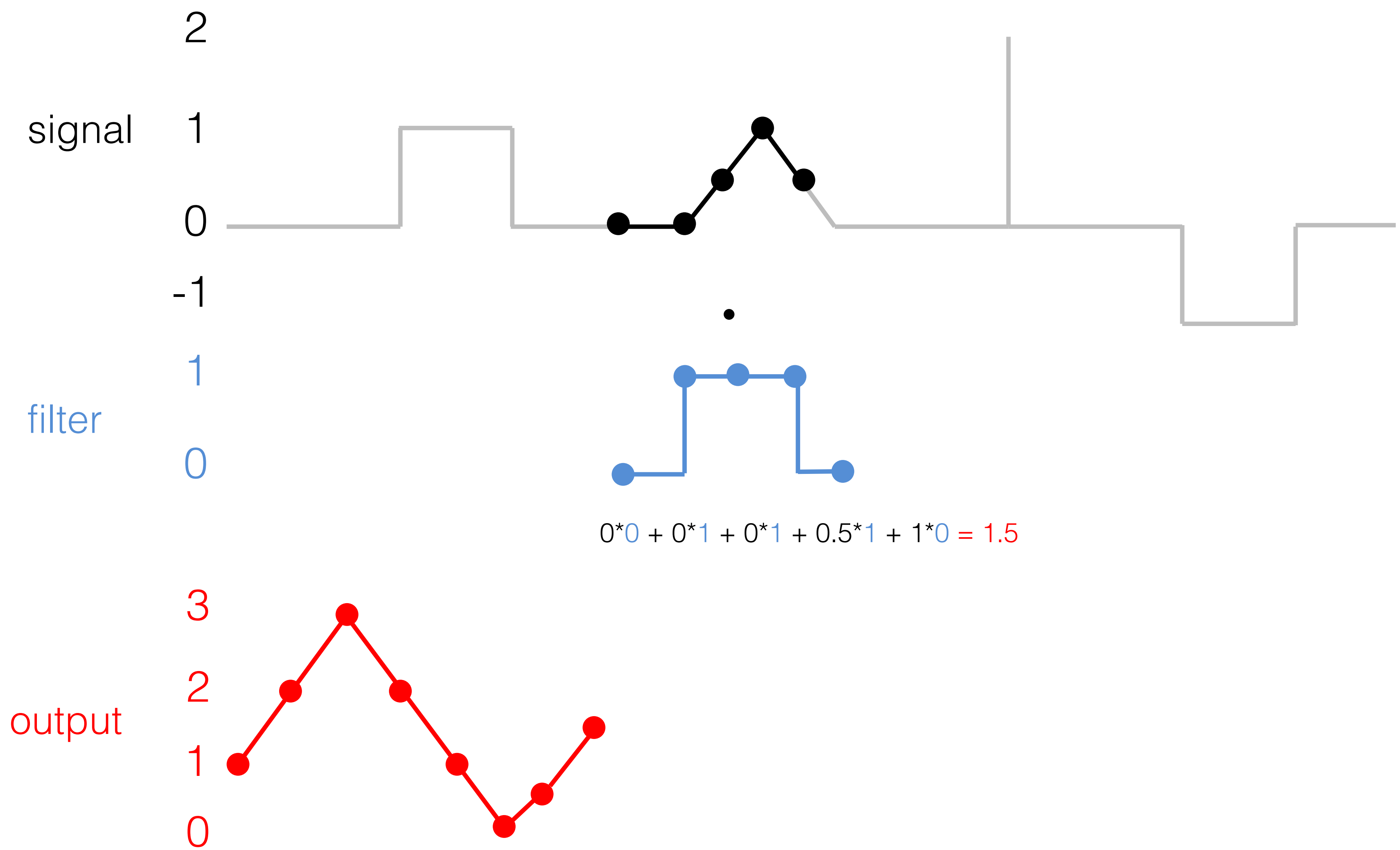
MATLAB: `conv(signal, filter, ‘valid’)`

Convolution



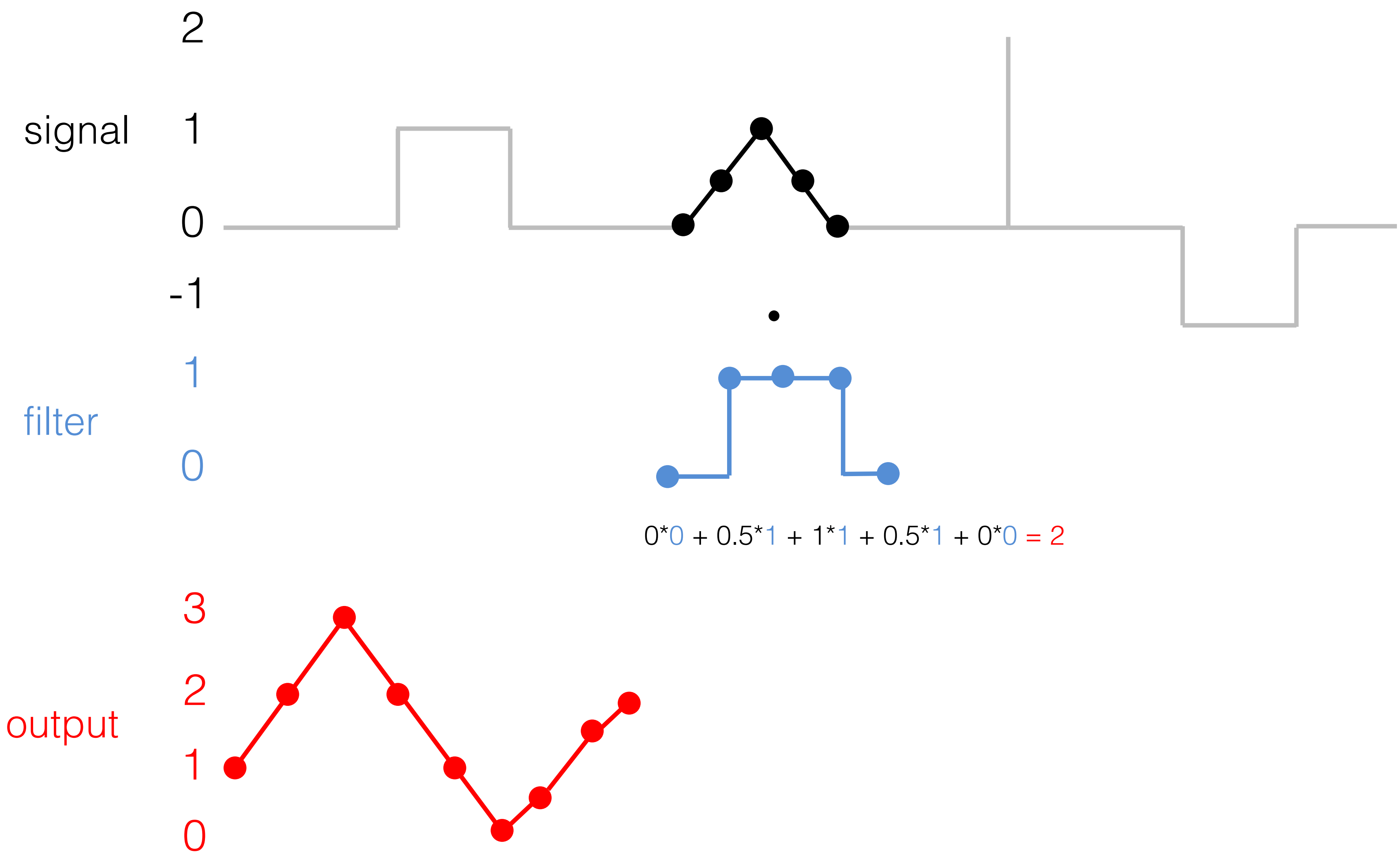
MATLAB: `conv(signal, filter, ‘valid’)`

Convolution



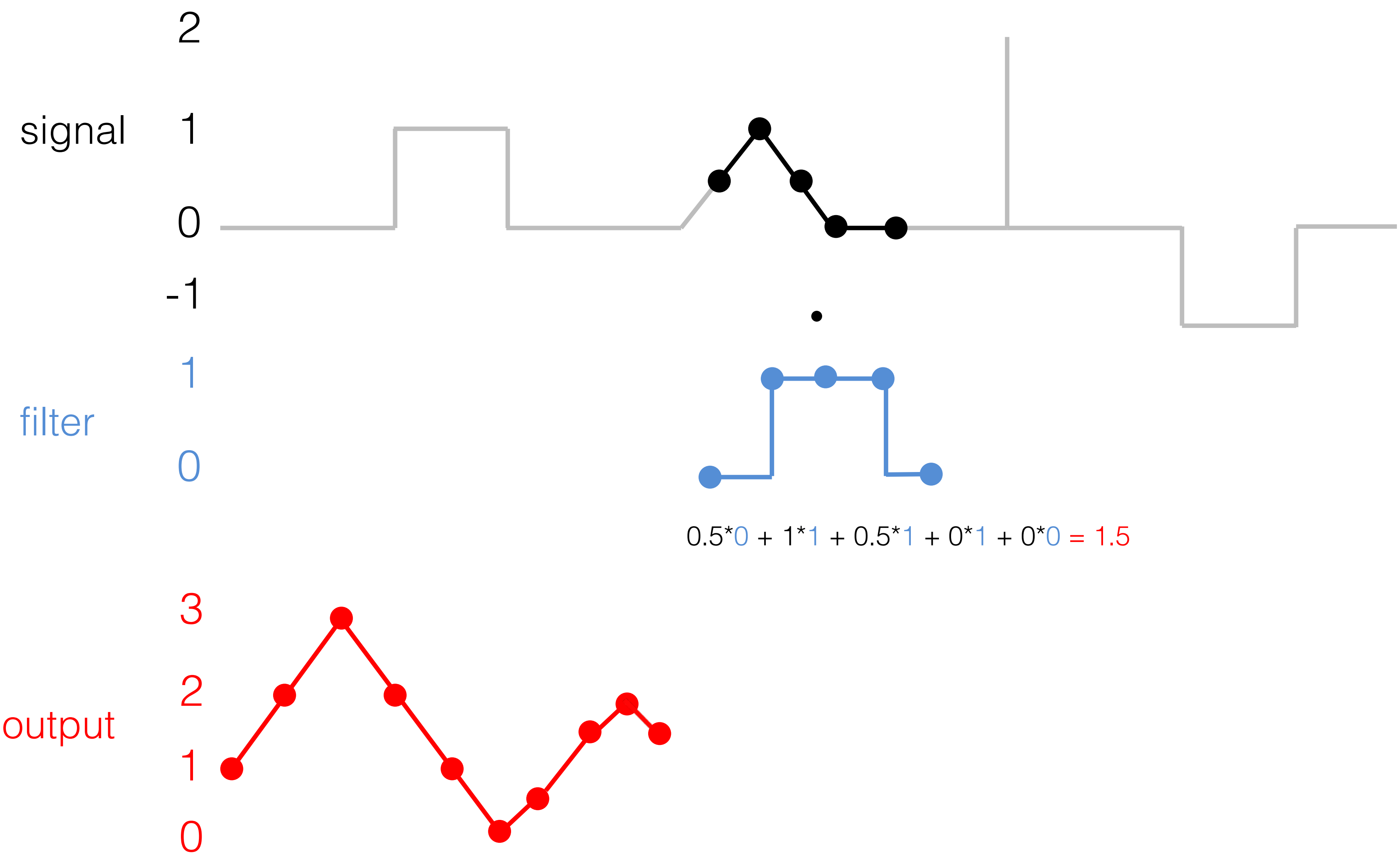
MATLAB: `conv(signal, filter, 'valid')`

Convolution



MATLAB: `conv(signal, filter, 'valid')`

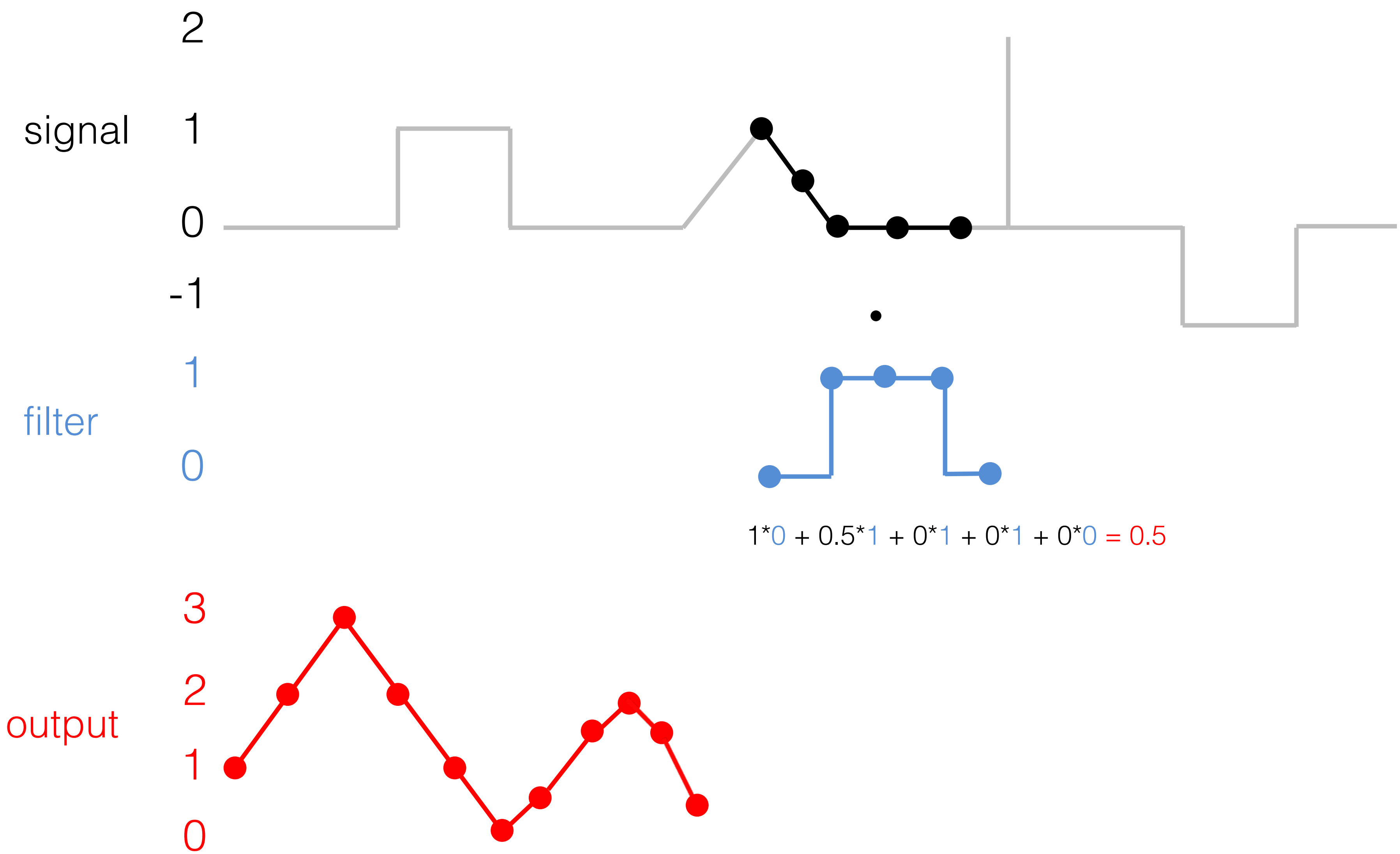
Convolution



0.5*0 + 1*1 + 0.5*1 + 0*1 + 0*0 = 1.5

MATLAB: `conv(signal, filter, ‘valid’)`

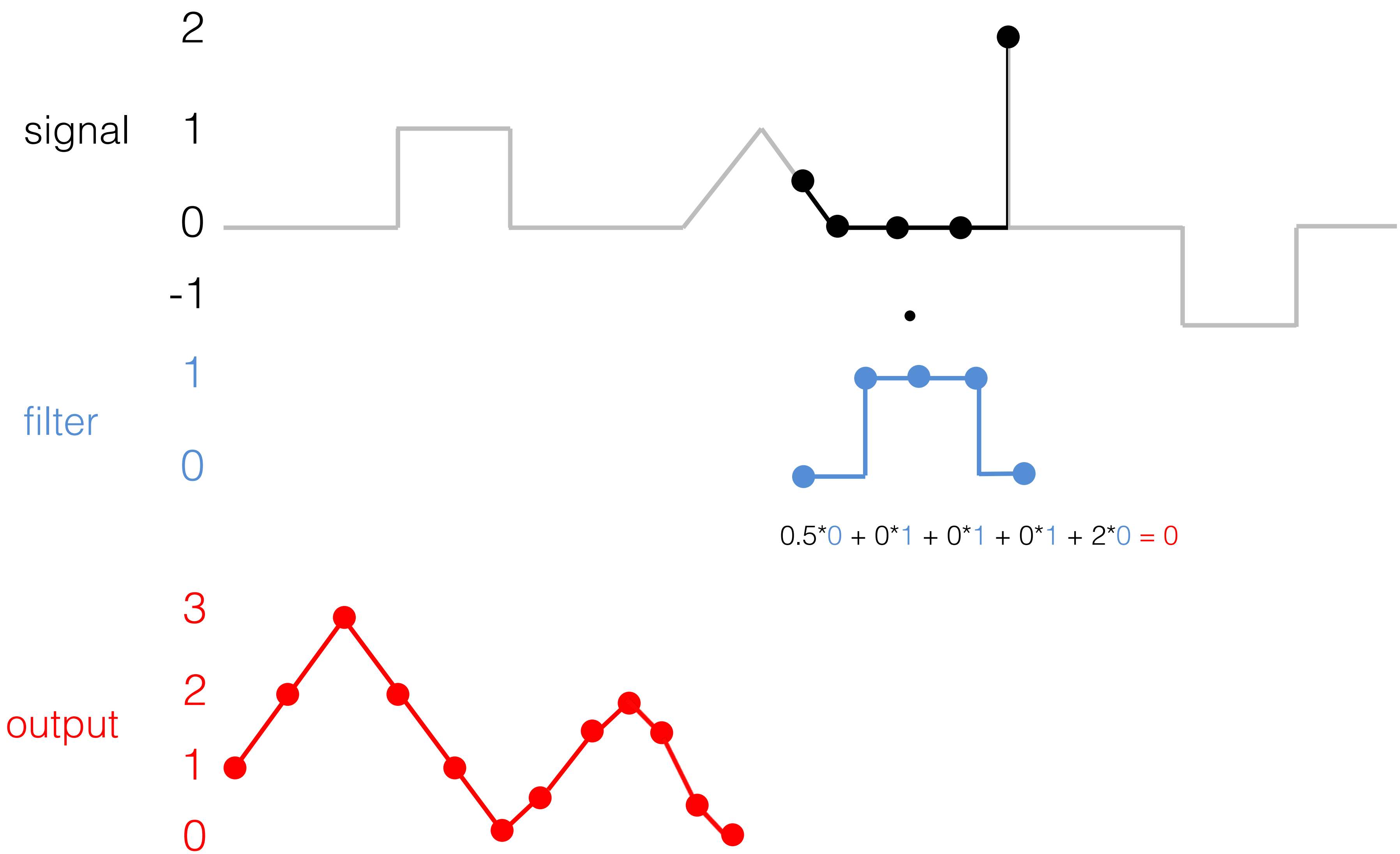
Convolution



1*0 + 0.5*1 + 0*1 + 0*1 + 0*0 = 0.5

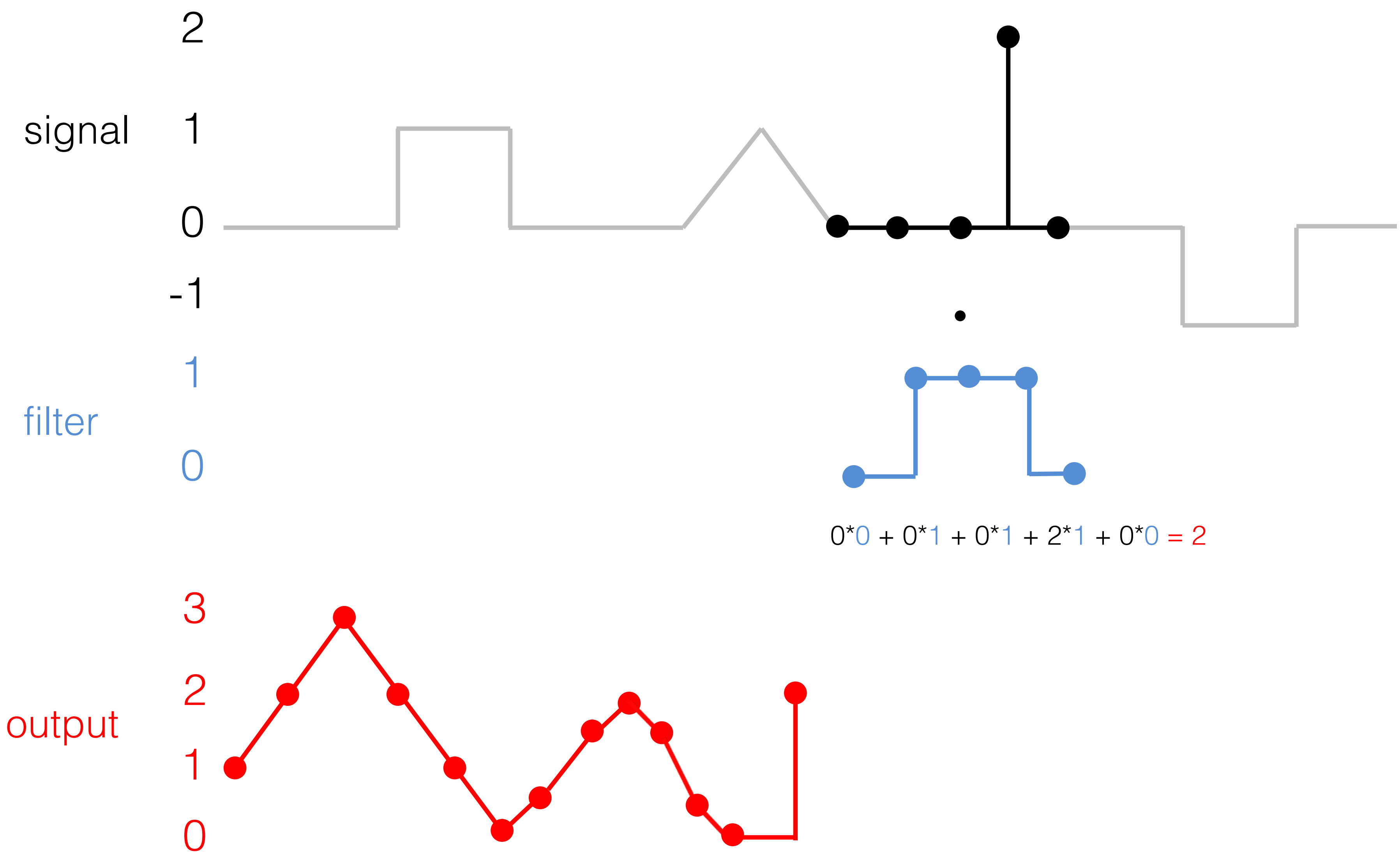
MATLAB: `conv(signal, filter, 'valid')`

Convolution



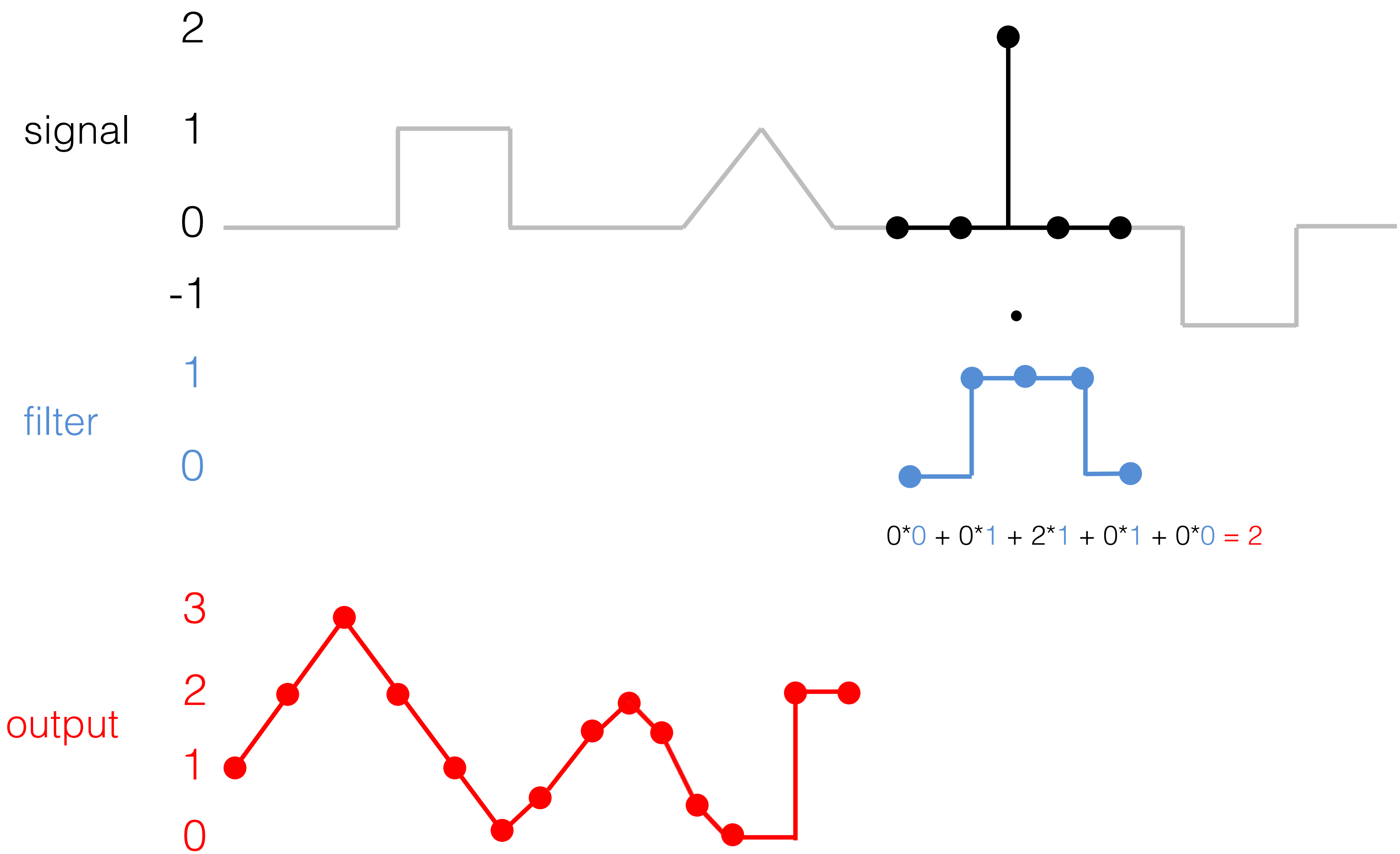
MATLAB: `conv(signal, filter, ‘valid’)`

Convolution



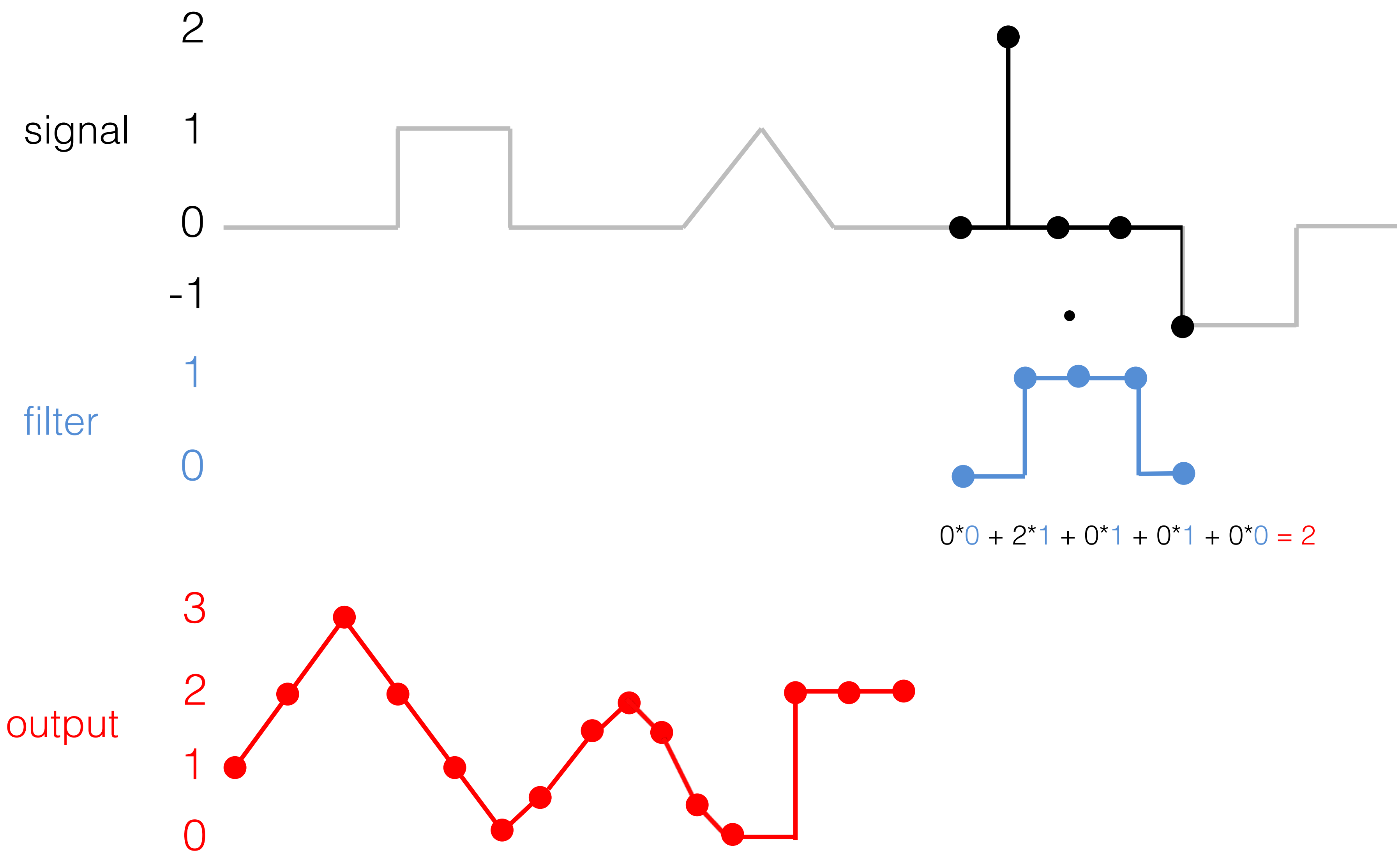
MATLAB: `conv(signal, filter, 'valid')`

Convolution



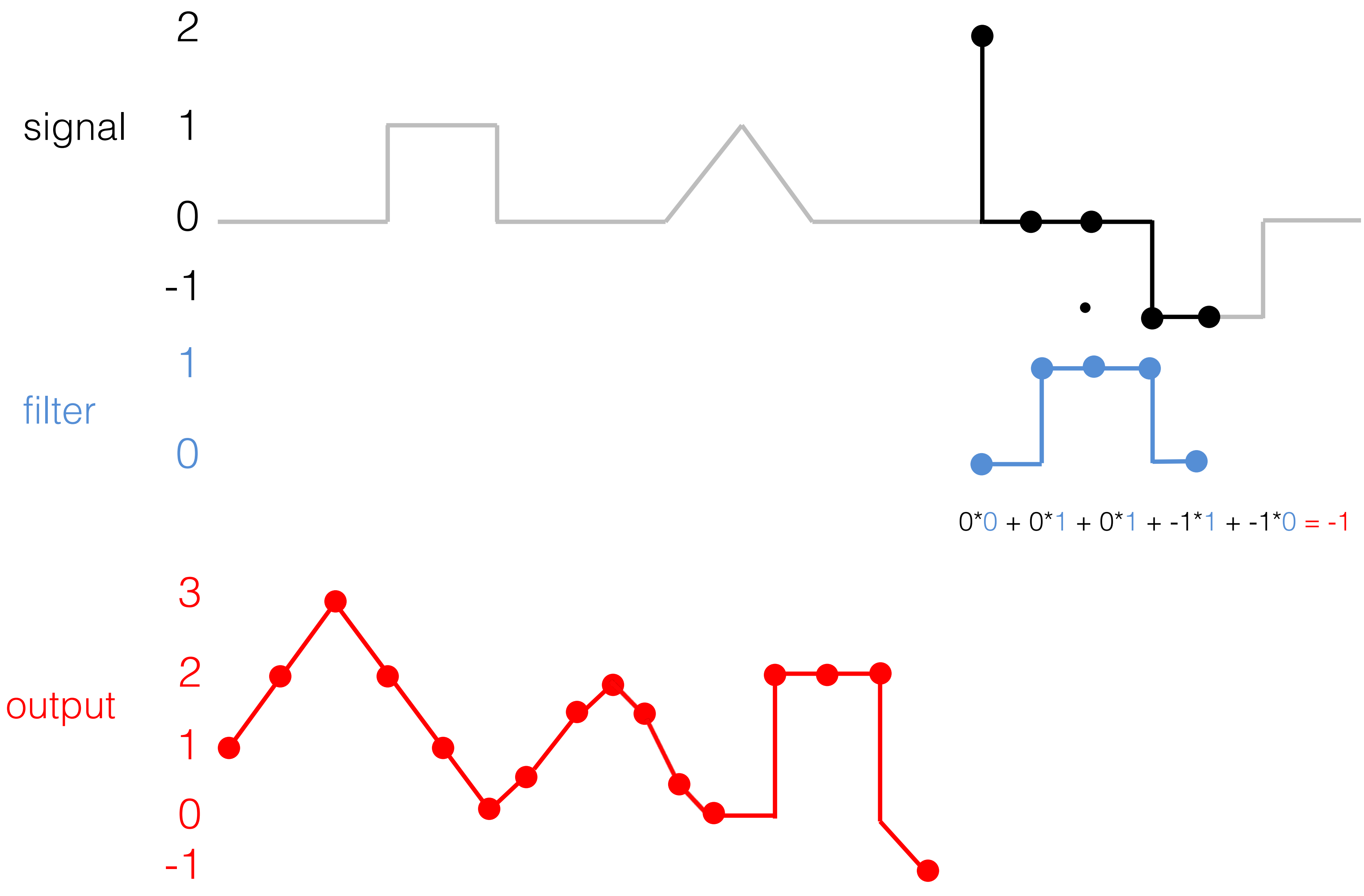
MATLAB: `conv(signal, filter, 'valid')`

Convolution

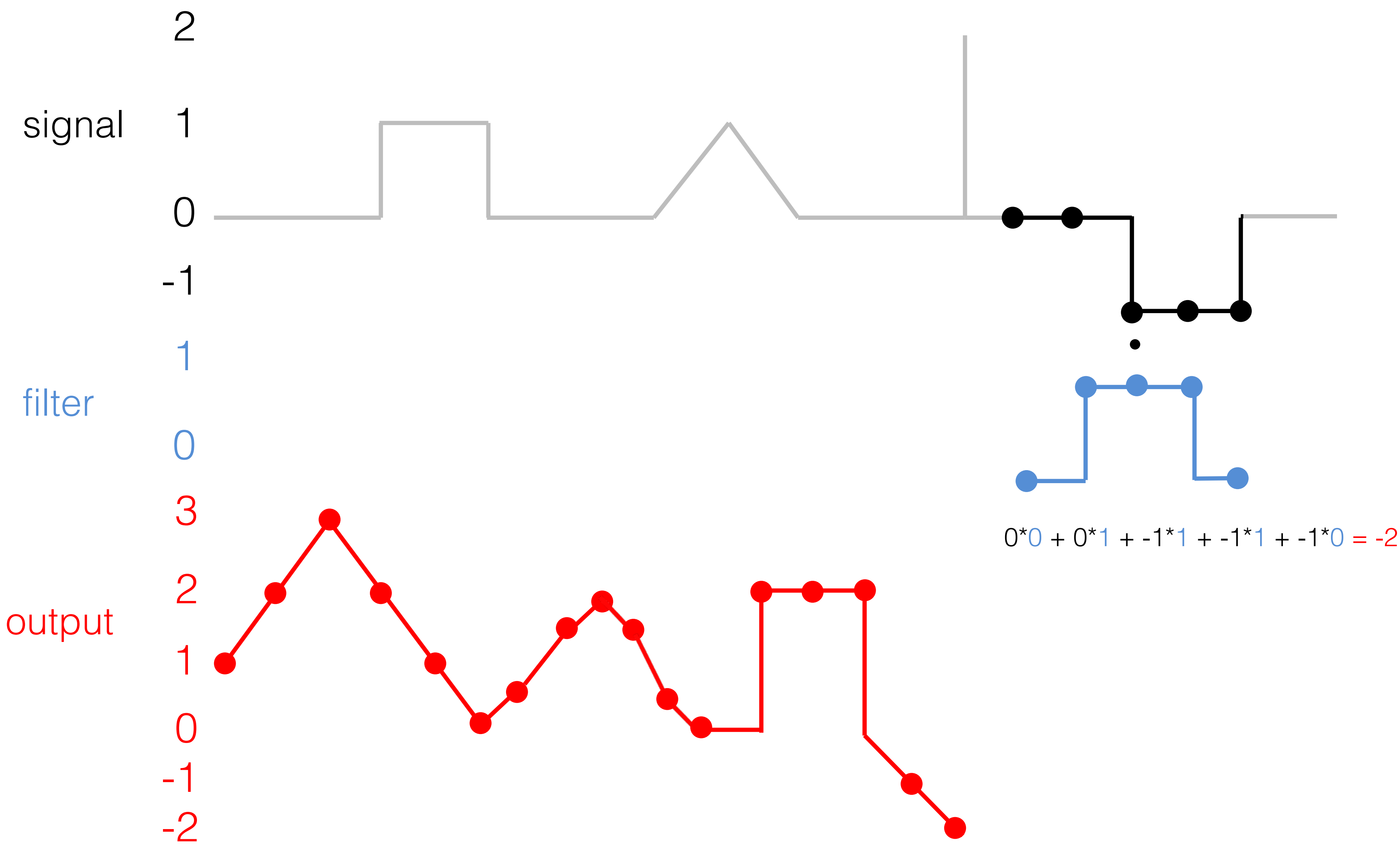


MATLAB: `conv(signal, filter, 'valid')`

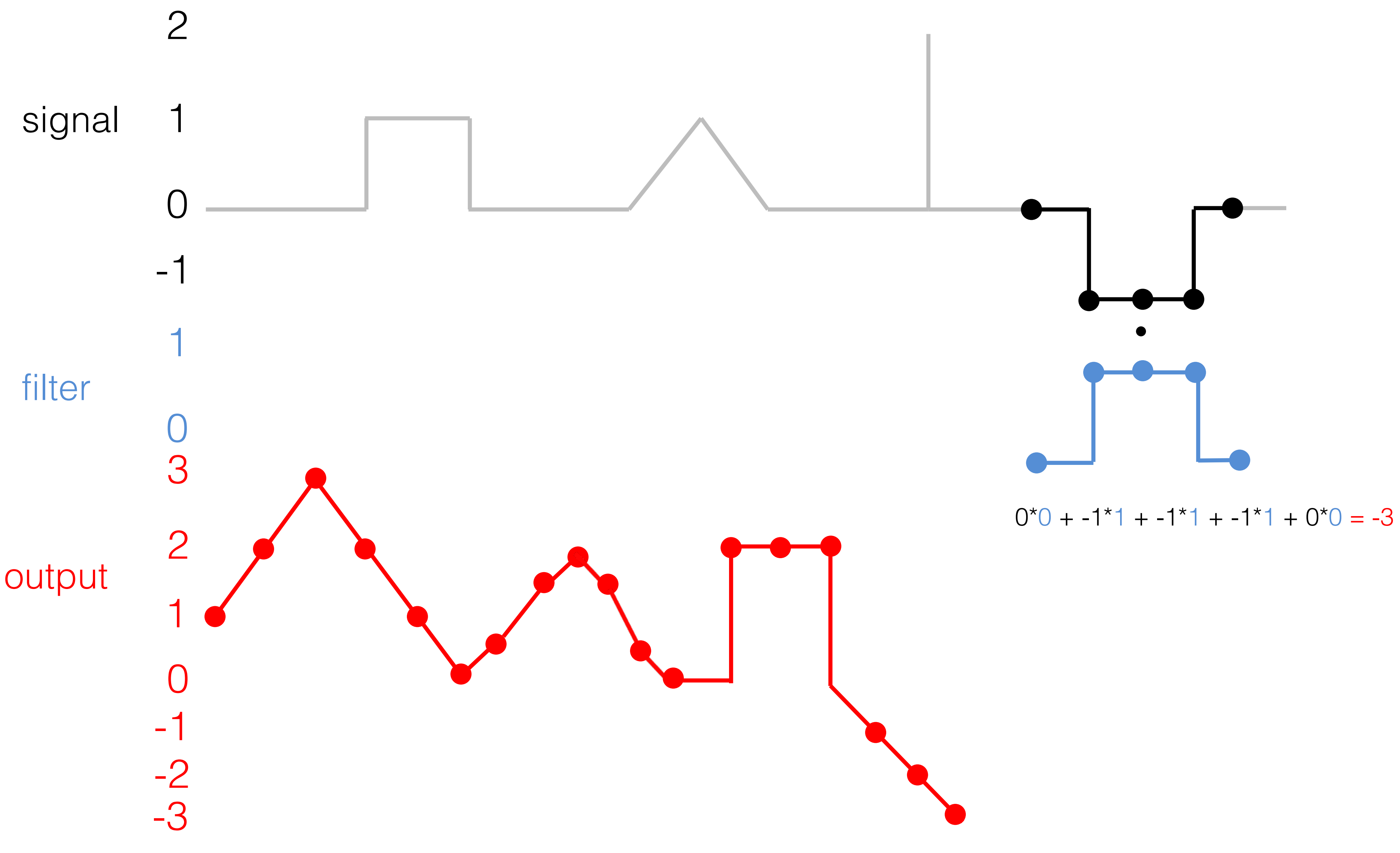
Convolution



Convolution

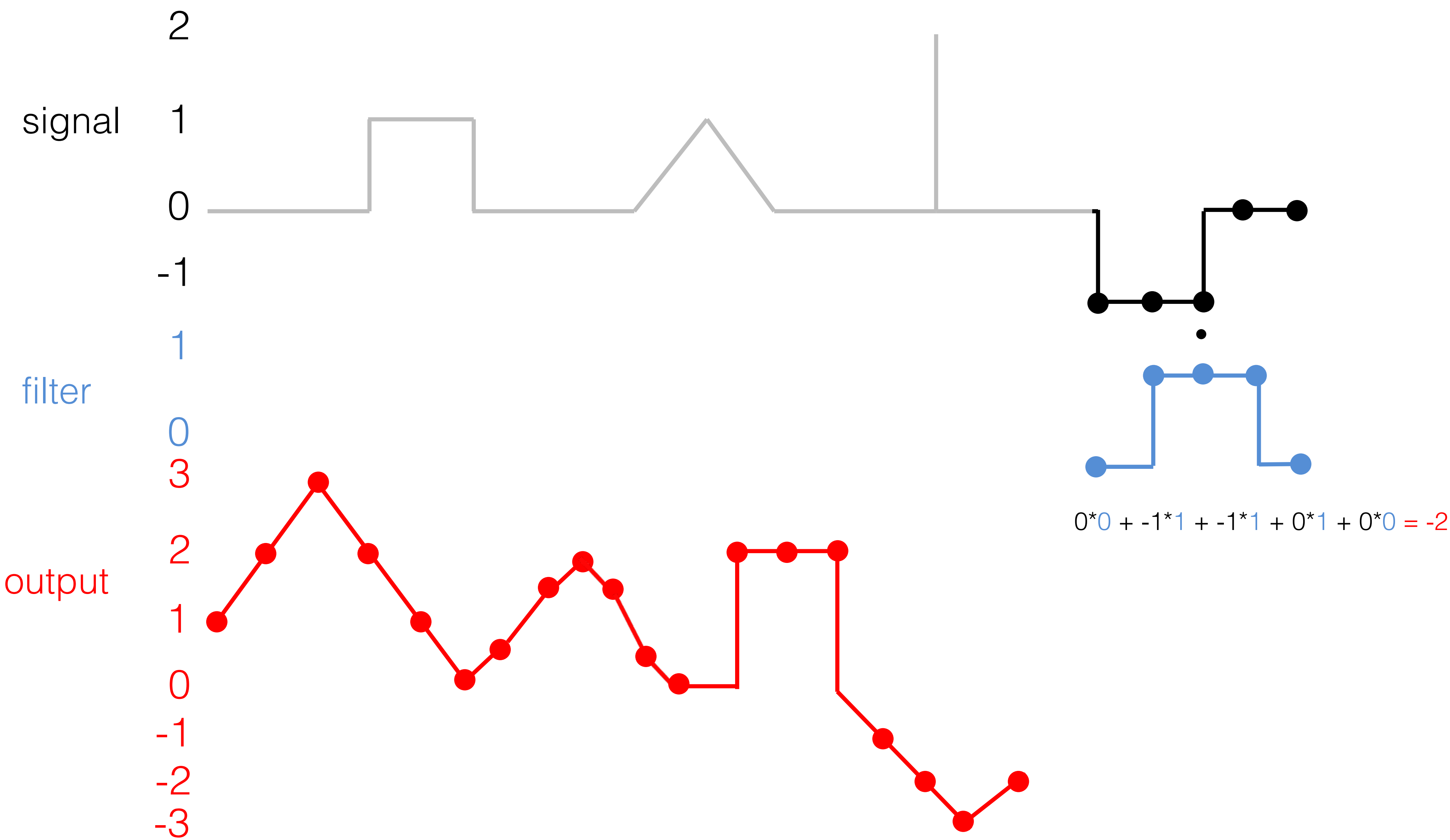


Convolution



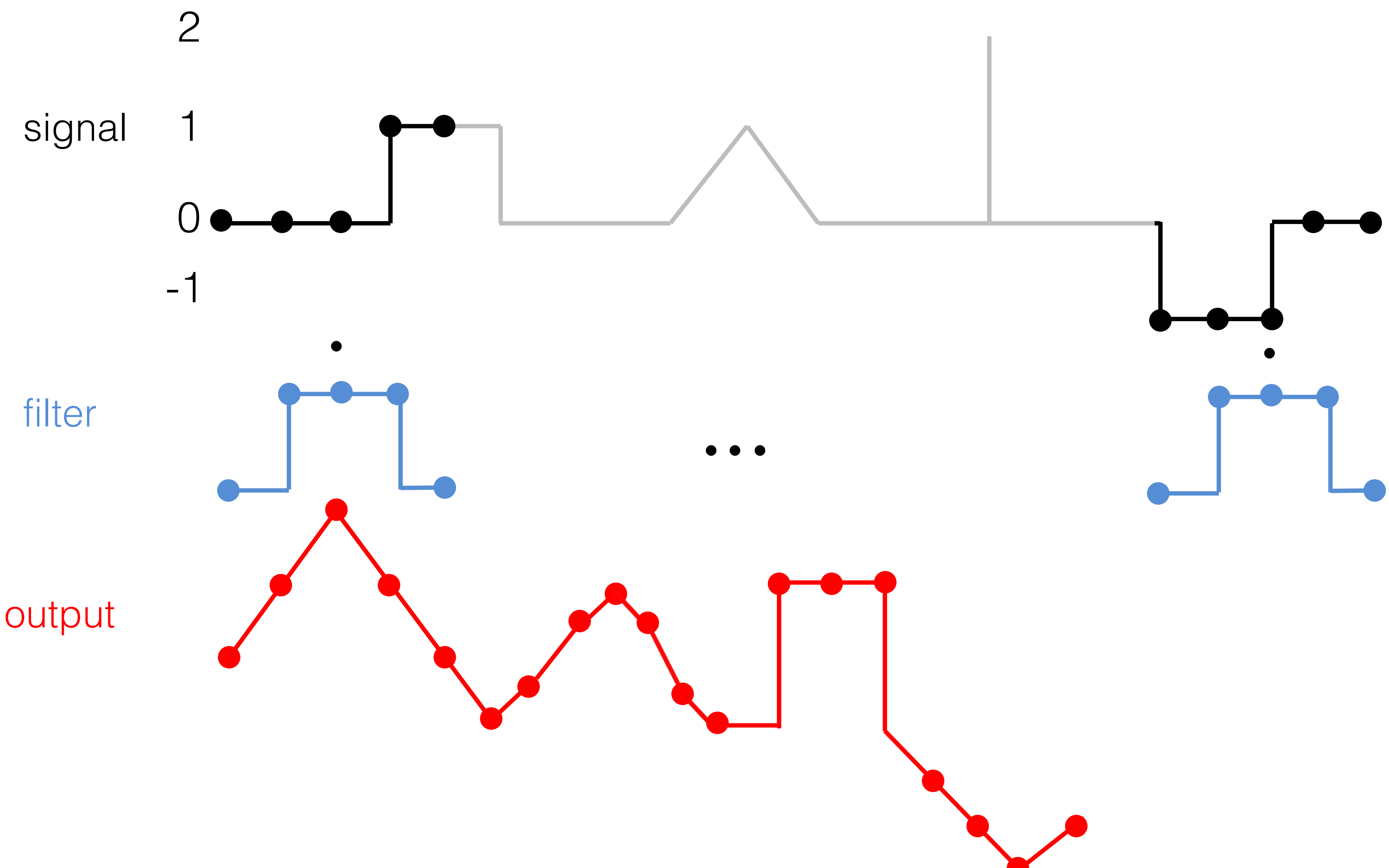
MATLAB: `conv(signal, filter, 'valid')`

Convolution



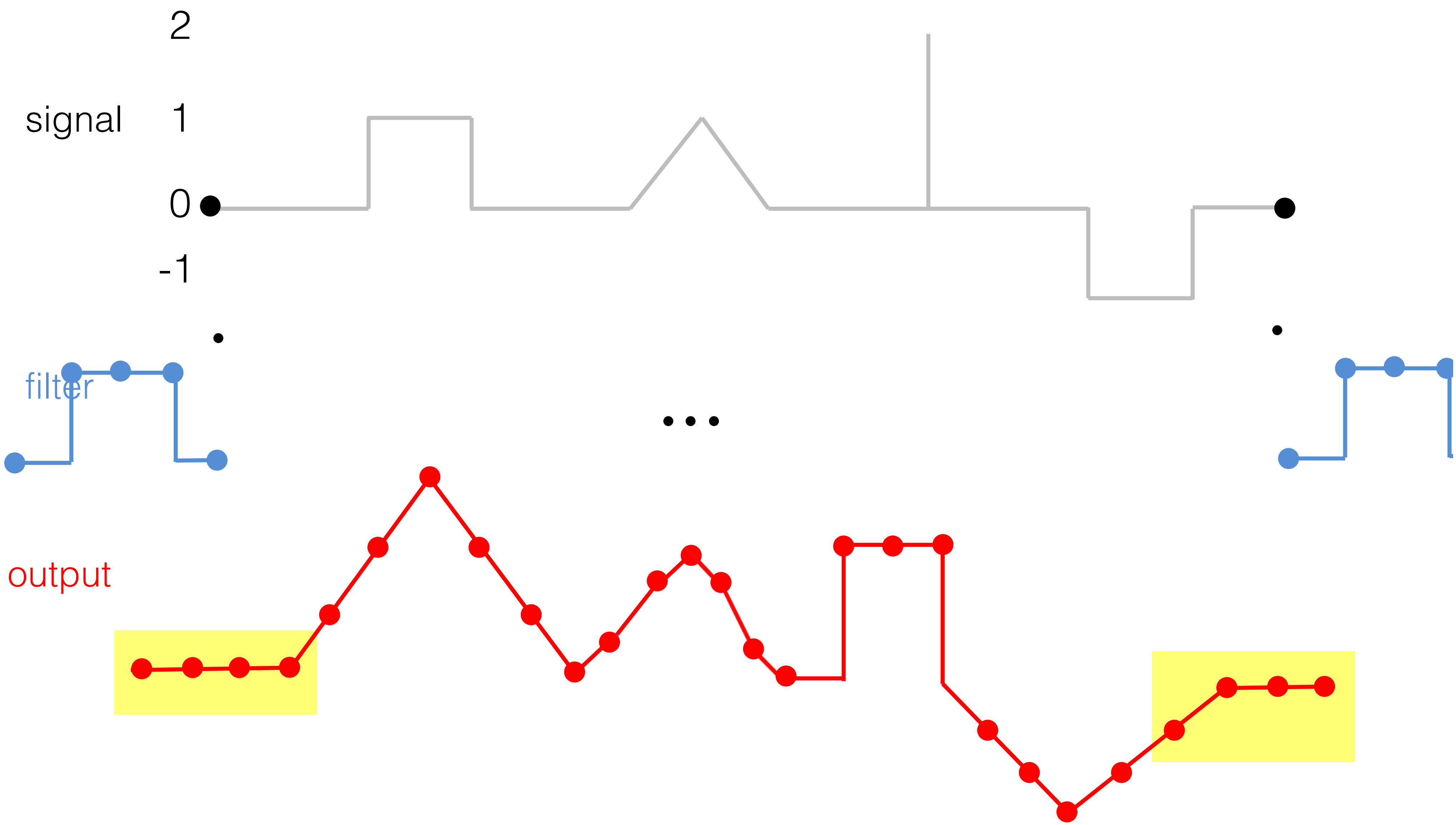
MATLAB: `conv(signal, filter, 'valid')`

Convolution



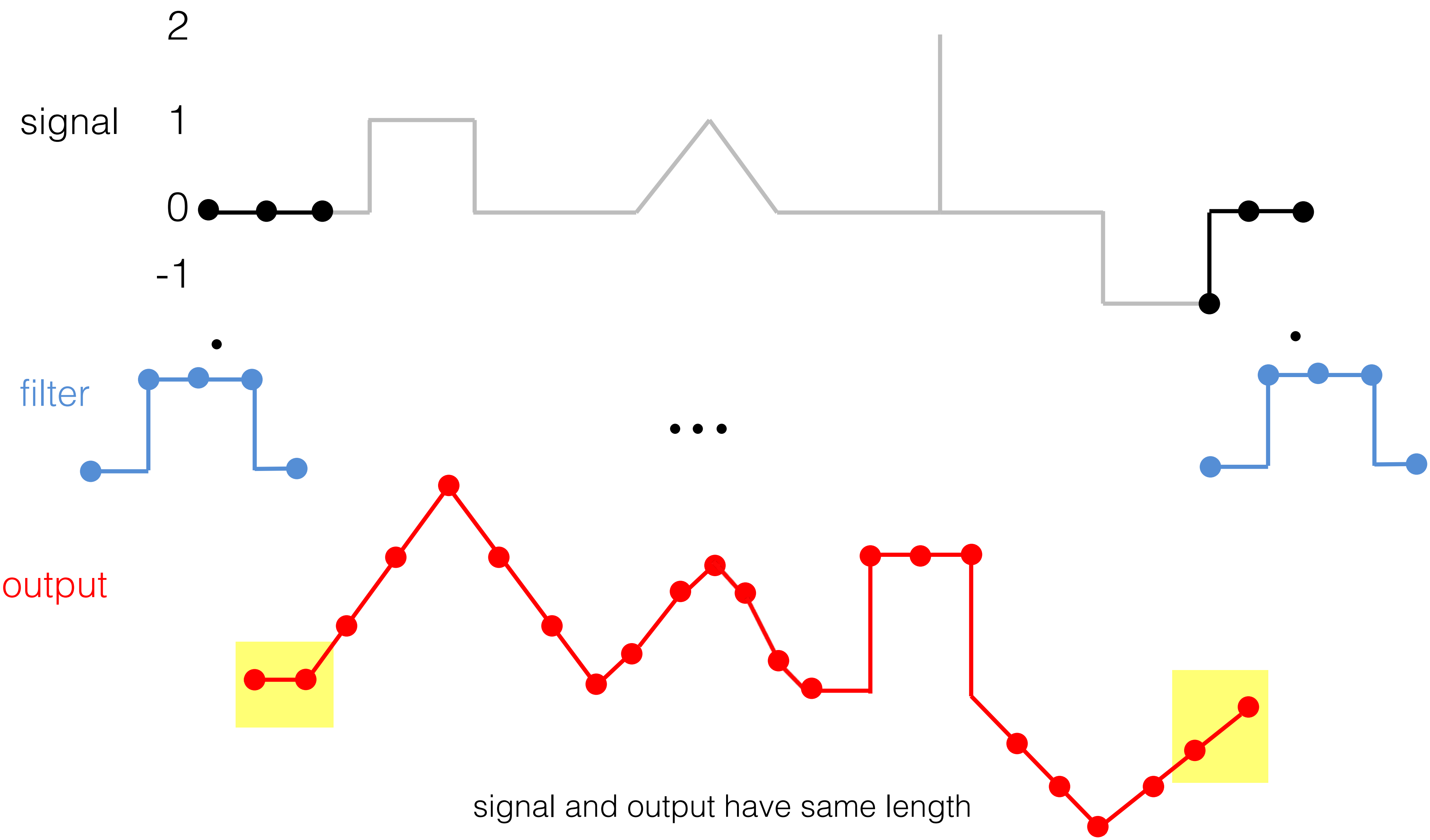
MATLAB: `conv(signal, filter, 'full')`

Convolution



MATLAB: `conv(signal, filter, 'same')`

Convolution

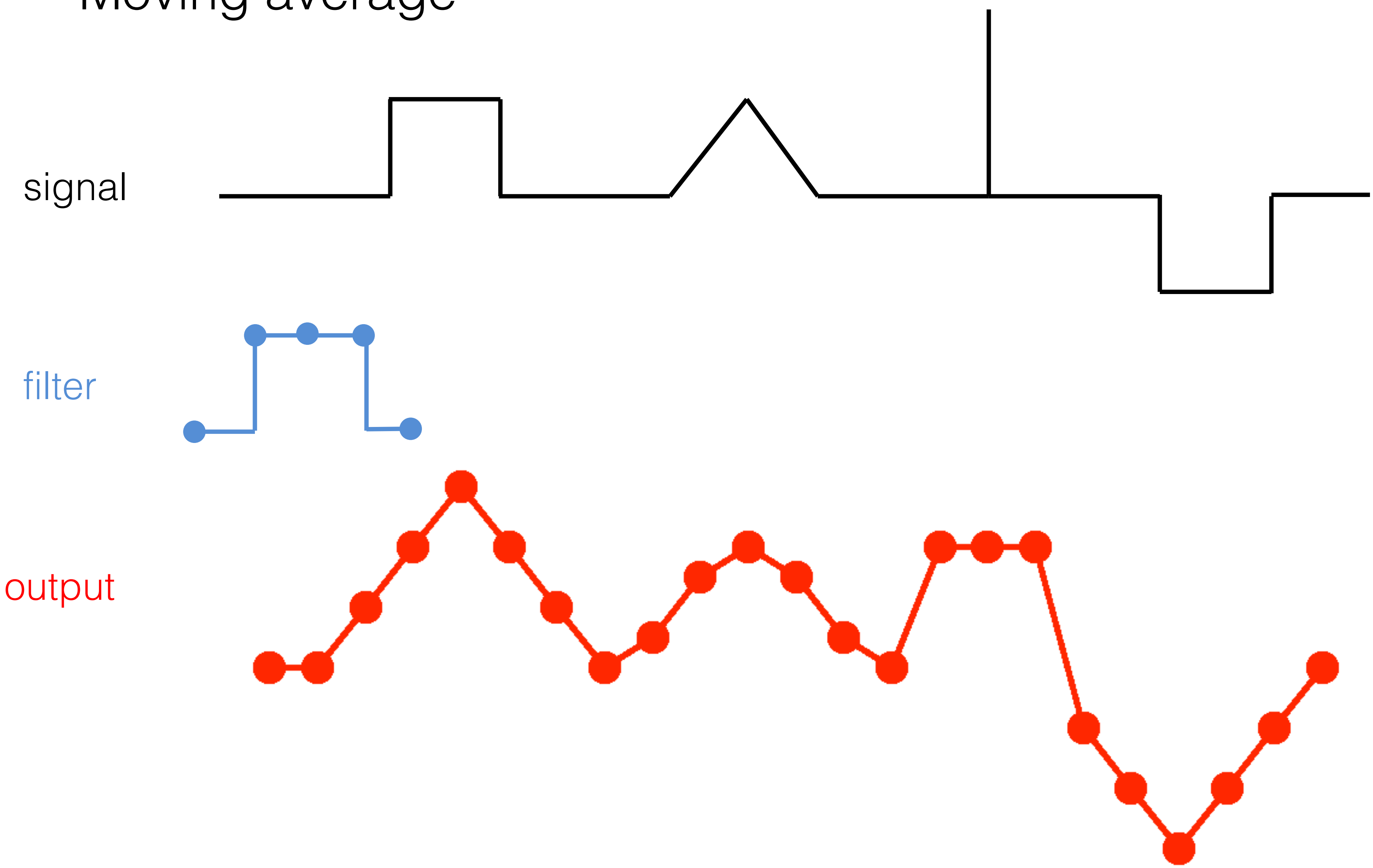


signal and output have same length

MATLAB: `conv(signal, filter, 'same')`

Convolution examples

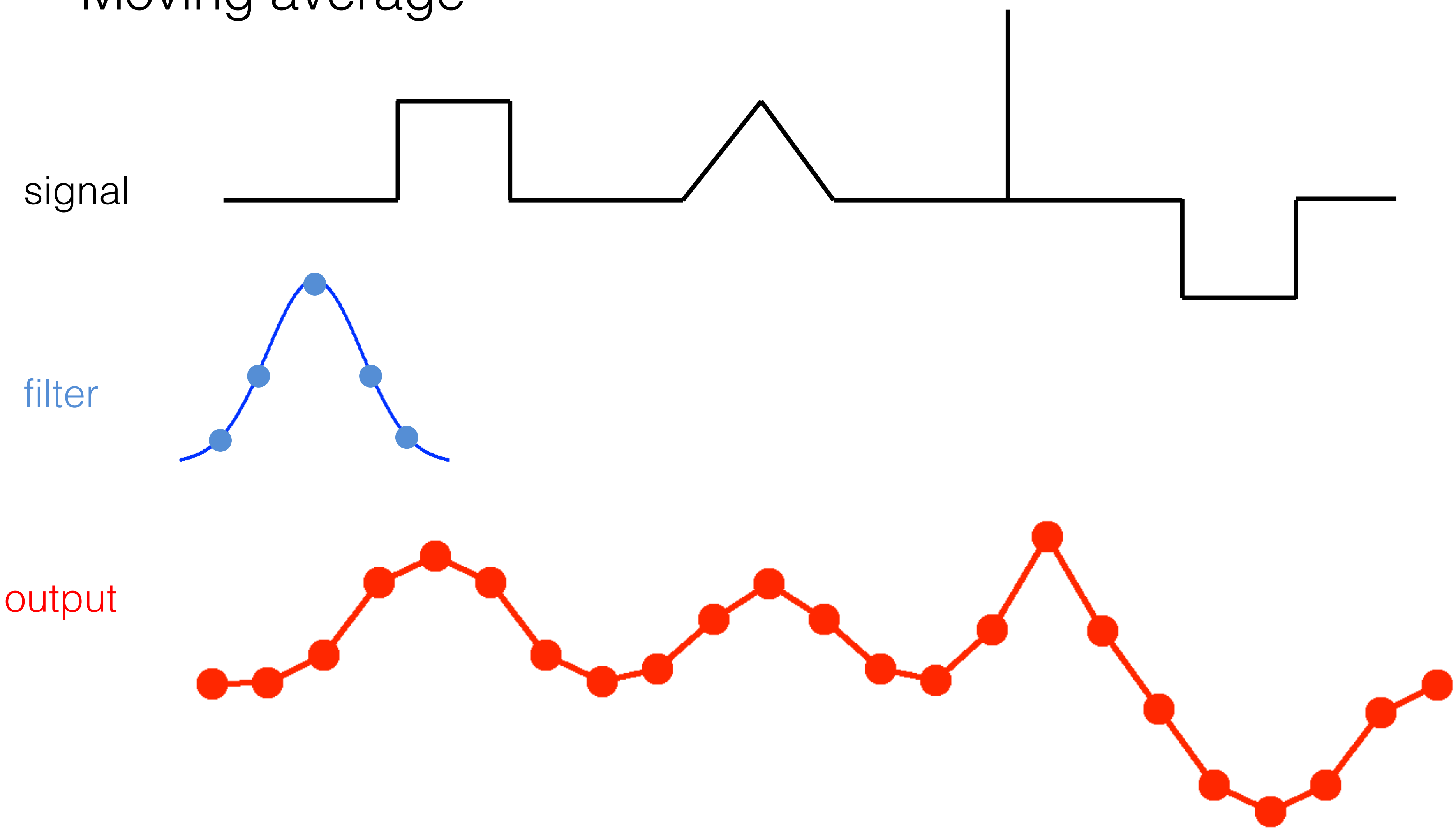
Moving average



MATLAB: `conv(signal, filter, 'same')`

Convolution examples

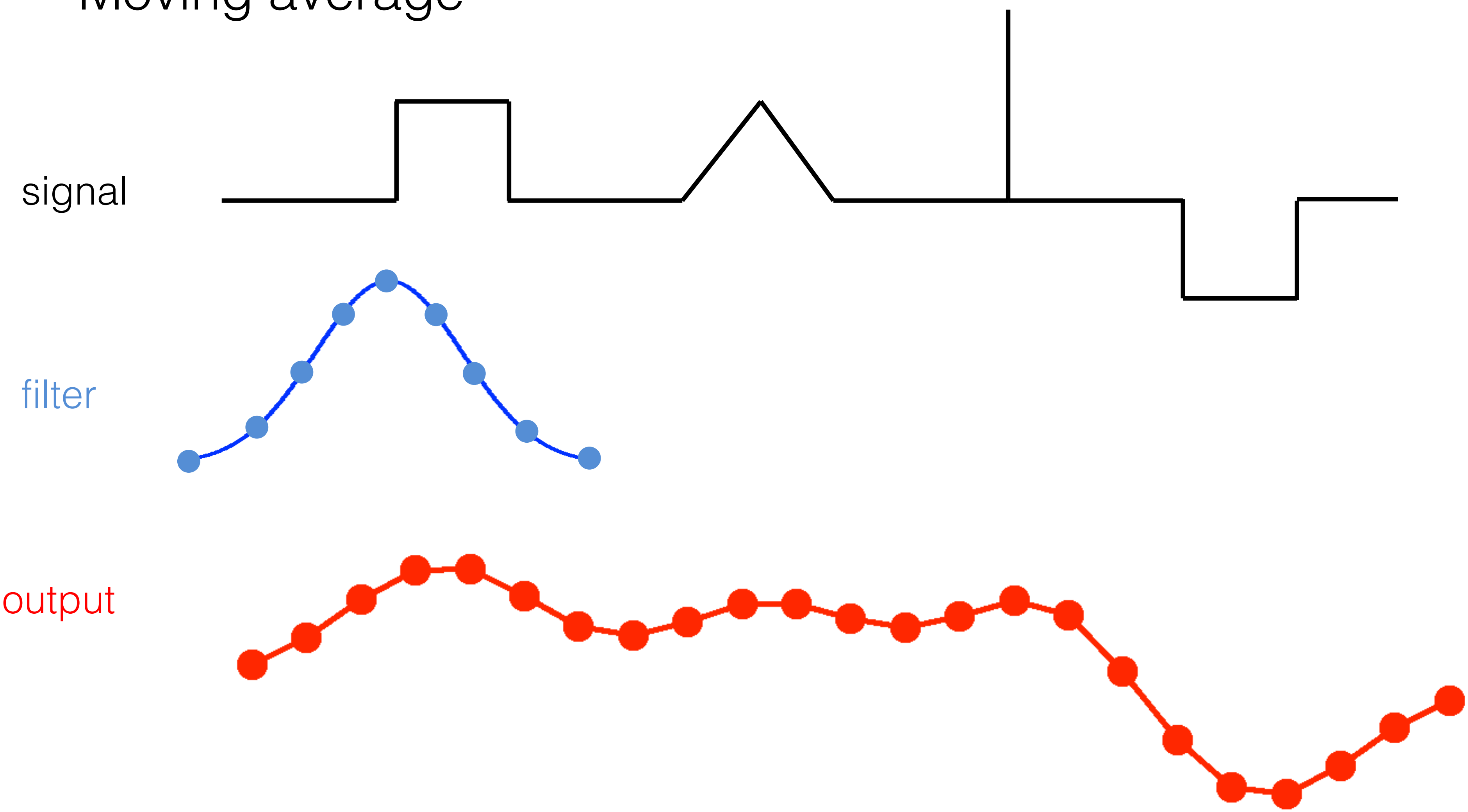
Moving average



MATLAB: `conv(signal, filter, 'same')`

Convolution examples

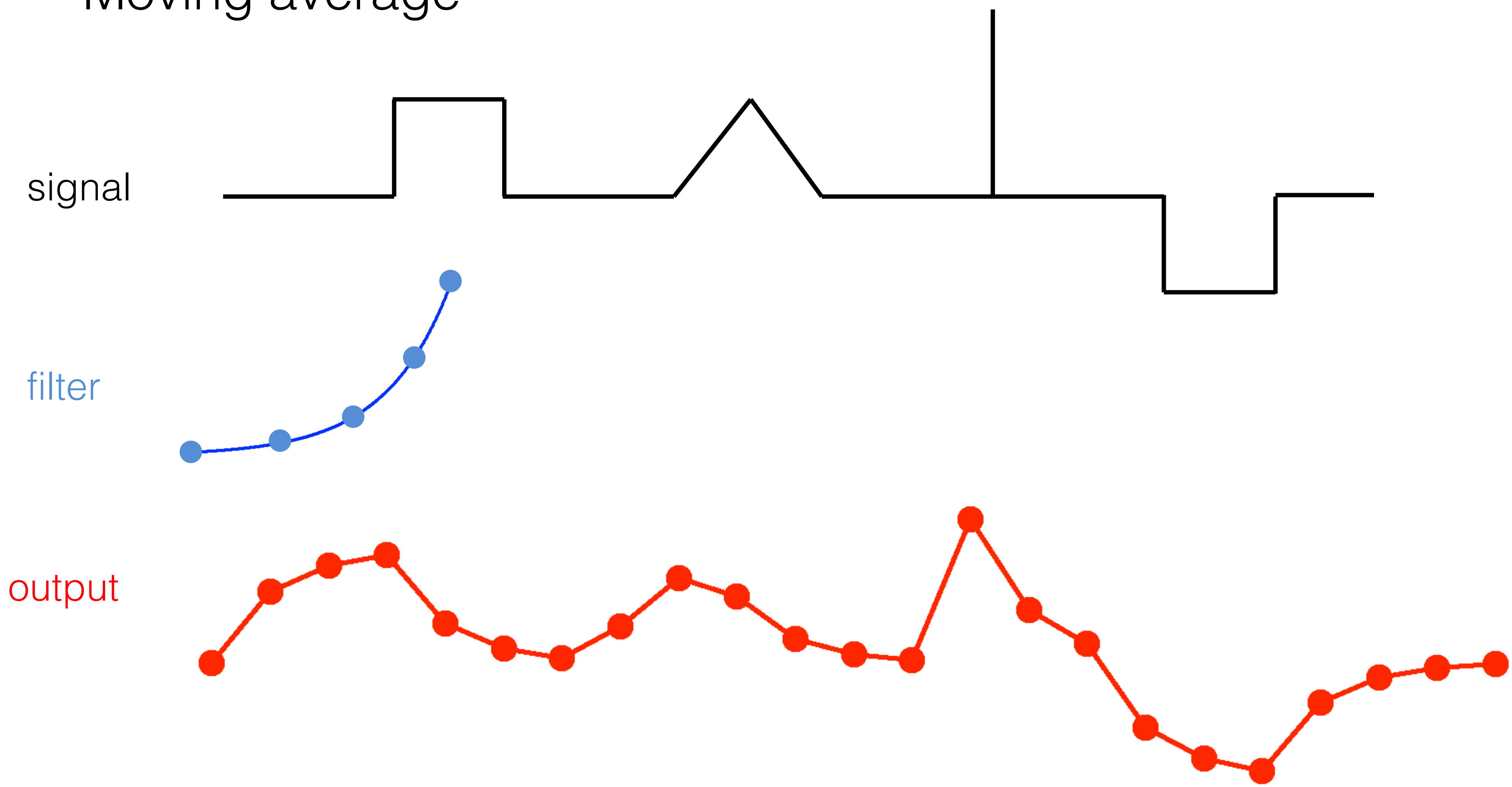
Moving average



MATLAB: `conv(signal, fliplr(filter), ‘same’)`

Convolution examples

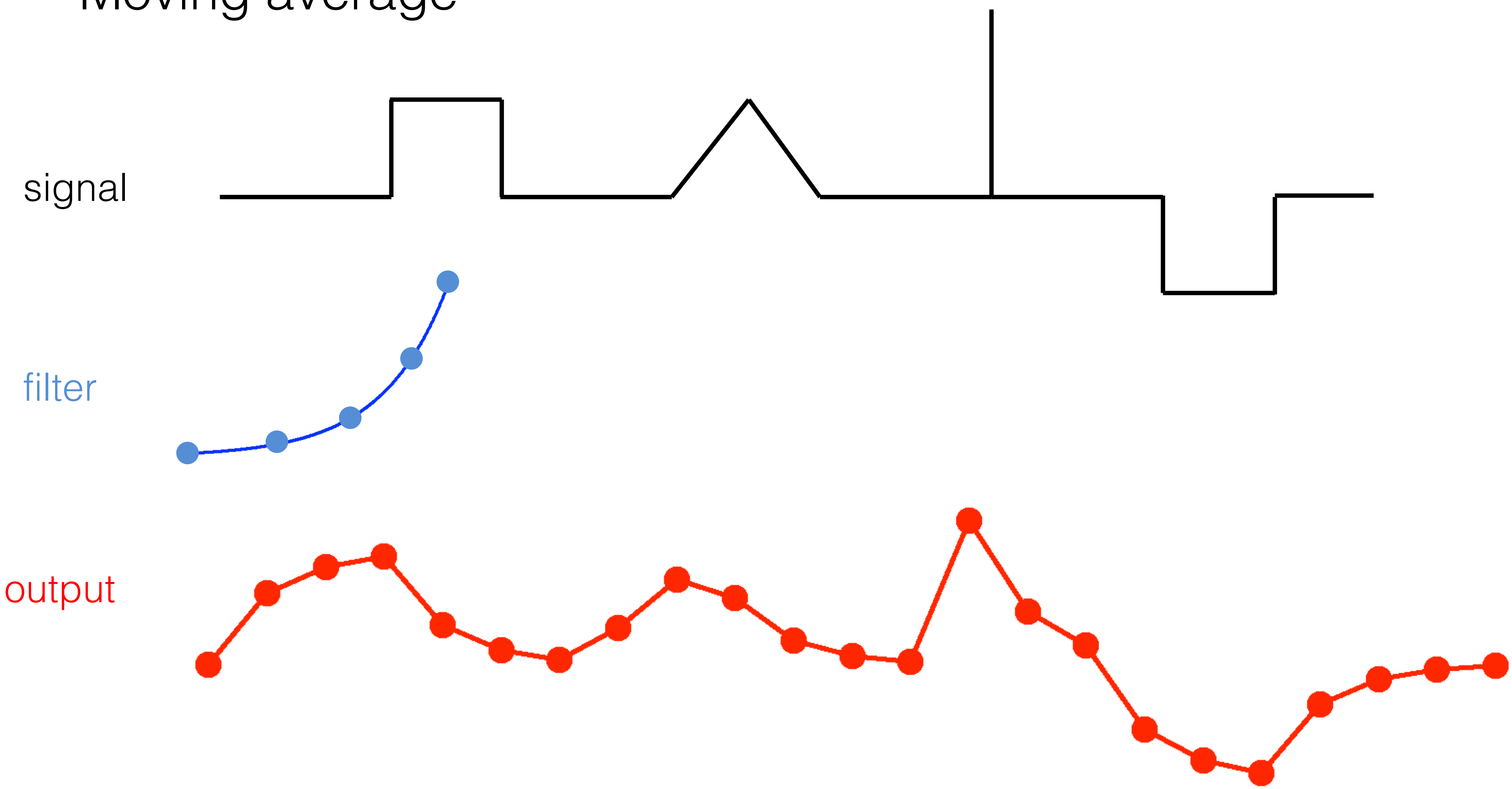
Moving average



MATLAB: `conv(signal, fliplr(filter), 'same')`

Convolution examples

Moving average

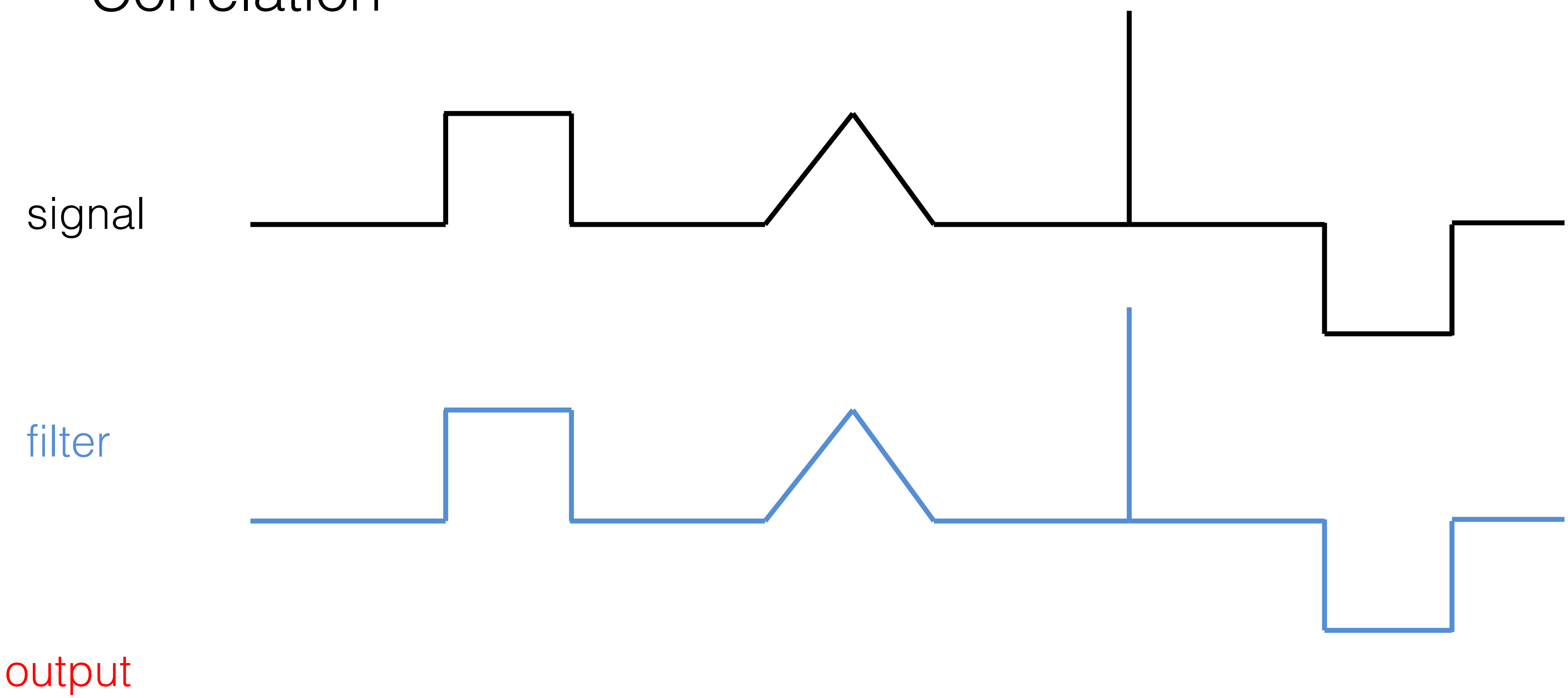


a note about using asymmetric filters in MATLAB

MATLAB: `conv(signal, fliplr(signal), 'same')`

Convolution examples

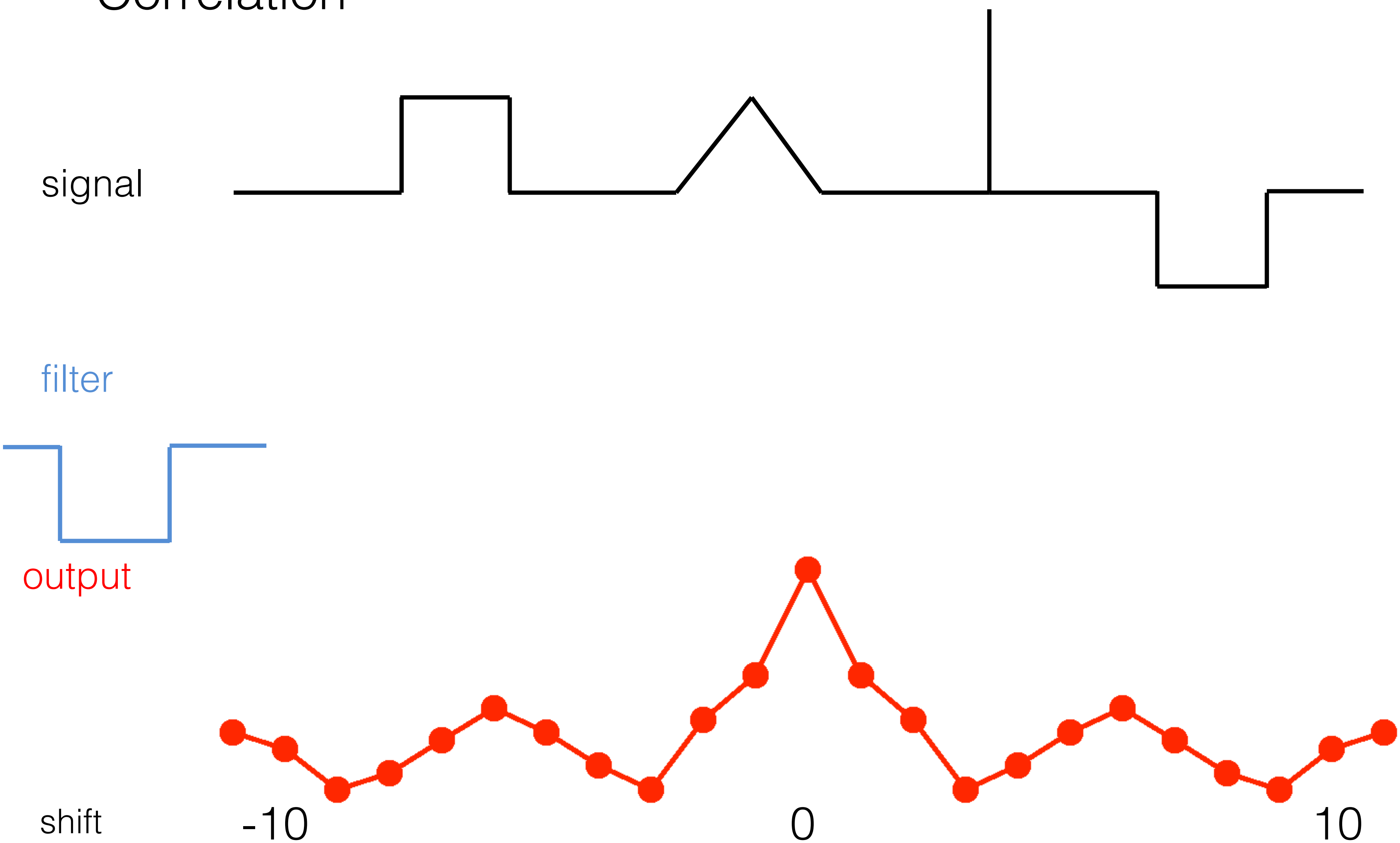
Correlation



MATLAB: `conv(signal, fliplr(signal), 'same')`

Convolution examples

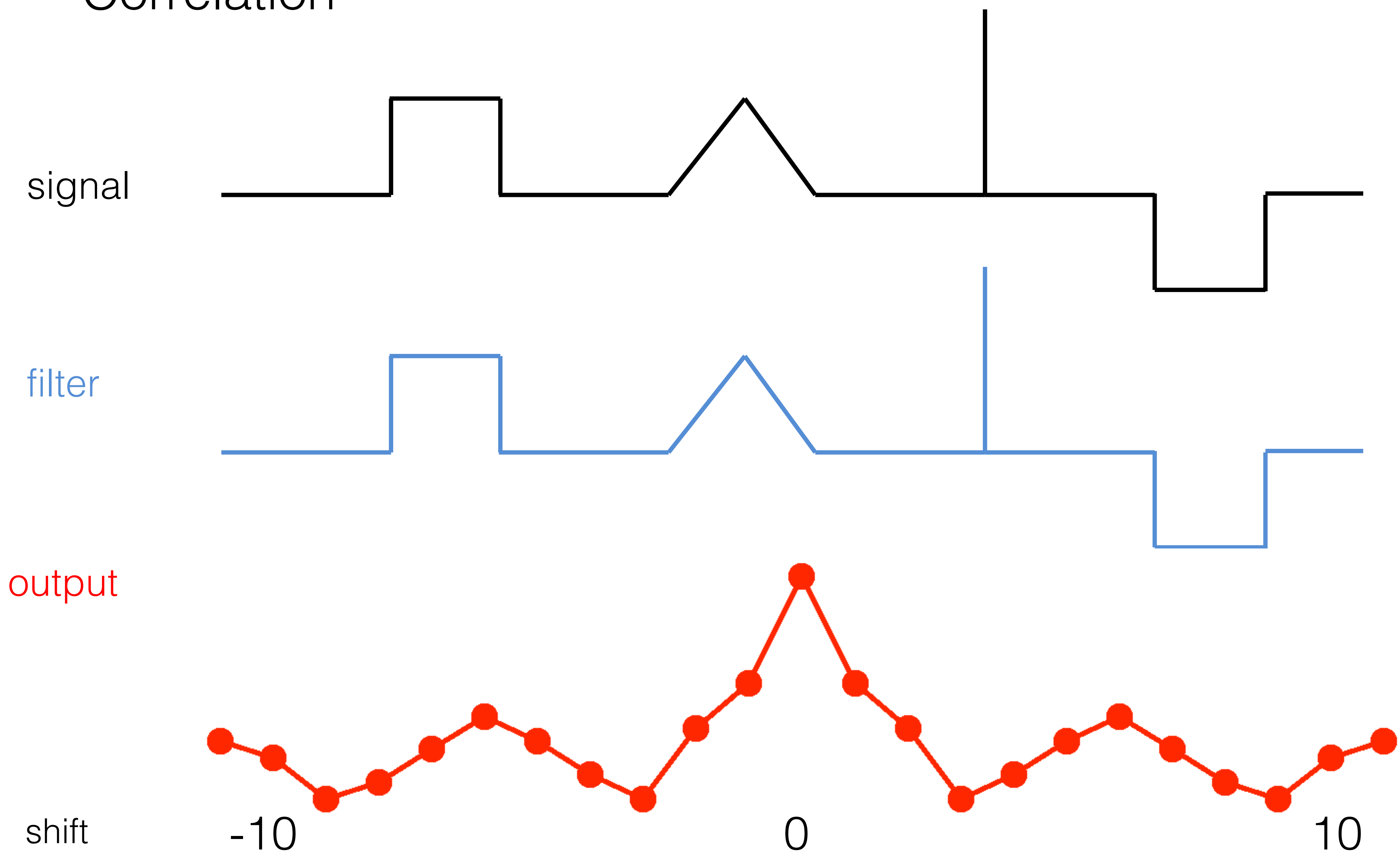
Correlation



MATLAB: `conv(signal, fliplr(signal), 'same')`

Convolution examples

Correlation



MATLAB: `conv2(signal, filter, 'same')`

2d Convolution examples

Image processing

signal

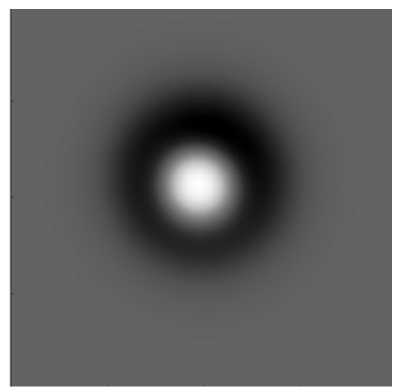
MATLAB: `conv2(signal, filter, 'same')`

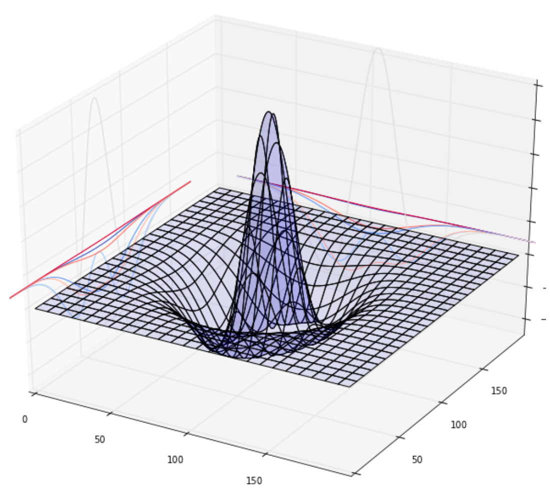
2d Convolution examples

Image processing

signal

filter





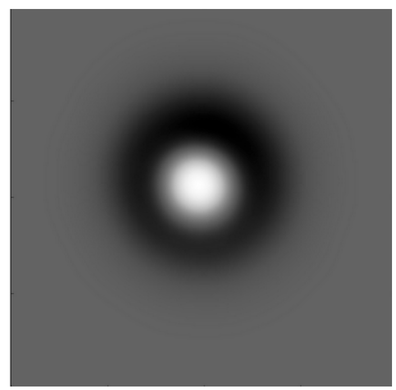
MATLAB: `conv2(signal, filter, 'same')`

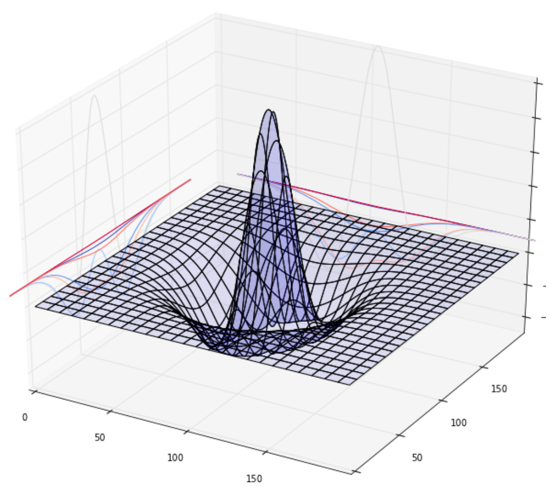
2d Convolution examples

Image processing

signal

filter





output

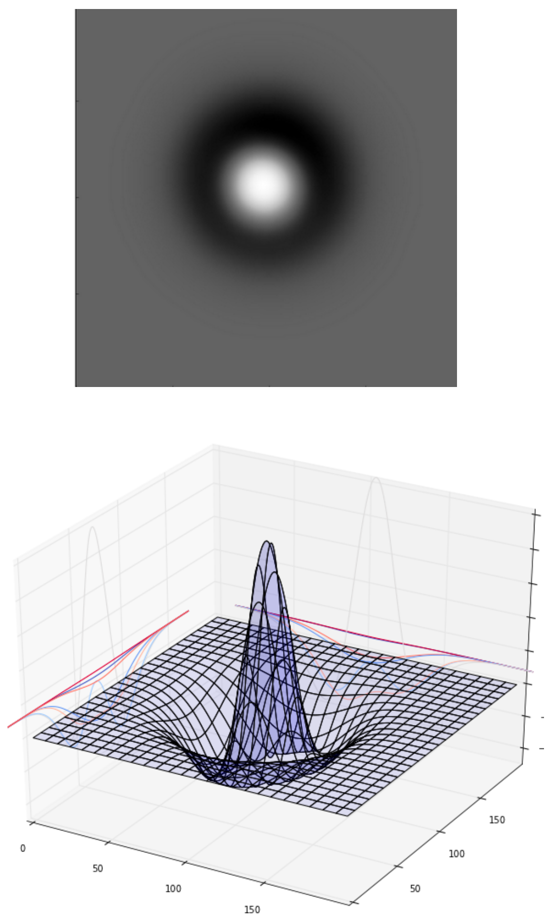
MATLAB: `conv2(signal, filter, 'same')`

2d Convolution examples

Image processing

signal

filter



output

Question

If you try smoothing your dataset with a Gaussian of 10 ms but your output looks noisy, should you try convolving your stimulus with a

a) wider Gaussian
b) narrower Gaussian

Question

If you try smoothing your dataset with a Gaussian of 10 ms but your output looks noisy, should you try convolving your stimulus with a

a) wider Gaussian

Question

To get the autocorrelation of a signal, you should do what first before convolving the signal with itself?

a) flip the copied signal
b) flip the copied signal and subtract the signal mean
c) flip the copied signal and divide by the signal std
d) find someone to help

Question

To get the autocorrelation of a signal, you should do what first before convolving the signal with itself?

b) flip the copied signal and subtract the signal mean

Today's lecture

When will this be useful?

Filtering in time or space

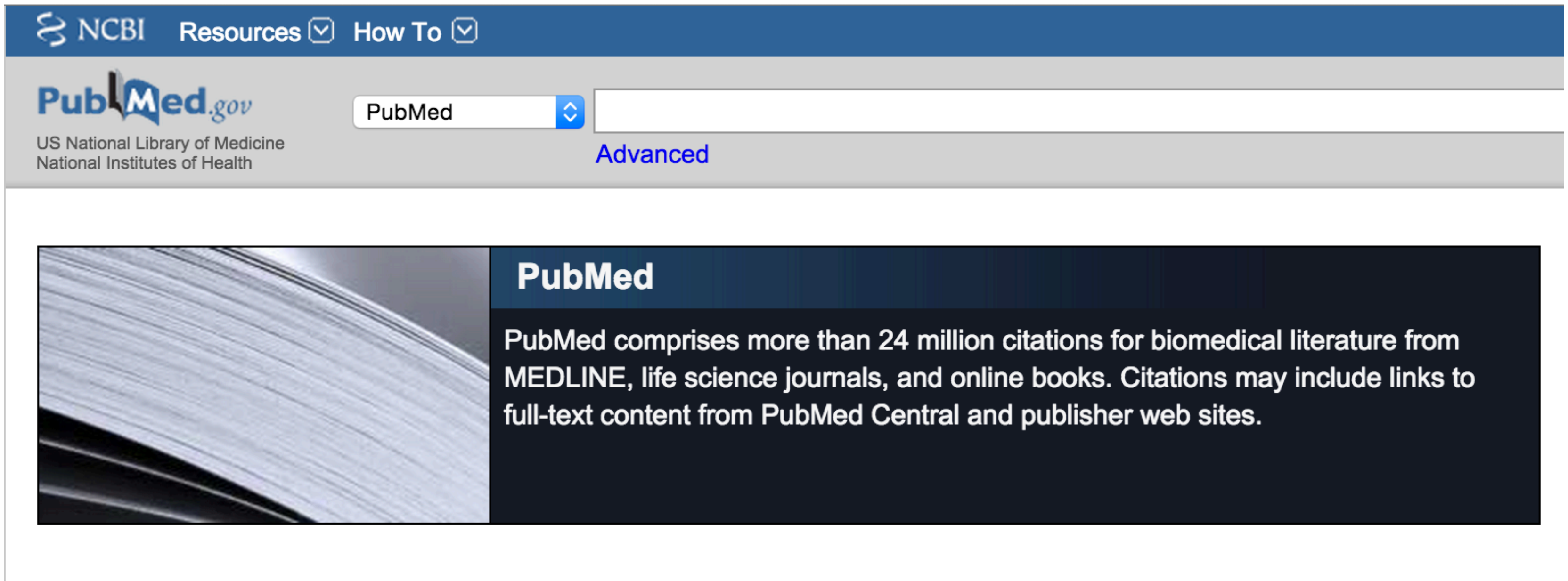
- What is convolution?
- How much should I filter?

Filtering in frequency

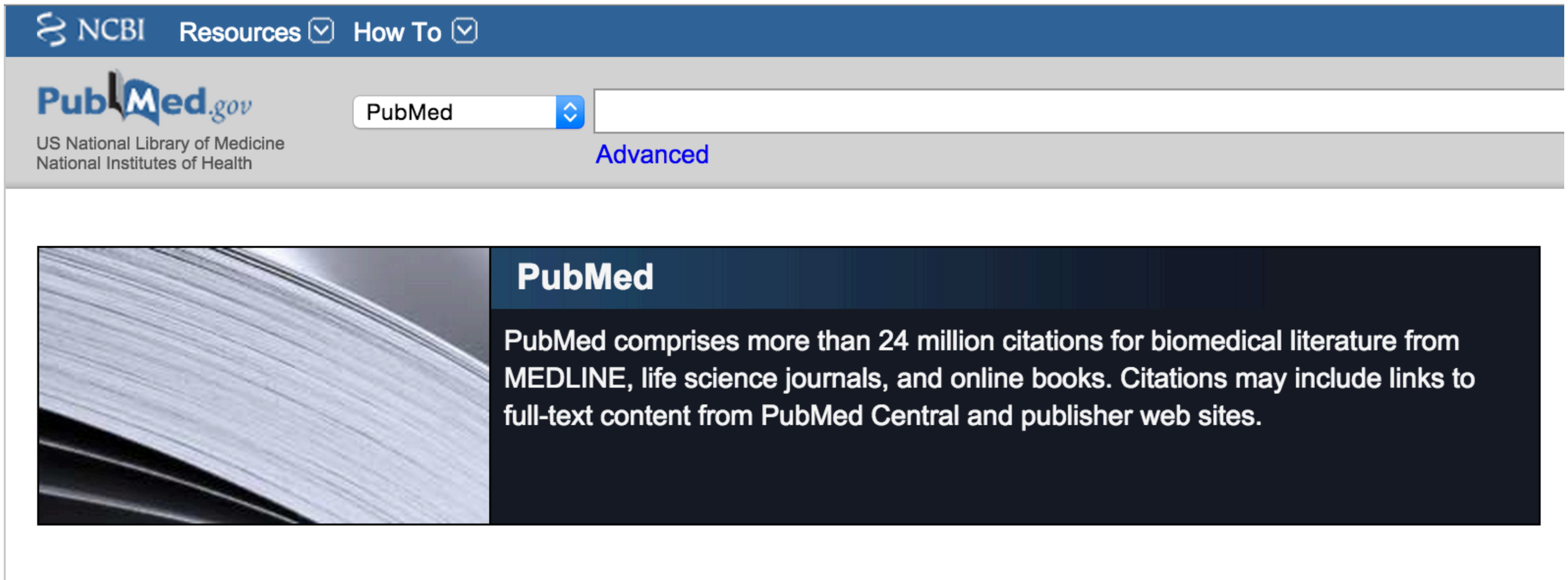
- Fourier series
- What is a Fourier transform?
- Designing filters in frequency domain

Fourier transform

Fourier transform



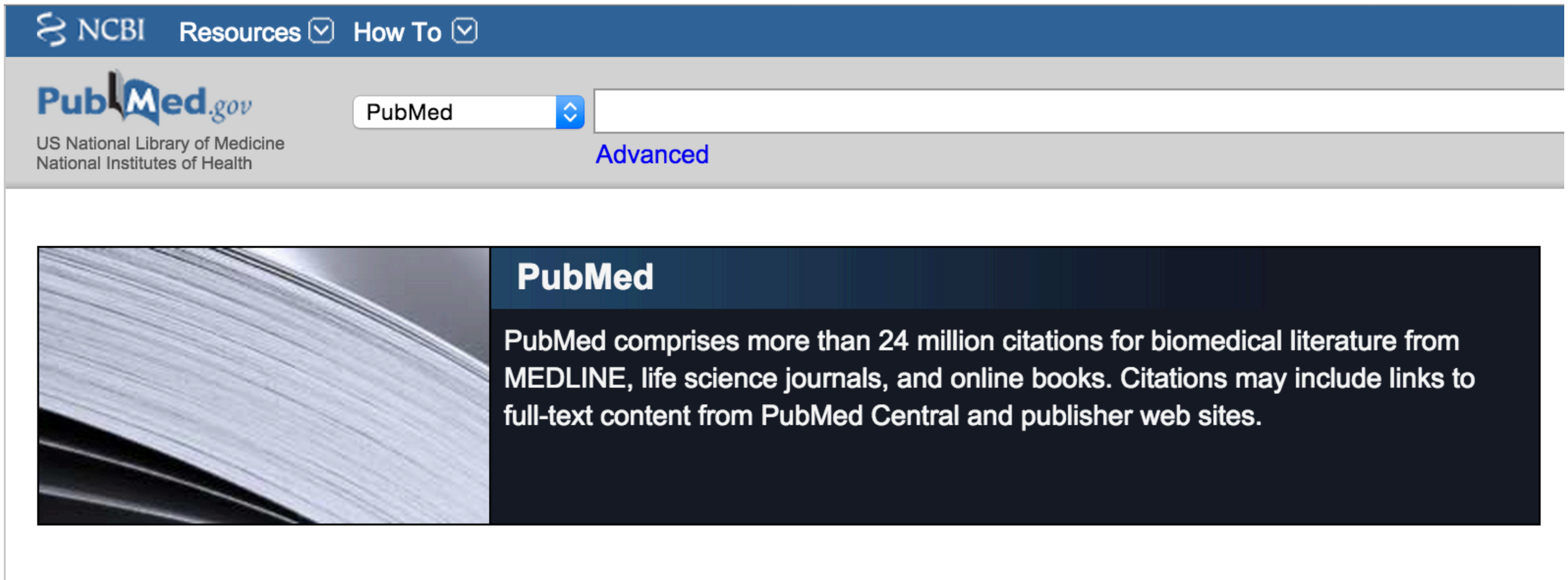
Fourier transform



"Fourier"	86,412

as of 2/1/2017

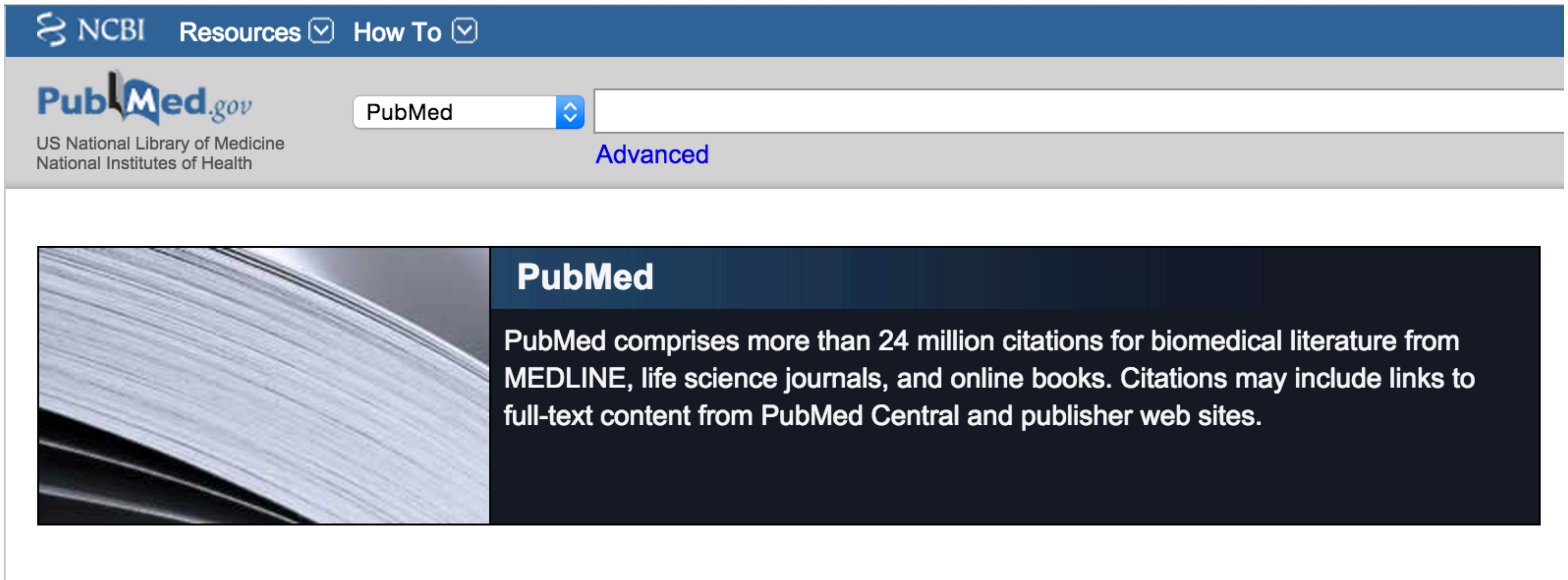
Fourier transform



“Fourier”	86,412
“Filter”	62,712

as of 2/1/2017

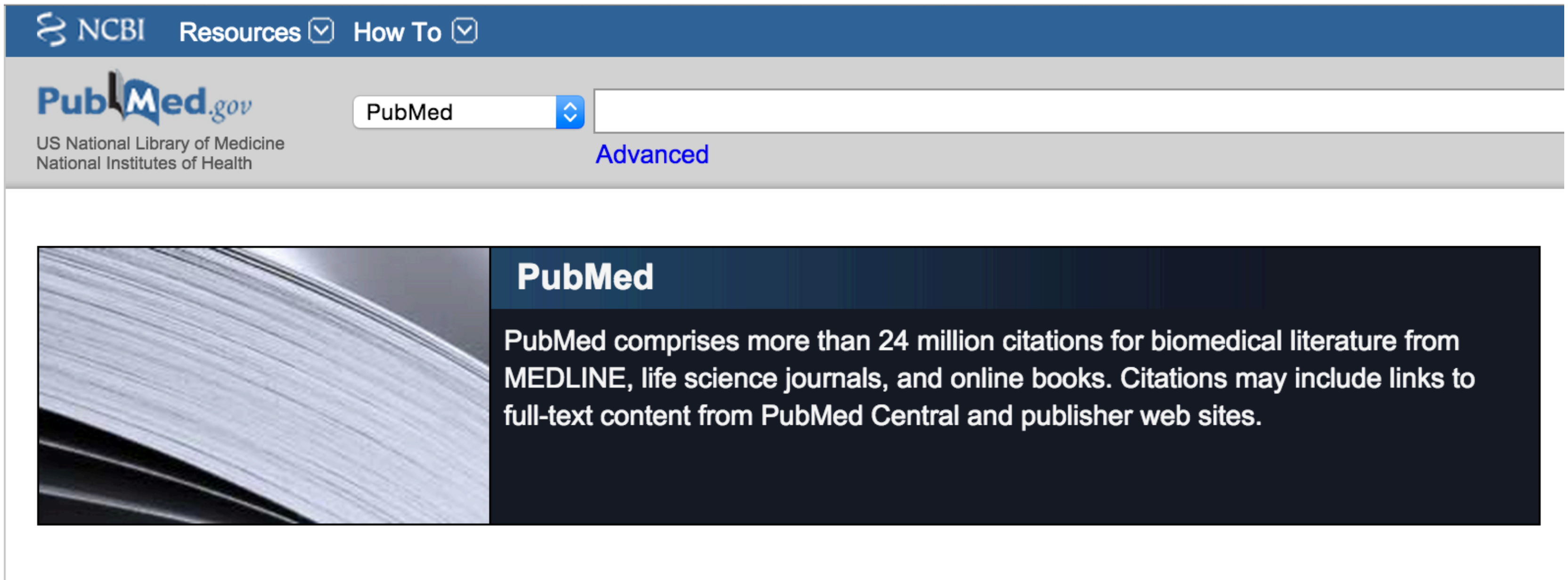
Fourier transform



"Fourier"	86,412
"Filter"	62,712
"Spectral analysis"	60,720

as of 2/1/2017

Fourier transform



“Fourier”	86,412	“Drosophila”	96,036
“Filter”	62,712	“Elegans *or* C. elegans *or* Caenorhabditis elegans”	29,251
“Spectral analysis”	60,720		

as of 2/1/2017

Other reasons Fourier transform is useful

Ease of computation

Filtering (convolution) is just element-wise multiplication in the frequency domain

Designing filters to remove noise is a lot easier in the frequency domain

Describing things

The cochlea transforms time domain signal into a frequency signal

Many brain regions have oscillations of a particular frequency

Fourier series

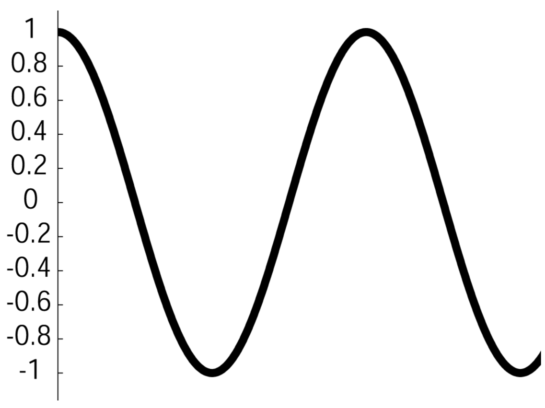
Every signal is equal to a sum of sines and cosines

Fourier series

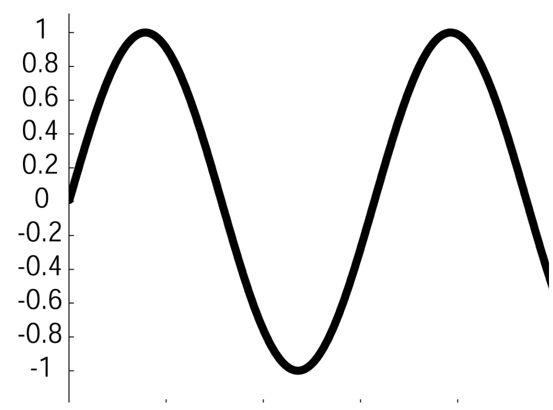
Every signal is equal to a sum of sines and cosines

$$a\cos(2\pi f x) + b\sin(2\pi f x)$$

cosine



sine

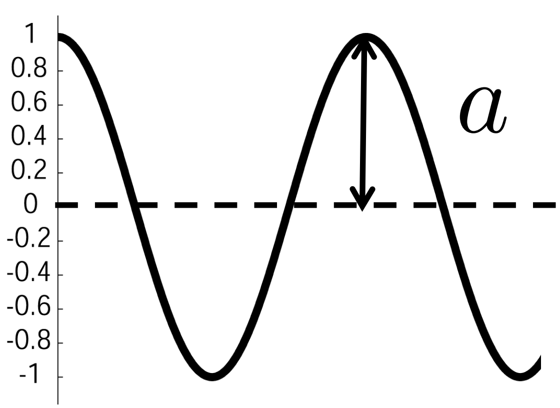


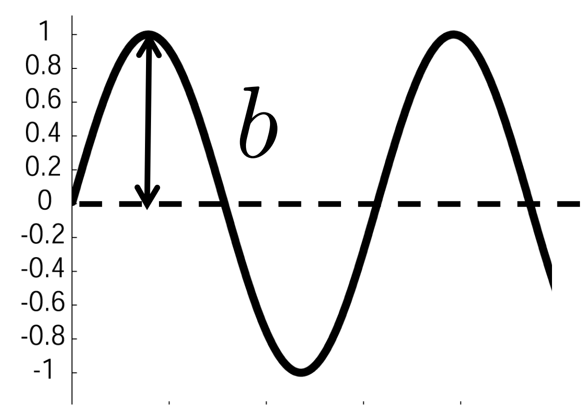
Fourier series

Every signal is equal to a sum of sines and cosines

$$a \cos(2\pi f x) + b \sin(2\pi f x)$$

coefficients (amplitudes)





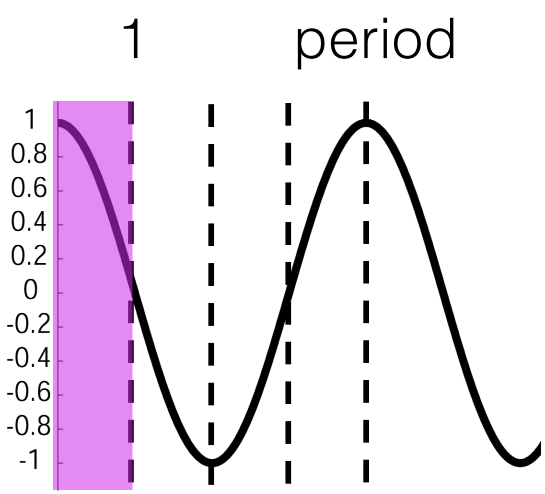
Fourier series

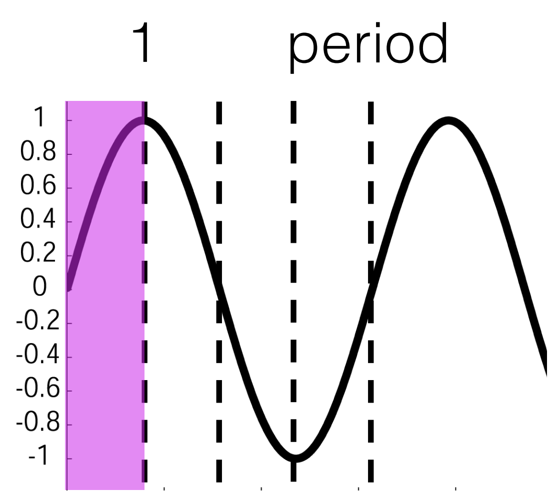
Every signal is equal to a sum of sines and cosines

$$\boxed{a}\cos(2\pi \boxed{f} x) + \boxed{b}\sin(2\pi \boxed{f} x)$$

coefficients (amplitudes)

frequencies





Fourier series

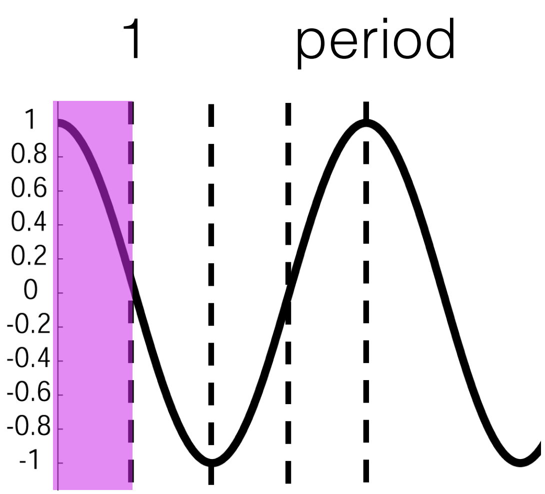
Every signal is equal to a sum of sines and cosines

$$a\cos(2\pi f x) + b\sin(2\pi f x)$$

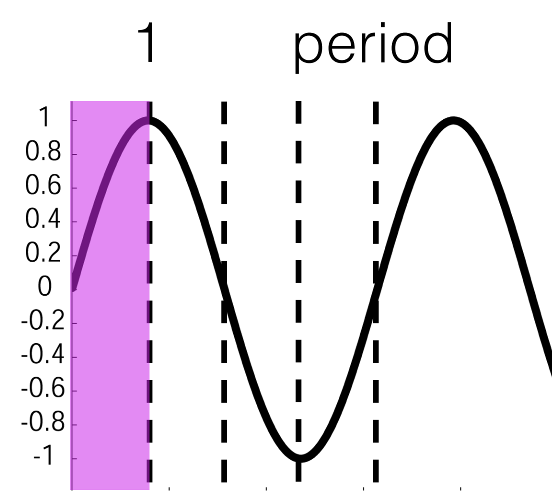
coefficients (amplitudes)

frequencies

$f = 1/4$

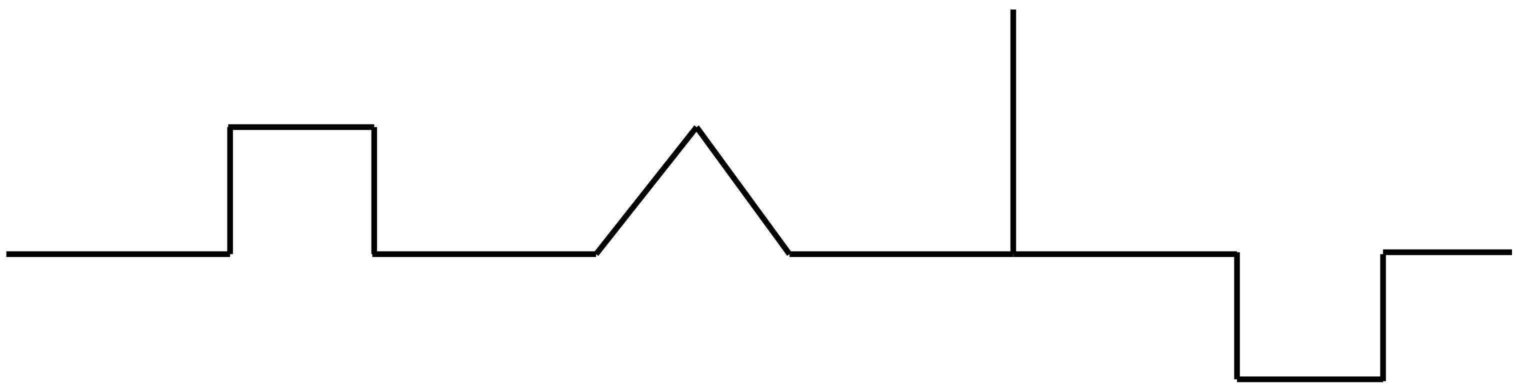


$f = 1/4$

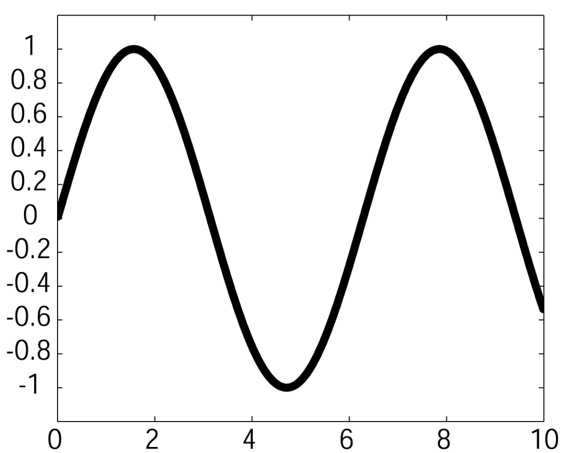


Fourier series

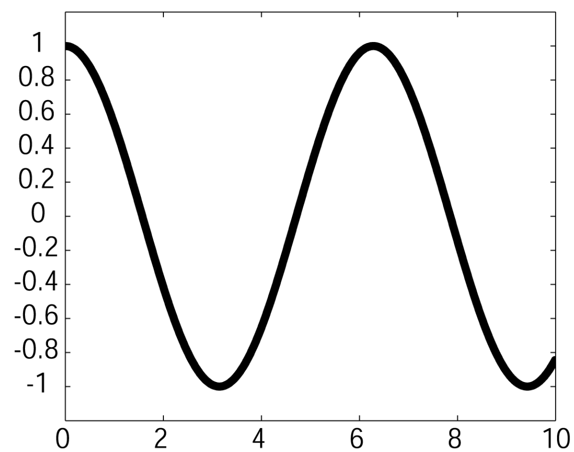
Every signal is equal to a sum of sines and cosines



sine (amplitude 1, period 2pi)



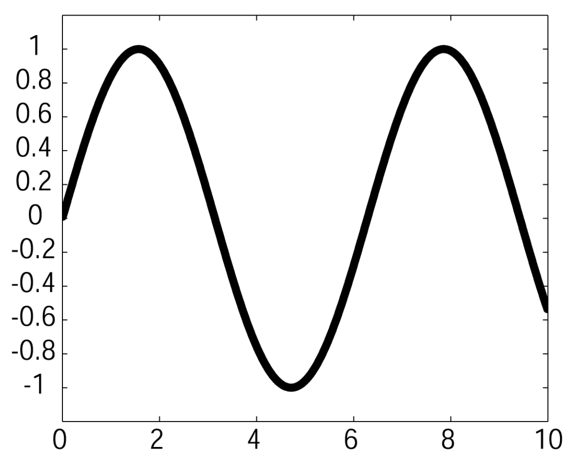
cosine (amplitude 1, period 2pi)



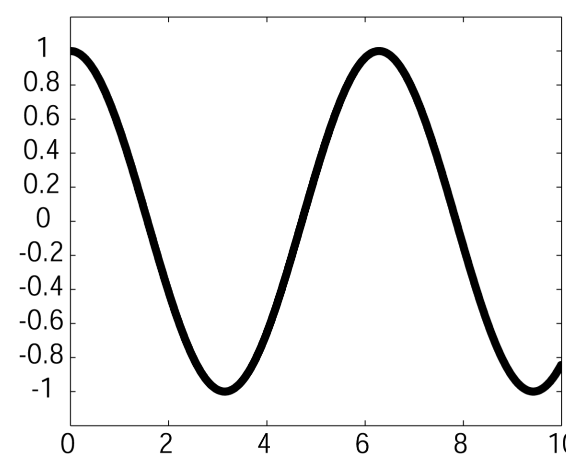
Fourier series

Every signal is equal to a sum of sines and cosines

sine (amplitude 1, period 2pi)

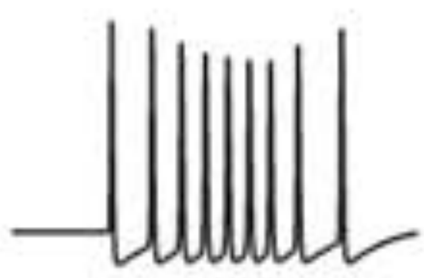


cosine (amplitude 1, period 2pi)

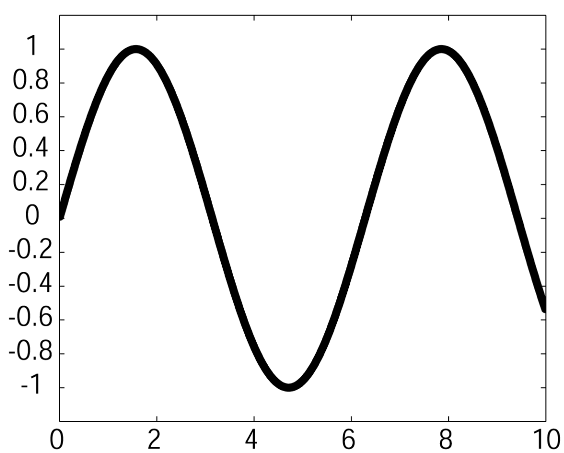


Fourier series

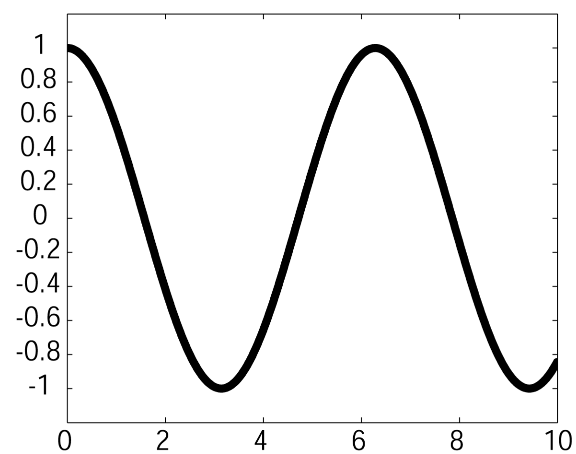
Every signal is equal to a sum of sines and cosines



sine (amplitude 1, period 2pi)

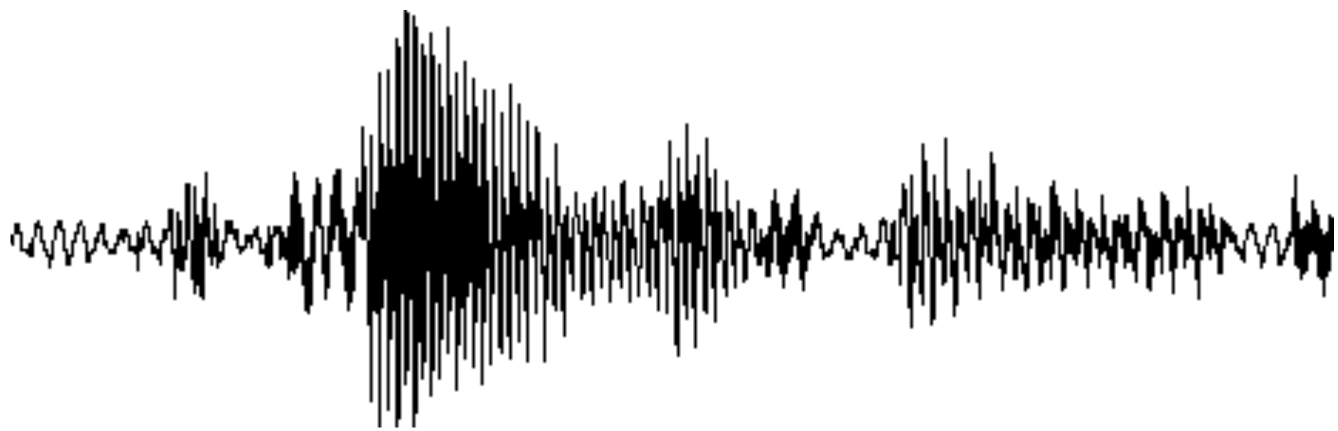


cosine (amplitude 1, period 2pi)

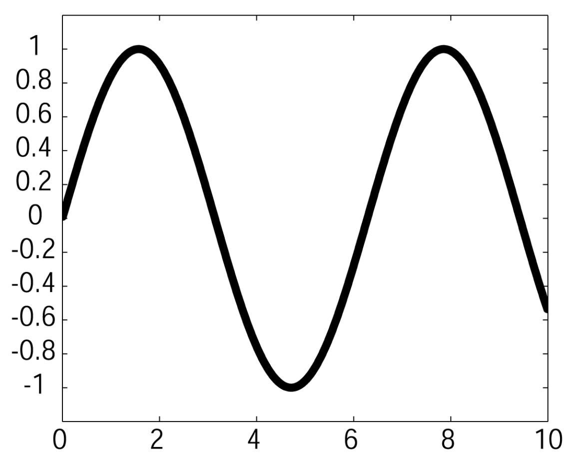


Fourier series

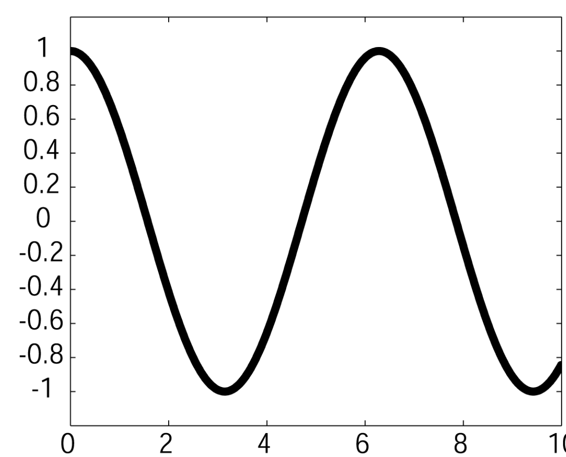
Every signal is equal to a sum of sines and cosines



sine (amplitude 1, period 2pi)



cosine (amplitude 1, period 2pi)



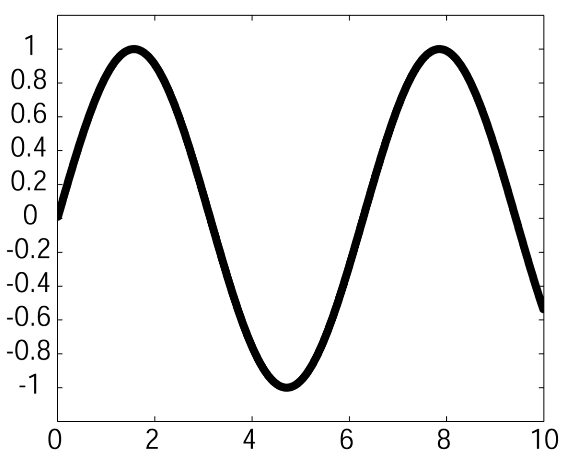
How is this possible?

Let's find the "Fourier series" representation of a simple signal

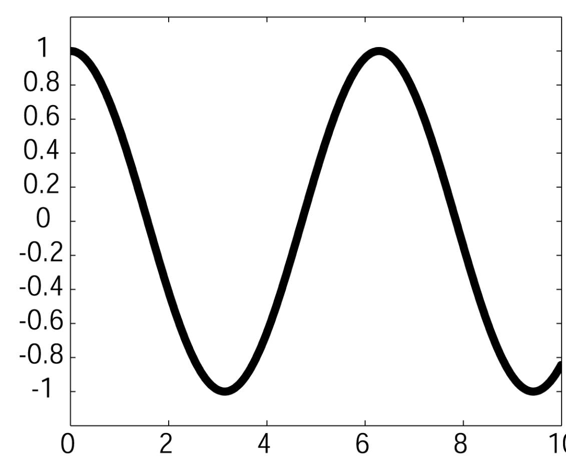
2

-1 ●

sine (amplitude 1, period 2pi)

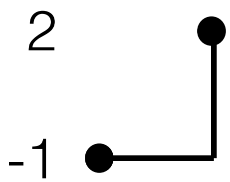


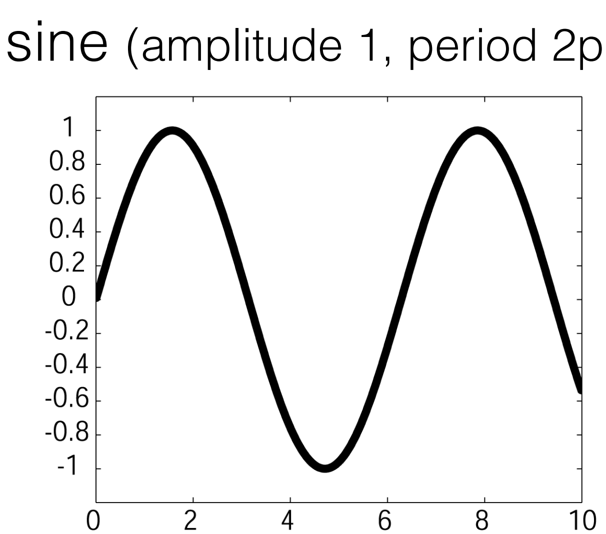
cosine (amplitude 1, period 2pi)

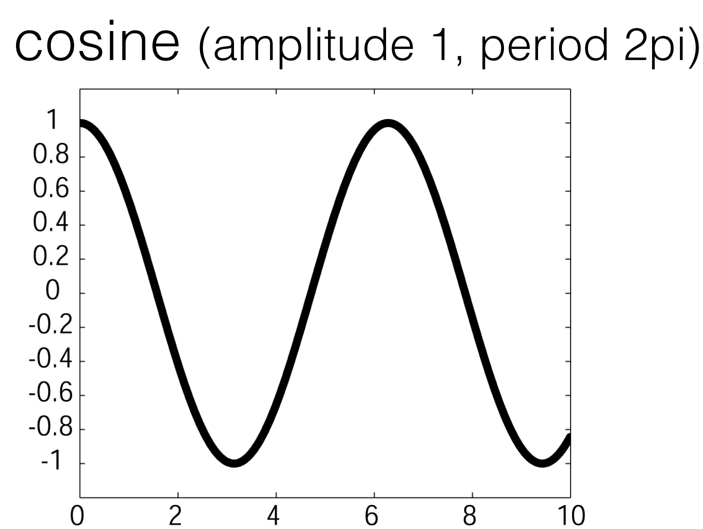


How is this possible?

Let's find the "Fourier series" representation of a simple signal

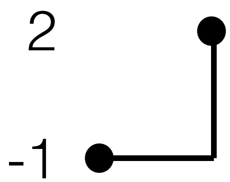


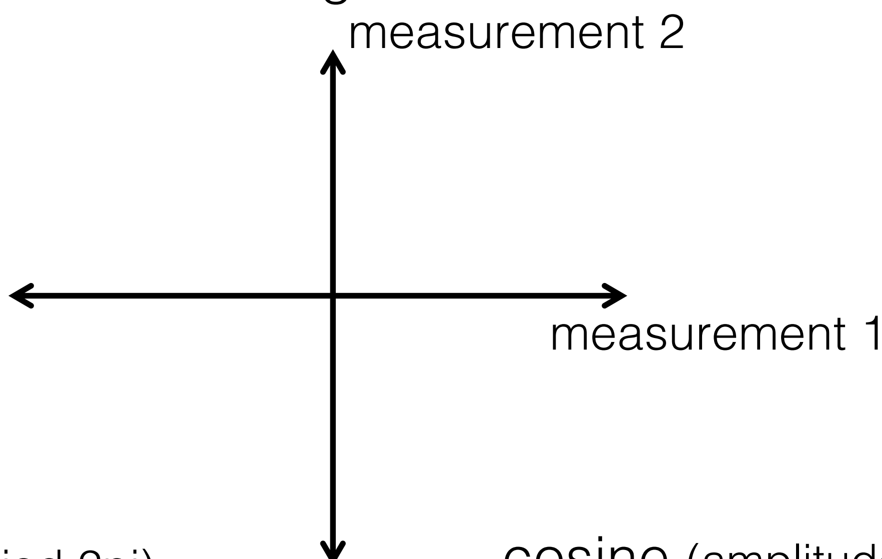




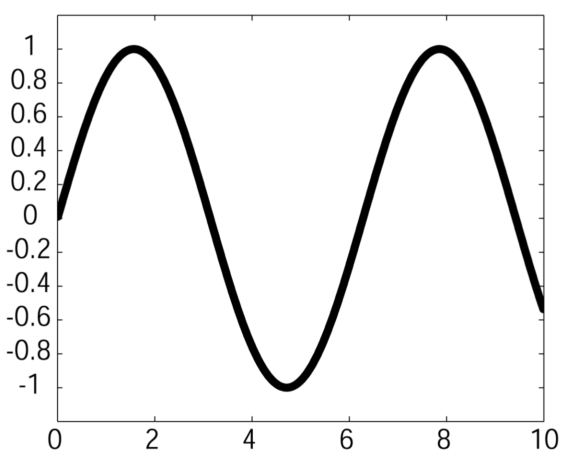
How is this possible?

Let’s find the “Fourier series” representation of a simple signal

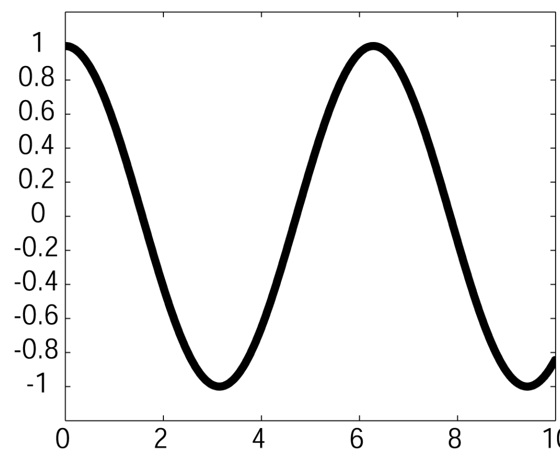




sine (amplitude 1, period 2pi)

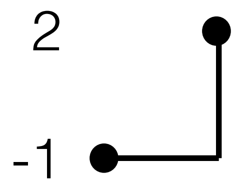


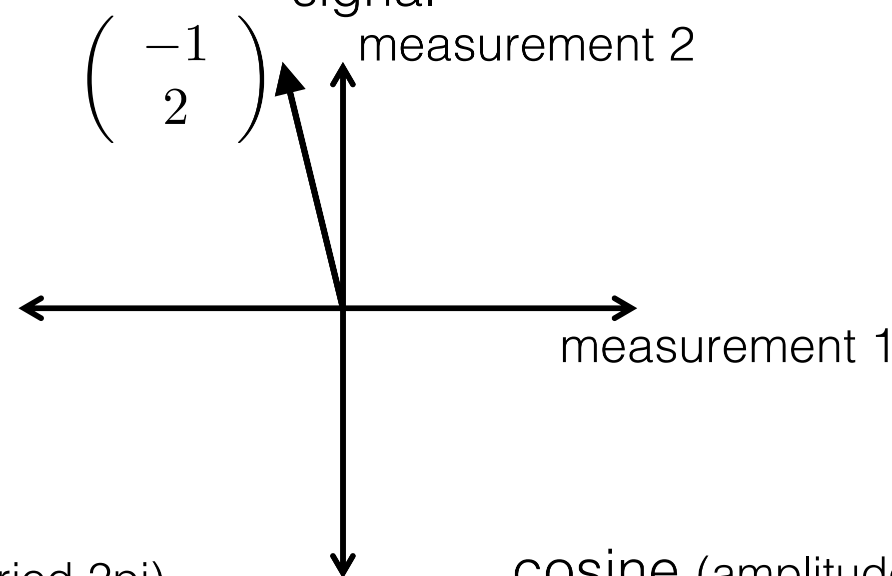
cosine (amplitude 1, period 2pi)



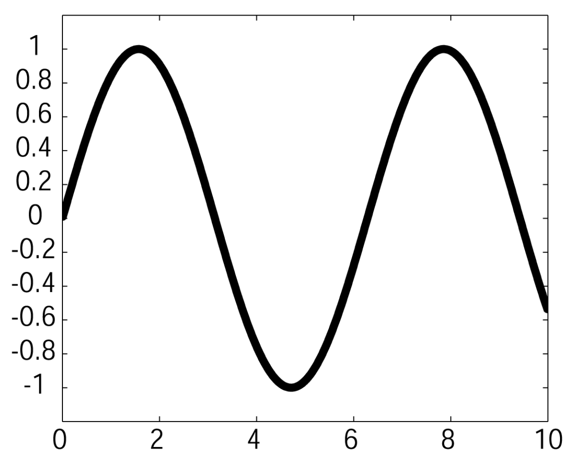
How is this possible?

Let’s find the “Fourier series” representation of a simple signal

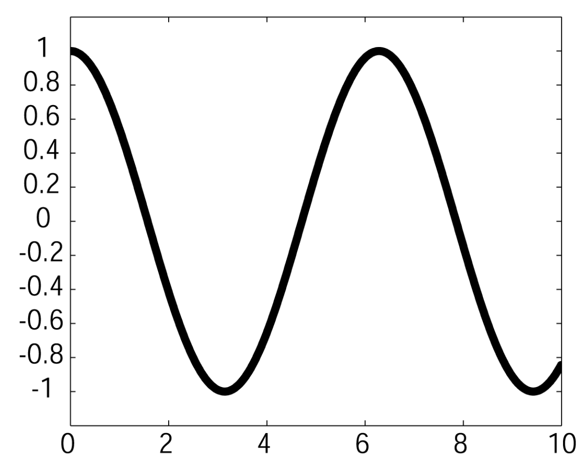




sine (amplitude 1, period 2pi)

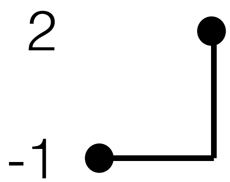


cosine (amplitude 1, period 2pi)

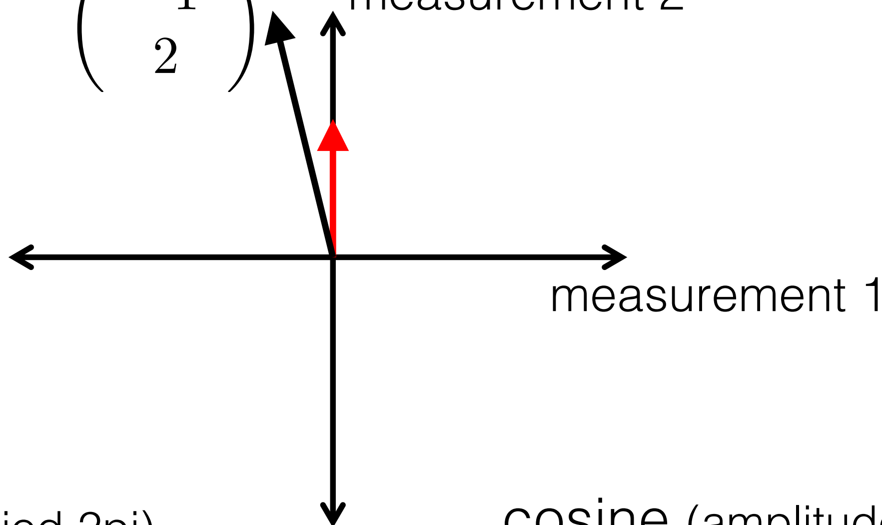


How is this possible?

Let's find the "Fourier series" representation of a simple signal

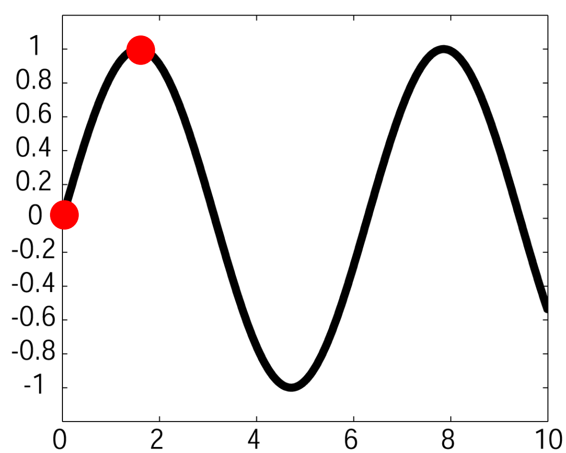


$$\begin{pmatrix} -1 \\ 2 \end{pmatrix}$$

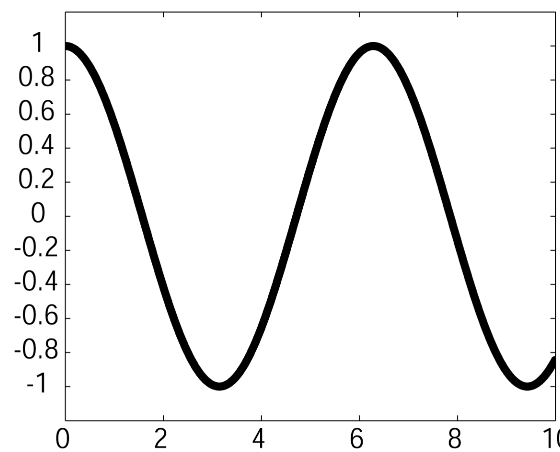


sine (amplitude 1, period 2pi)

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

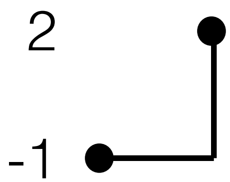


cosine (amplitude 1, period 2pi)

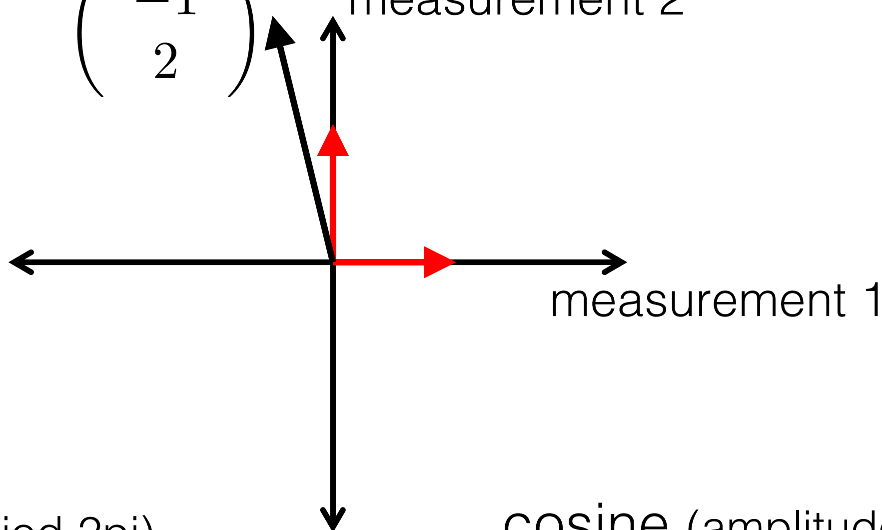


How is this possible?

Let's find the "Fourier series" representation of a simple signal

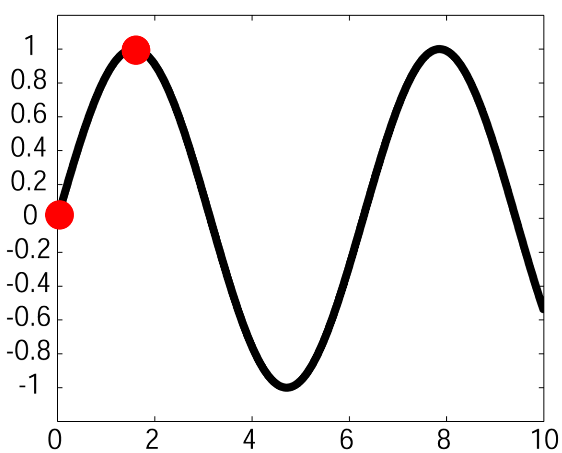


$$\begin{pmatrix} -1 \\ 2 \end{pmatrix}$$



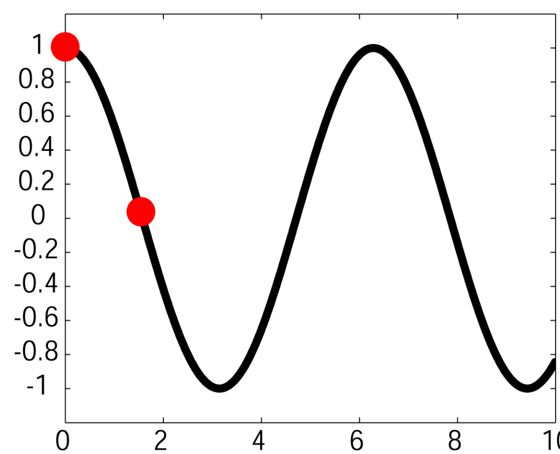
sine (amplitude 1, period 2pi)

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$



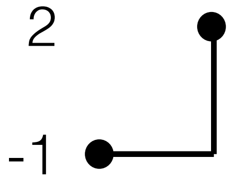
cosine (amplitude 1, period 2pi)

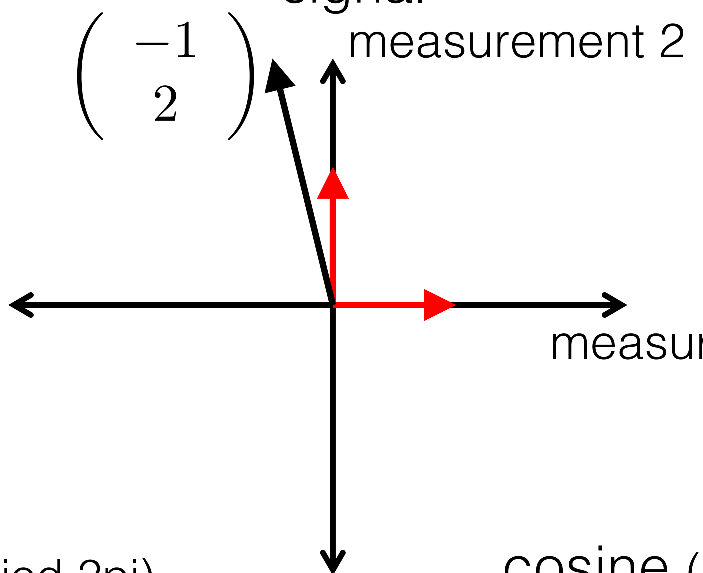
$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$



How is this possible?

Let’s find the “Fourier series” representation of a simple signal

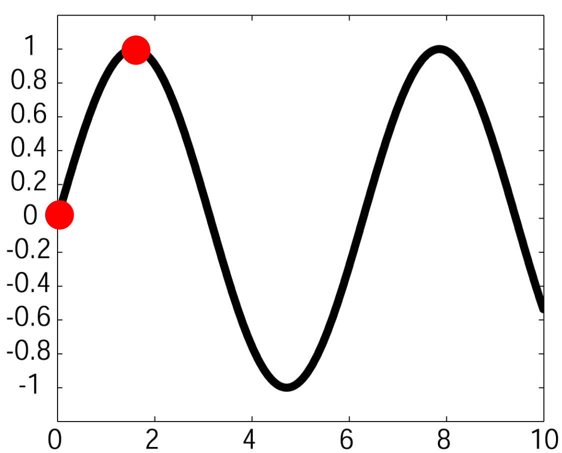




$$2\sin(x) - \cos(x)$$

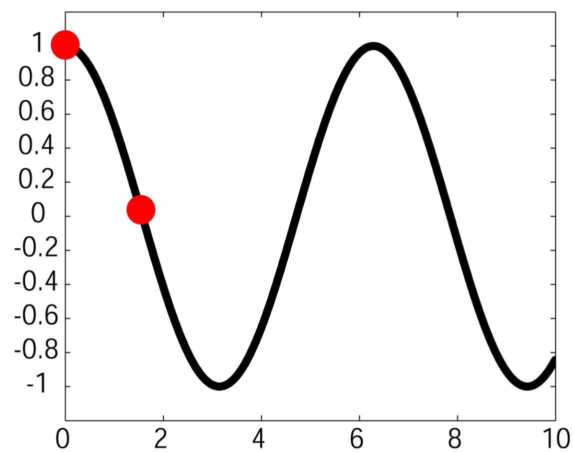
sine (amplitude 1, period 2pi)

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$



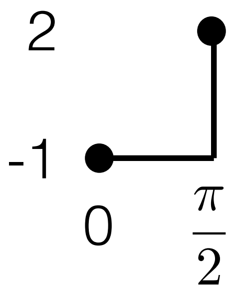
cosine (amplitude 1, period 2pi)

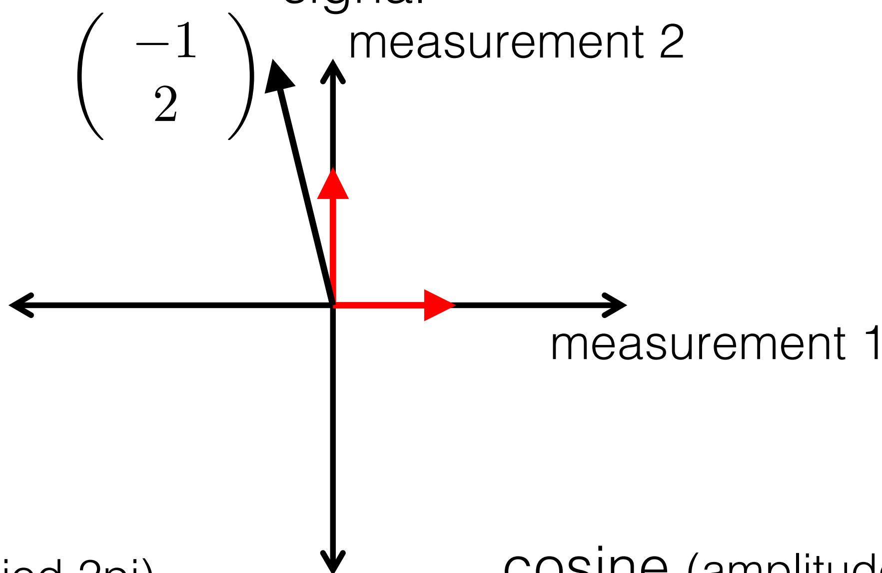
$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$



How is this possible?

Let's find the "Fourier series" representation of a simple signal

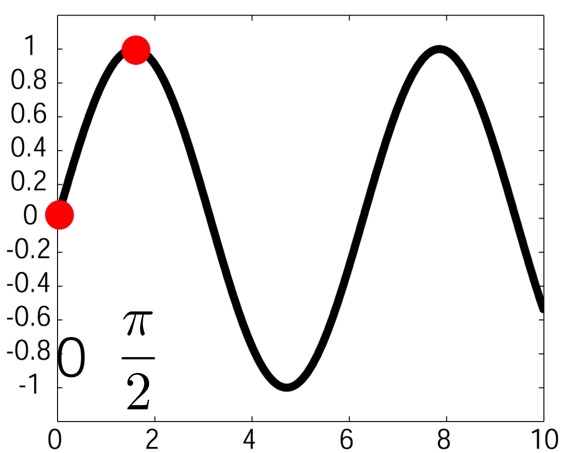




$$2\sin(x) - \cos(x)$$

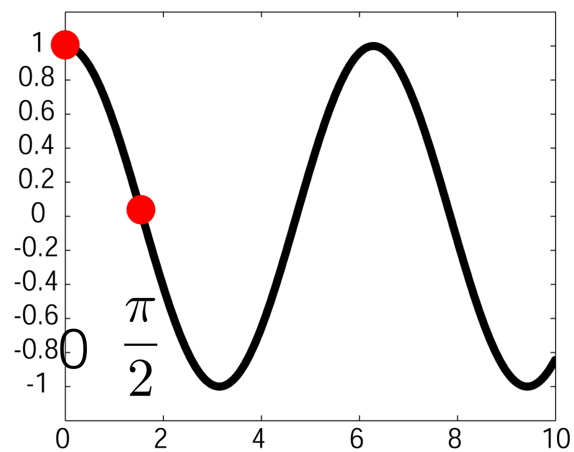
sine (amplitude 1, period 2pi)

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$



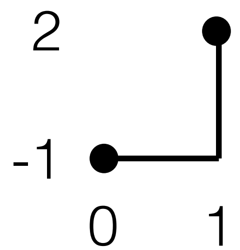
cosine (amplitude 1, period 2pi)

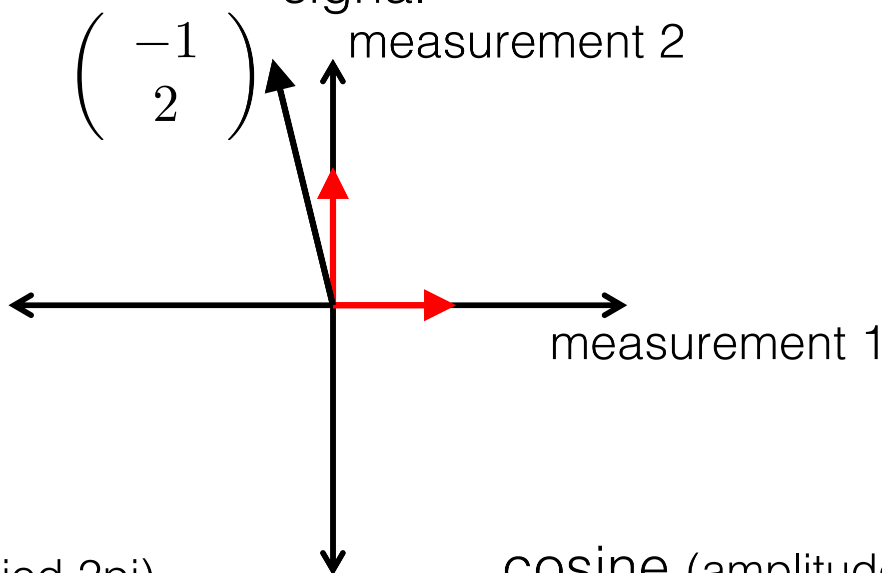
$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$



How is this possible?

Let's find the "Fourier series" representation of a simple signal

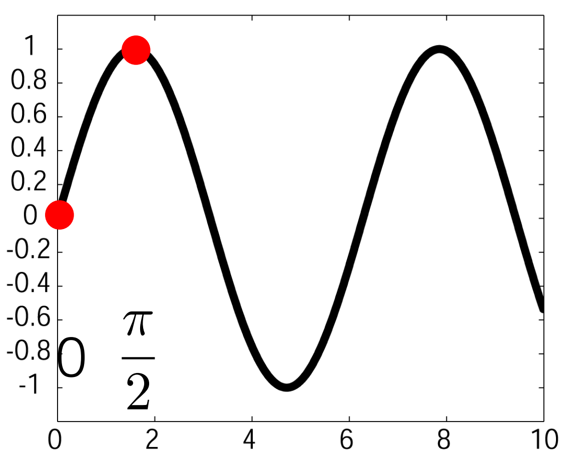




$$2\sin(x) - \cos(x)$$

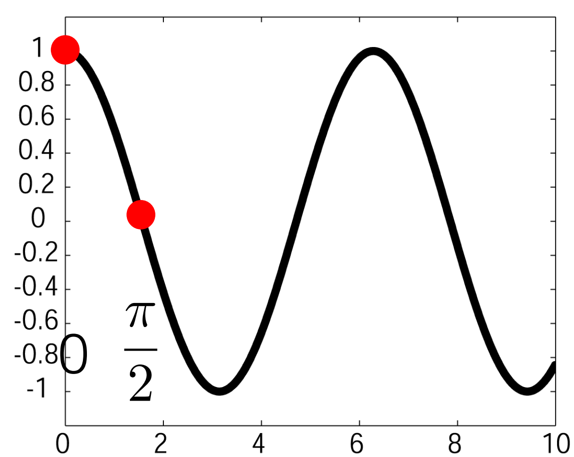
sine (amplitude 1, period 2pi)

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$



cosine (amplitude 1, period 2pi)

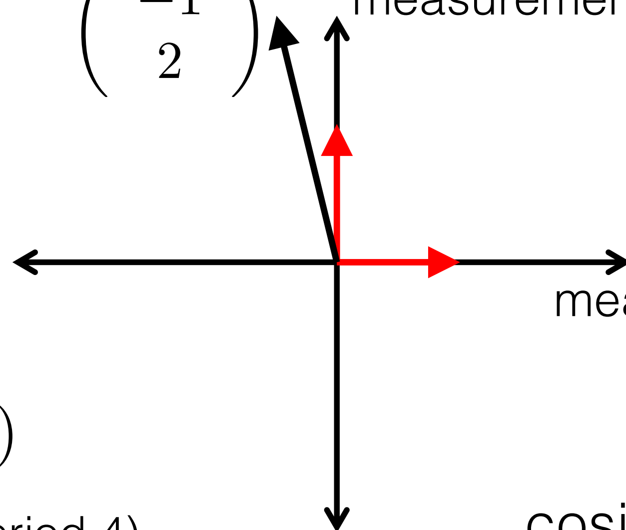
$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

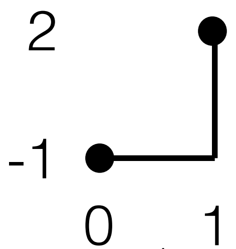


How is this possible?

Let's find the "Fourier series" representation of a simple signal

$$\begin{pmatrix} -1 \\ 2 \end{pmatrix}$$

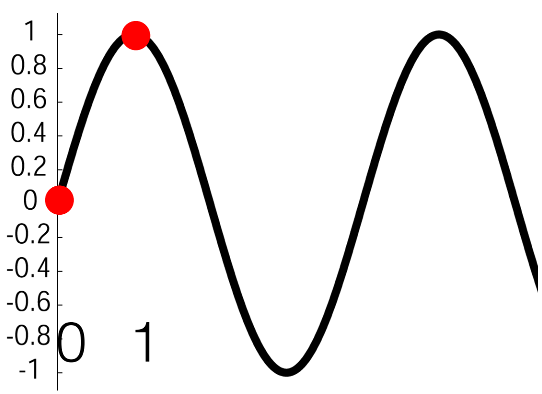




$$2\sin(2\pi\frac{1}{4}x) - \cos(2\pi\frac{1}{4}x)$$

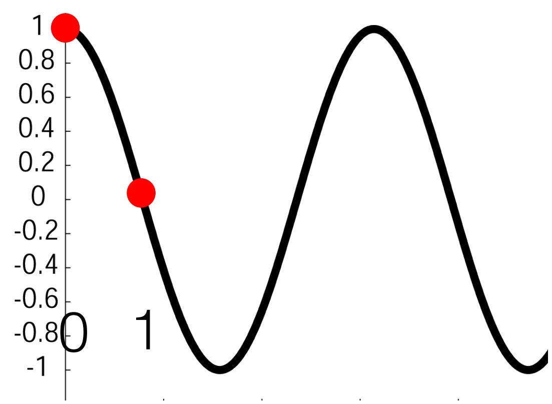
sine (amplitude 1, period 4)

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$



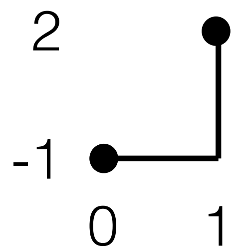
cosine (amplitude 1, period 4)

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

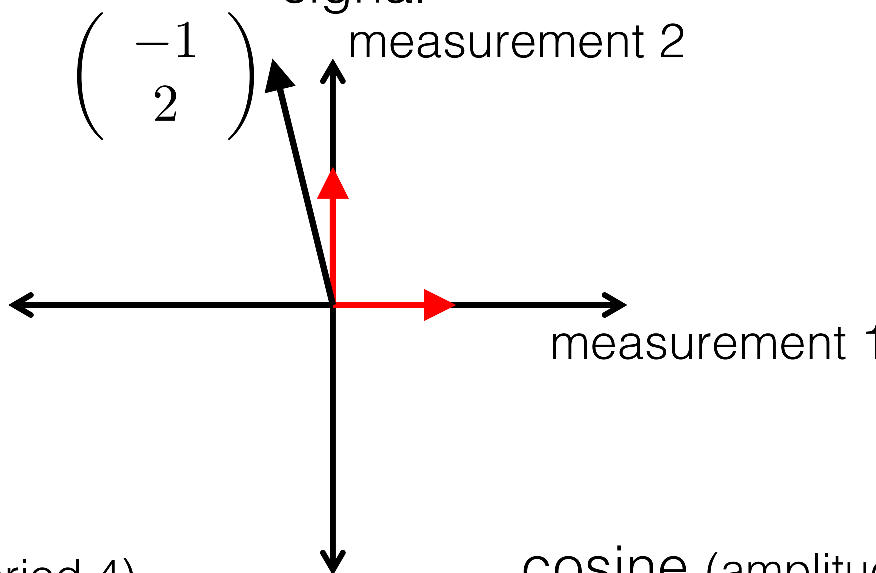


How is this possible?

Let's find the "Fourier series" representation of a simple signal

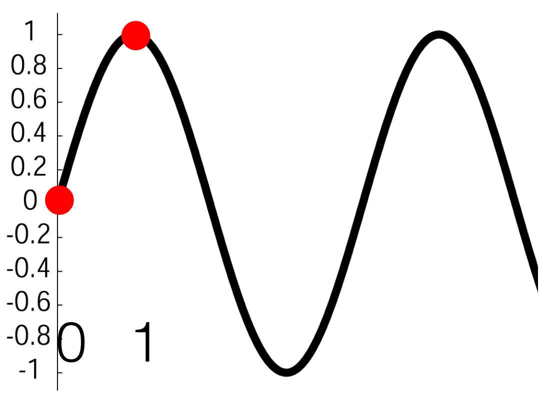


$$2\sin(\frac{\pi}{2}x) - \cos(\frac{\pi}{2}x)$$



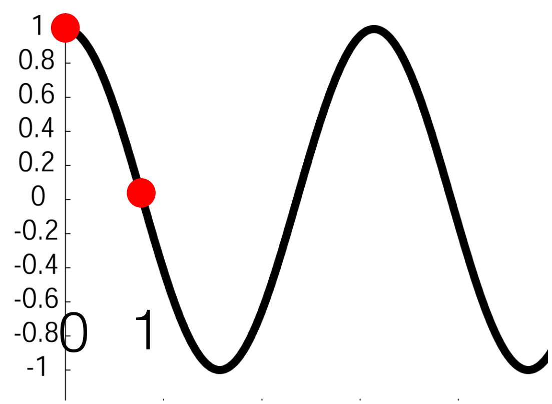
sine (amplitude 1, period 4)

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$



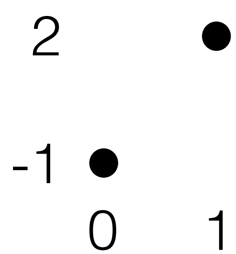
cosine (amplitude 1, period 4)

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

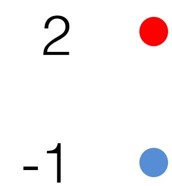


Moving to frequency space

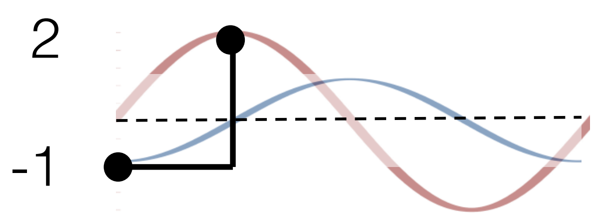
signal measurements
vs
time



sinusoid coefficients
vs
frequency



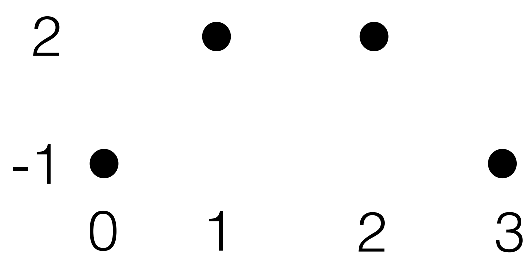
$\frac{1}{4}$

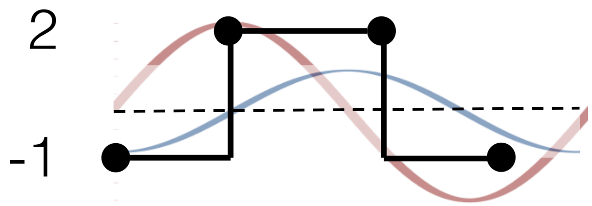


$$-1\cos(\frac{\pi}{2}x) + 2\sin(\frac{\pi}{2}x)$$

Moving to frequency space

signal measurements
vs
time

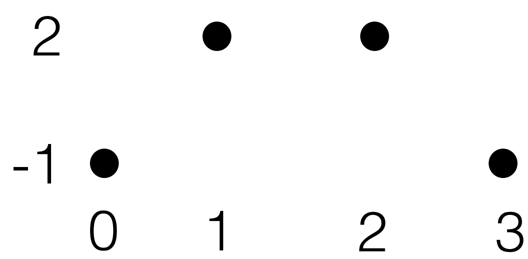




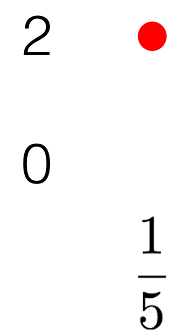
sinusoid coefficients
vs
frequency

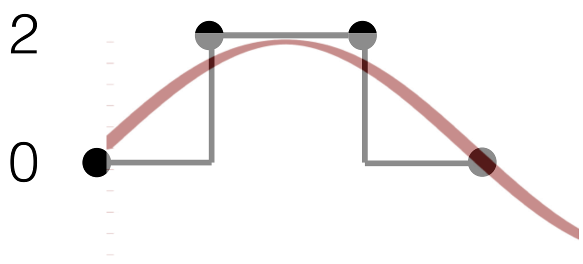
Moving to frequency space

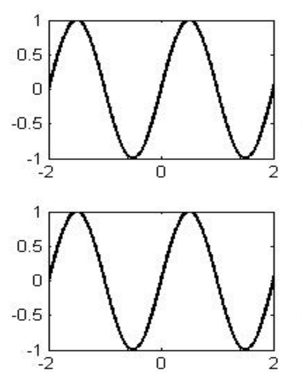
signal measurements
vs
time



sinusoid coefficients
vs
frequency

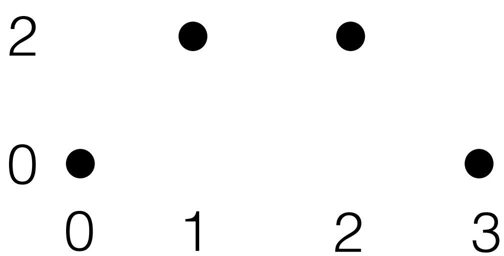




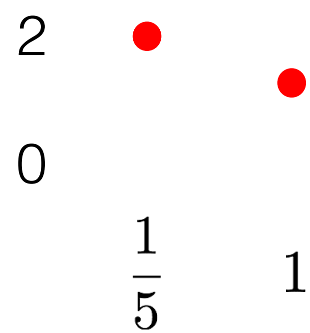


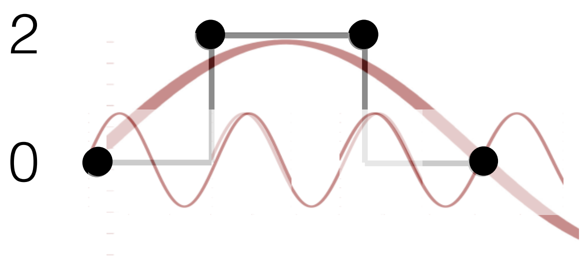
Moving to frequency space

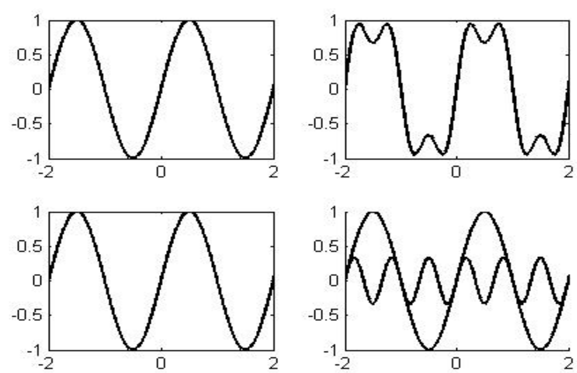
signal measurements
vs
time



sinusoid coefficients
vs
frequency

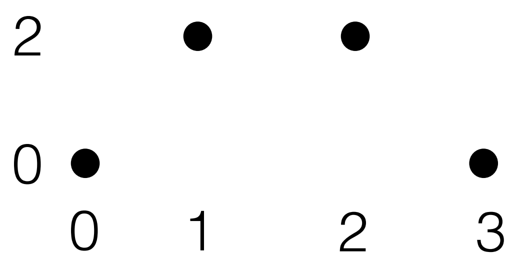




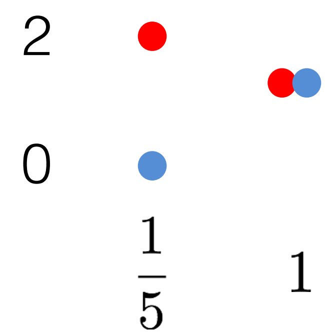


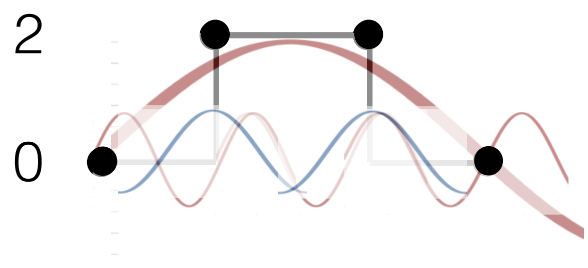
Moving to frequency space

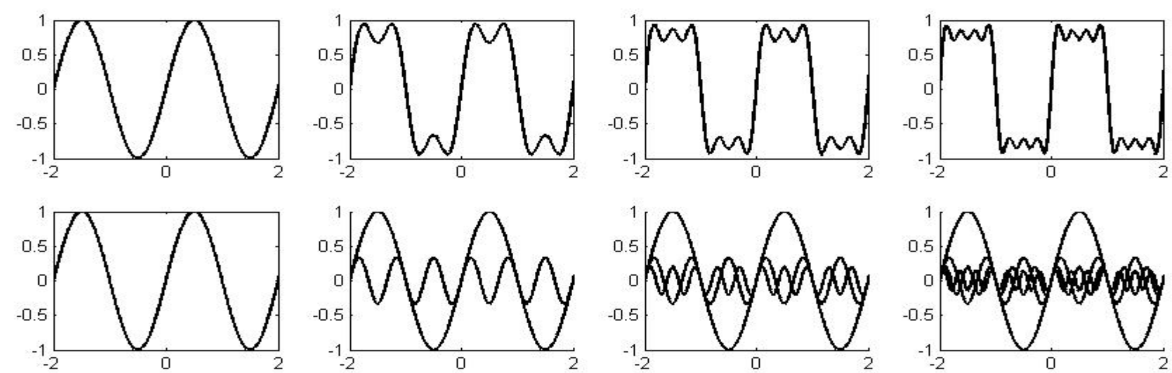
signal measurements
vs
time



sinusoid coefficients
vs
frequency

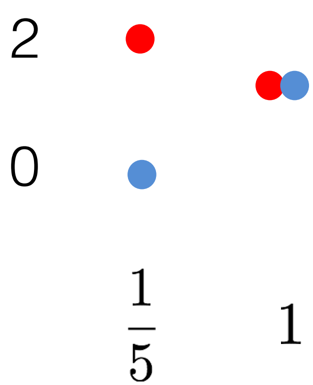






Fourier transform

Fourier series

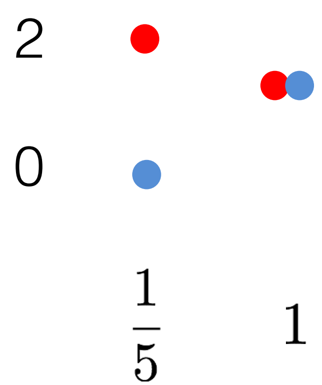


$$\color{blue}{0}\color{black}\cos(\frac{2\pi}{5}x) + \color{red}{2}\color{black}\sin(\frac{2\pi}{5}x)$$

$$\begin{pmatrix} 0 \\ 2 \end{pmatrix}$$

Fourier transform

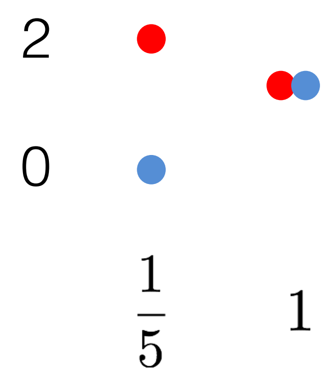
Fourier series



$$\color{blue}{0}\cos(\frac{2\pi}{5}x) + \color{red}{2}\sin(\frac{2\pi}{5}x)$$

$$\begin{pmatrix} 0 \\ 2 \end{pmatrix}$$

Fourier transform

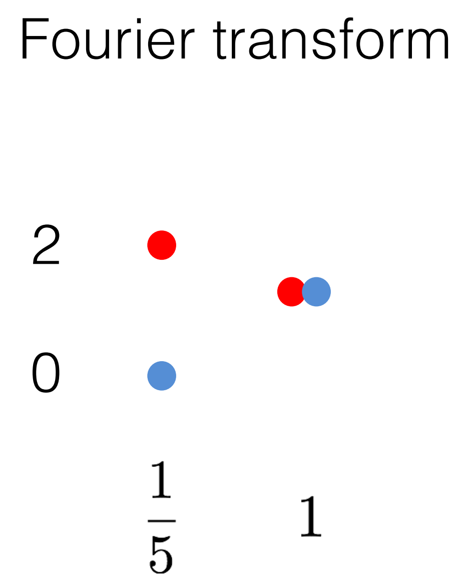


$$\color{blue}{0}\cos(\frac{2\pi}{5}x) - i\color{red}{2}\sin(\frac{2\pi}{5}x)$$

$$0 - 2i$$

Fourier transform

$$F[\text{signal}]_f = \sum_{t}^{T-1} \text{signal}(t) \cdot e^{-i2\pi f t}$$



$$0 \cos(\frac{2\pi}{5}x) - i2 \sin(\frac{2\pi}{5}x)$$

$$0 - 2i$$

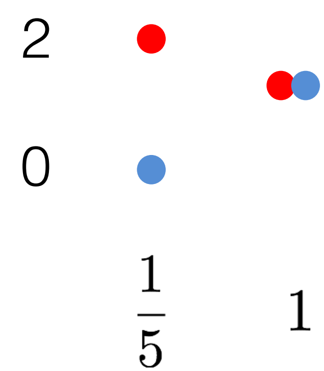
MATLAB: `fft(signal)`

Fourier transform

$$F[\text{signal}]_f = \sum_{t}^{T-1} \text{signal}(t) \cdot e^{-i2\pi f t}$$

$$e^{-i2\pi f t} = \cos(2\pi f t) - i \sin(2\pi f t)$$

Fourier transform



$$\color{blue}{0}\color{black}\cos(\frac{2\pi}{5}x) - i\color{red}{2}\color{black}\sin(\frac{2\pi}{5}x)$$

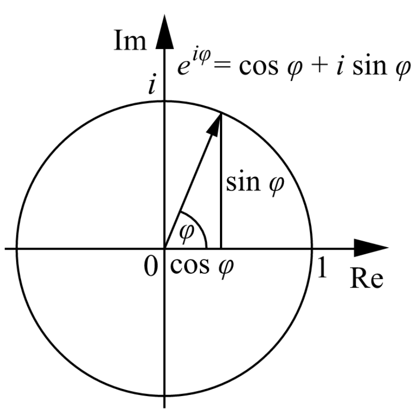
$$0 - 2i$$

MATLAB: `fft(signal)`

Fourier transform

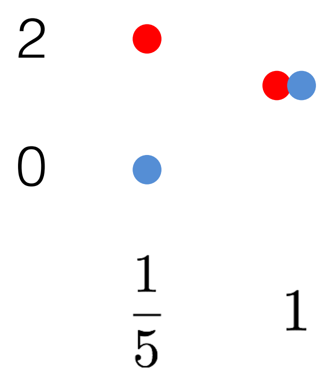
$$F[\text{signal}]_f = \sum_{t}^{T-1} \text{signal}(t) \cdot e^{-i2\pi f t}$$

$$e^{-i2\pi f t} = \cos(2\pi f t) - i\sin(2\pi f t)$$



MATLAB: `fft(signal)`

Fourier transform



$$0\cos(\frac{2\pi}{5}x) - i2\sin(\frac{2\pi}{5}x)$$

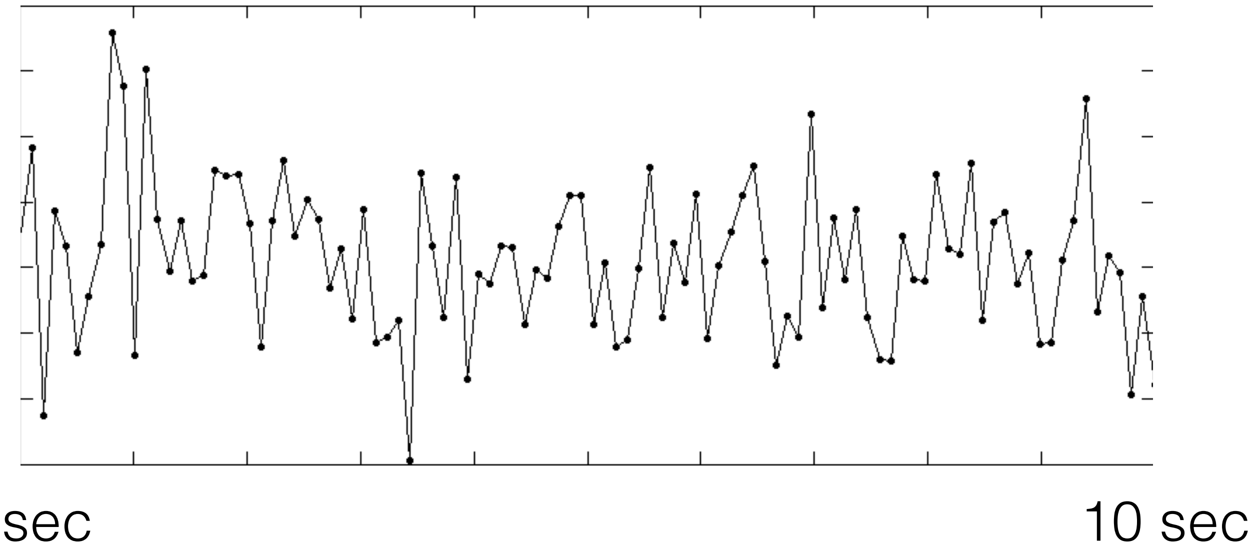
$$0 - 2i$$

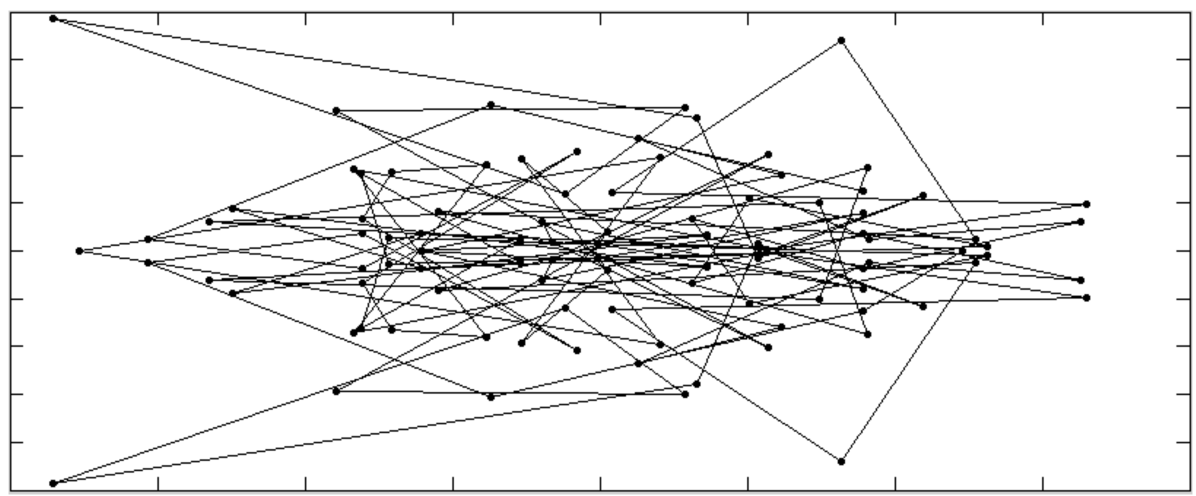
Fourier transform example

random white noise
sampled at 10 Hz

`randn(100,1)`

`linspace(0,10,100)`



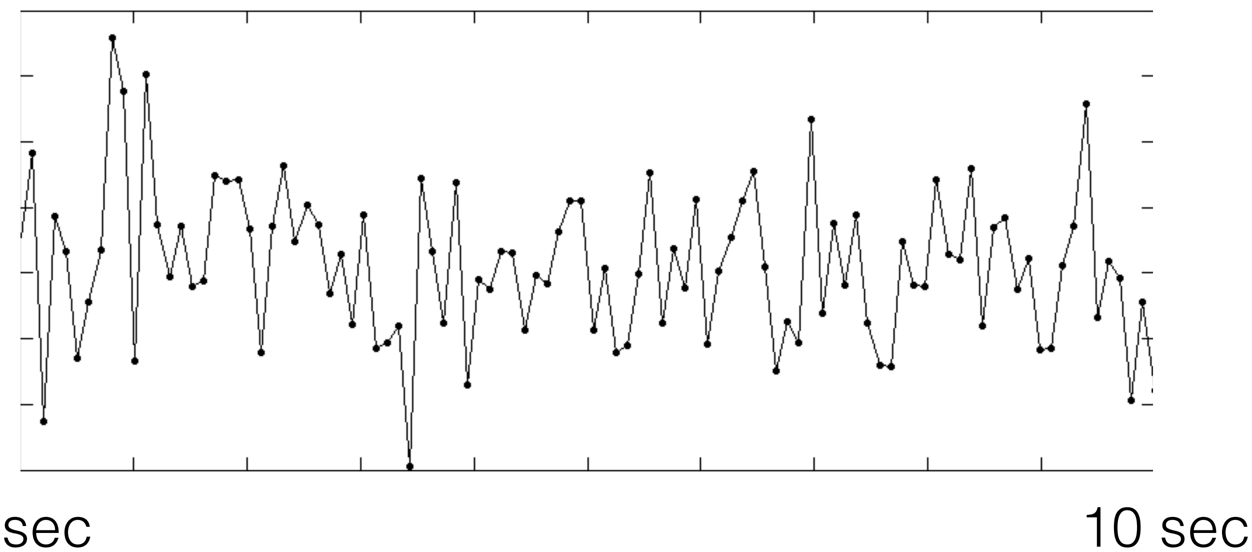


MATLAB: `fft(signal)`

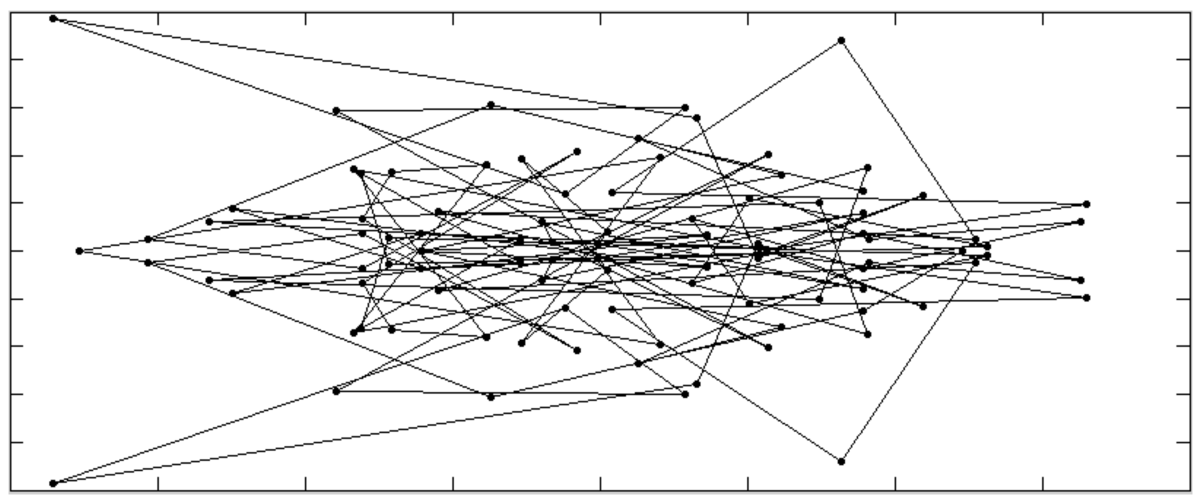
Fourier transform example

random white noise
sampled at 10 Hz

`randn(100,1)`
`linspace(0,10,100)`




```
F    = fft(signal)
F(1) = 12.3081
F(2) = 8.95 - 6.25i
F(3) = -3.71 - 15.2i
...
```

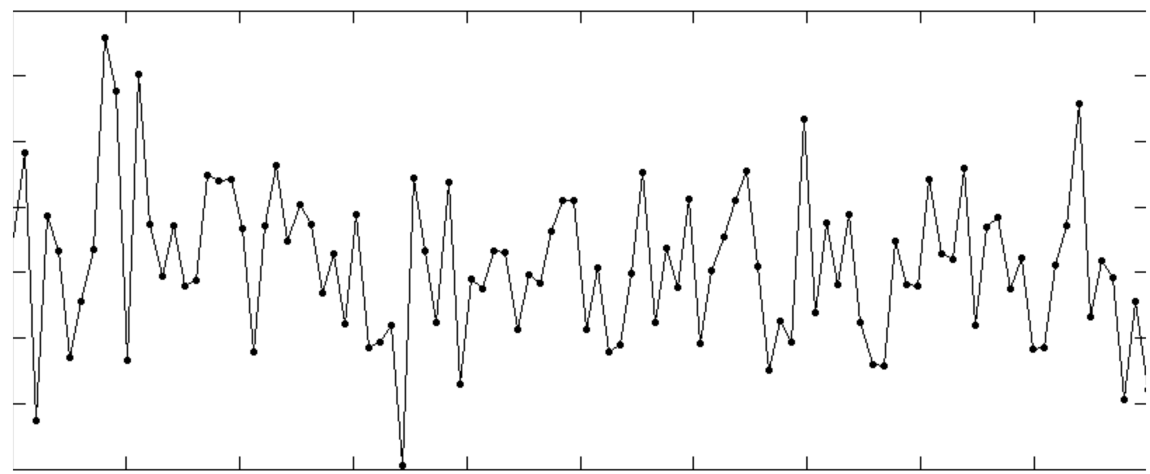


MATLAB: `fft(signal)`

Fourier transform example

random white noise
sampled at 10 Hz

`randn(100,1)`
`linspace(0,10,100)`




```
F    = fft(signal)
F(1) = 12.3081
F(2) = 8.95 - 6.25i
F(3) = -3.71 - 15.2i
...
```

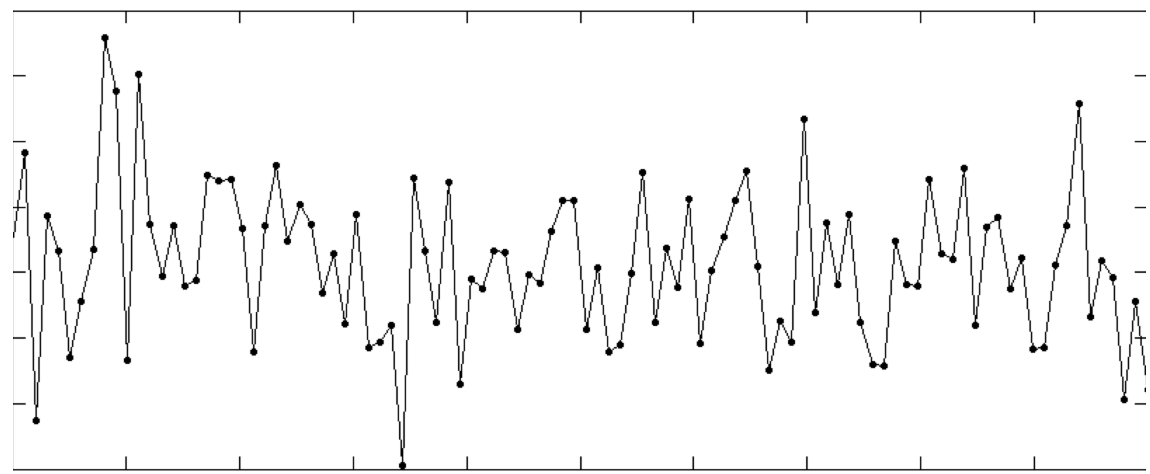
$$F[\text{signal}]_f = \sum_{t}^{T-1} \text{signal}(t) \cdot e^{-i 2\pi f t}$$

MATLAB: `fft(signal)`

Fourier transform example

random white noise
sampled at 10 Hz

```
randn(100,1)
linspace(0,10,100)
```




```
F    = fft(signal)
F(1) = 12.3081
F(2) = 8.95 - 6.25i
F(3) = -3.71 - 15.2i
...
```

$$F[\text{signal}]_f = \sum_{t}^{T-1} \text{signal}(t) \cdot e^{-i2\pi f t}$$

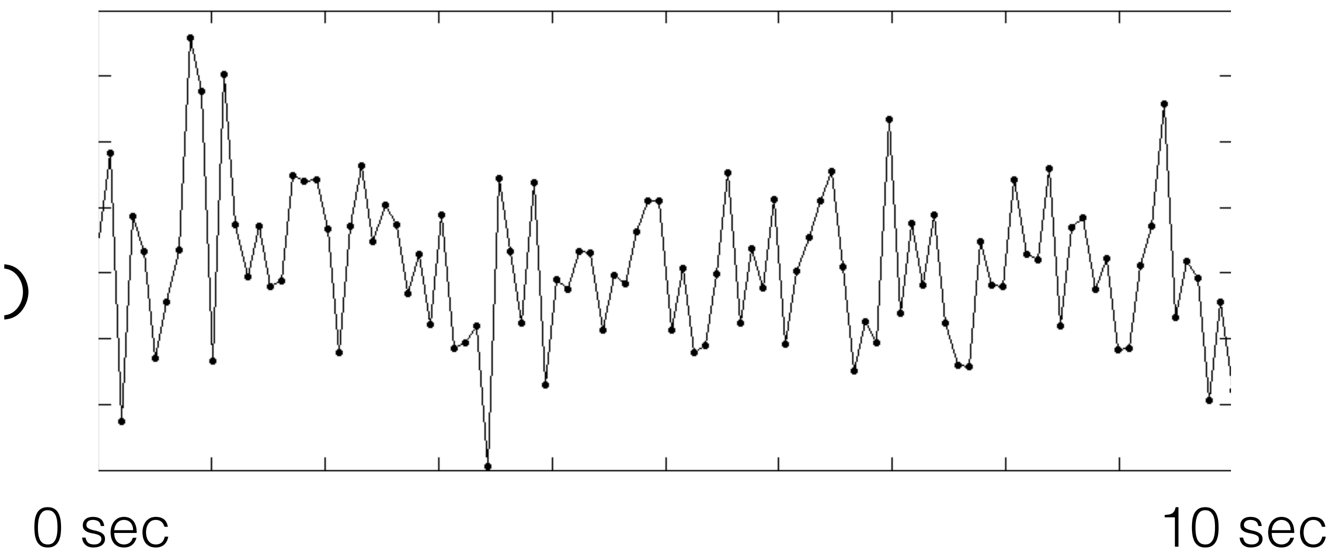
MATLAB: $\overbrace{\texttt{(1/length(signal))}}^{\text{typical normalization so F[0 Hz] = mean(signal)}}$ * `fft(signal)`

Amplitude spectrum example

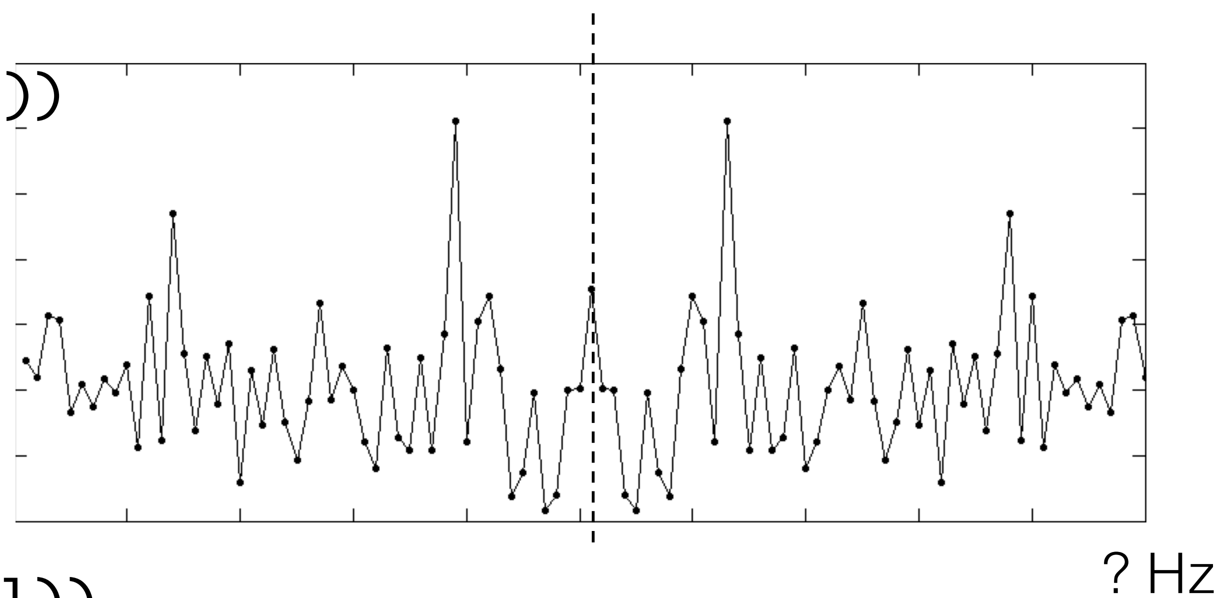
random white noise
sampled at 10 Hz

`randn(100,1)`

`linspace(0,10,100)`




```
F    = abs(fft(signal))
F(1) = 12.3081
F(2) = 10.9257
F(3) = 15.6638
...
```

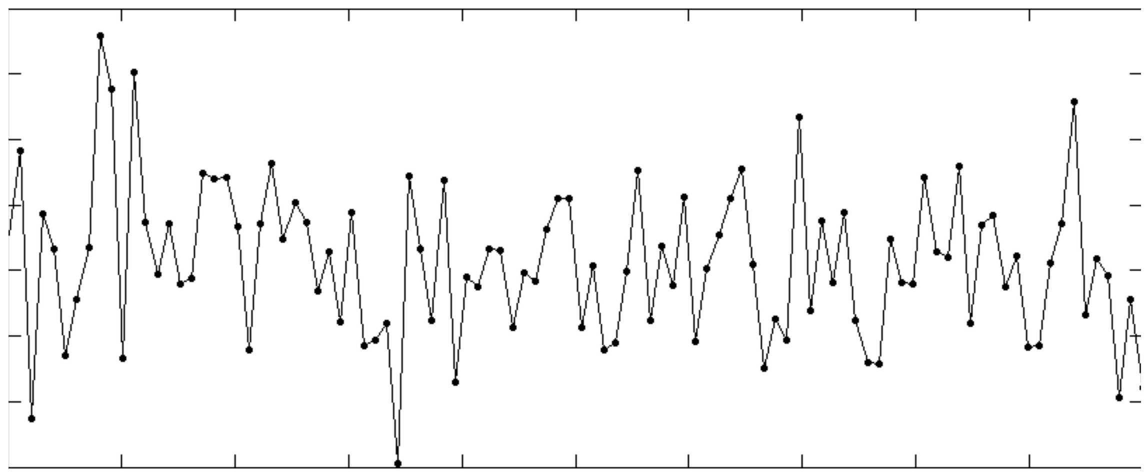


MATLAB: `abs(fft(signal))`

Amplitude spectrum example

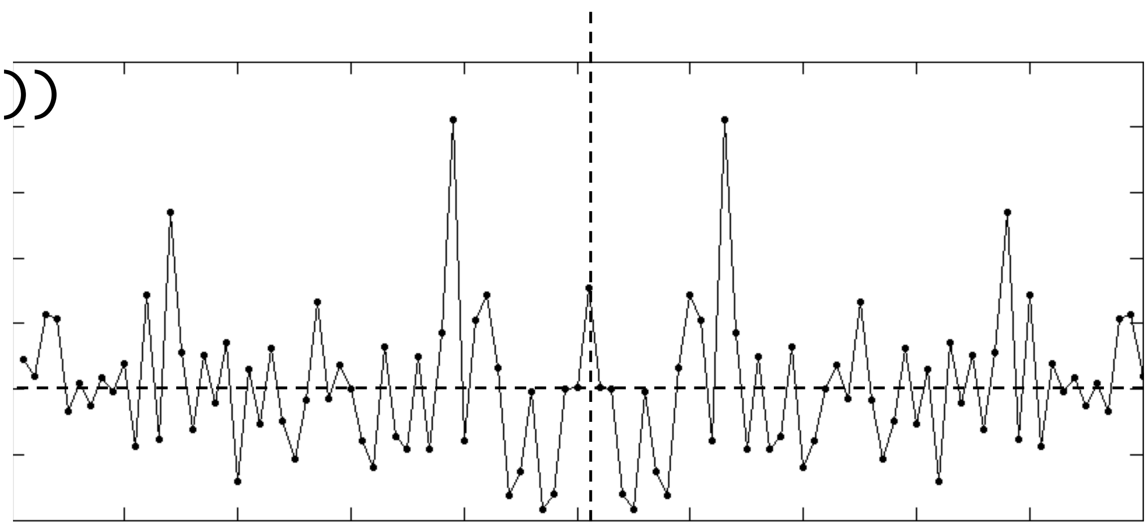
random white noise
sampled at 10 Hz

`randn(100,1)`
`linspace(0,10,100)`



0 sec 10 sec

```
F    = abs(fft(signal))
F(1) = 12.3081
F(2) = 10.9257
F(3) = 15.6638
...
```



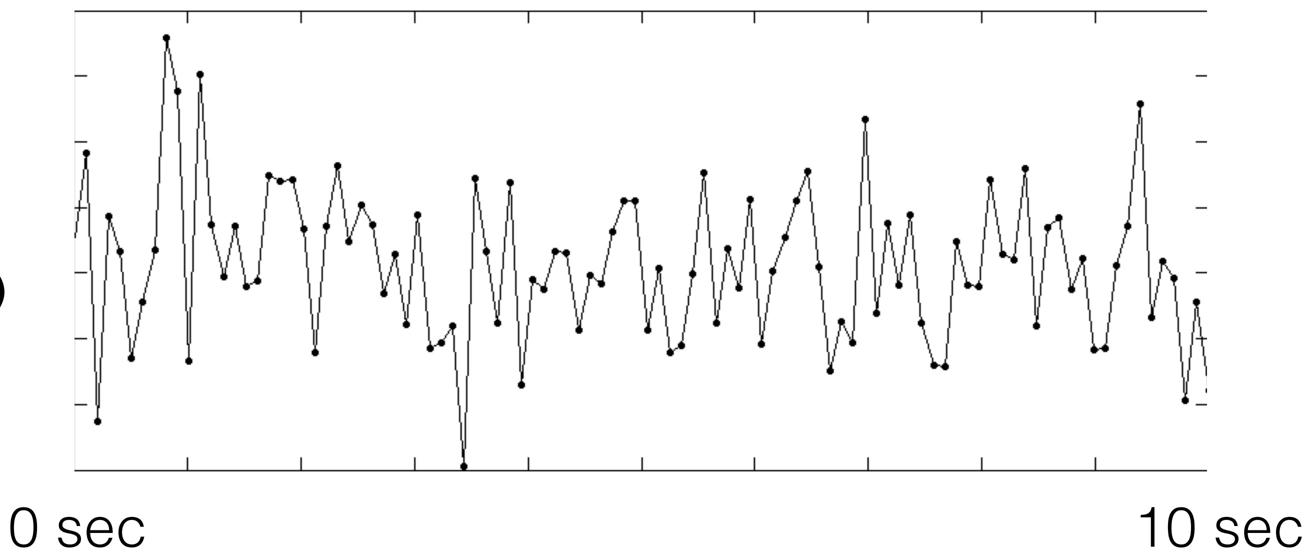
? Hz

MATLAB: `abs(fft(signal))`

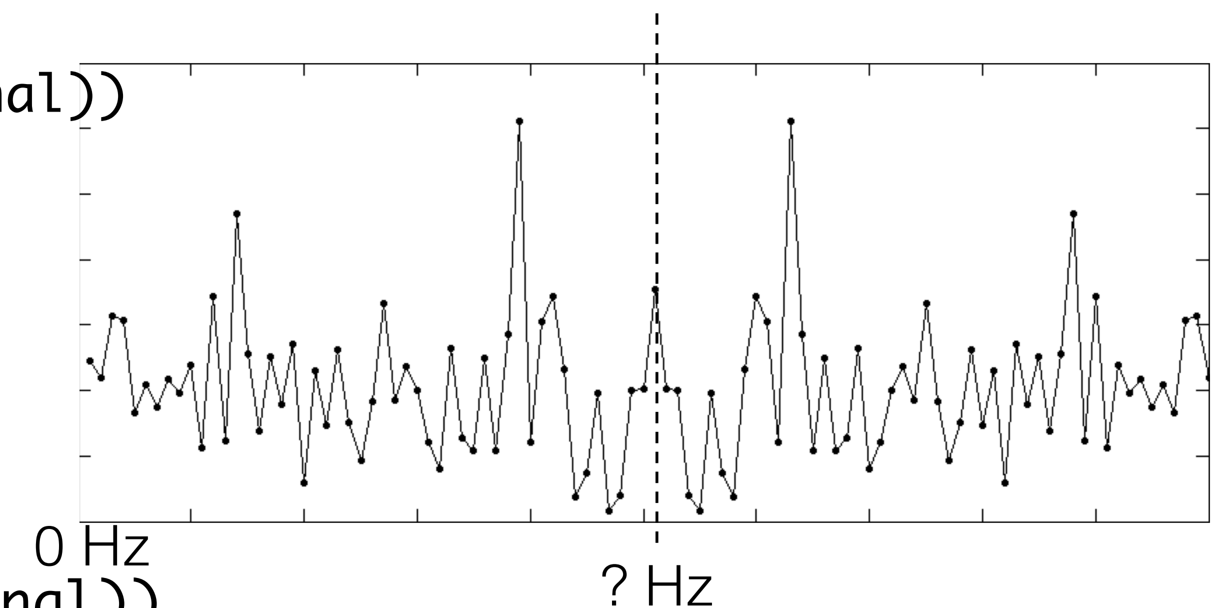
Amplitude spectrum example

random white noise
sampled at 10 Hz

```
randn(100,1)
linspace(0,10,100)
```




```
F    = abs(fft(signal))
F(1) = 12.3081
F(2) = 10.9257
F(3) = 15.6638
...
```

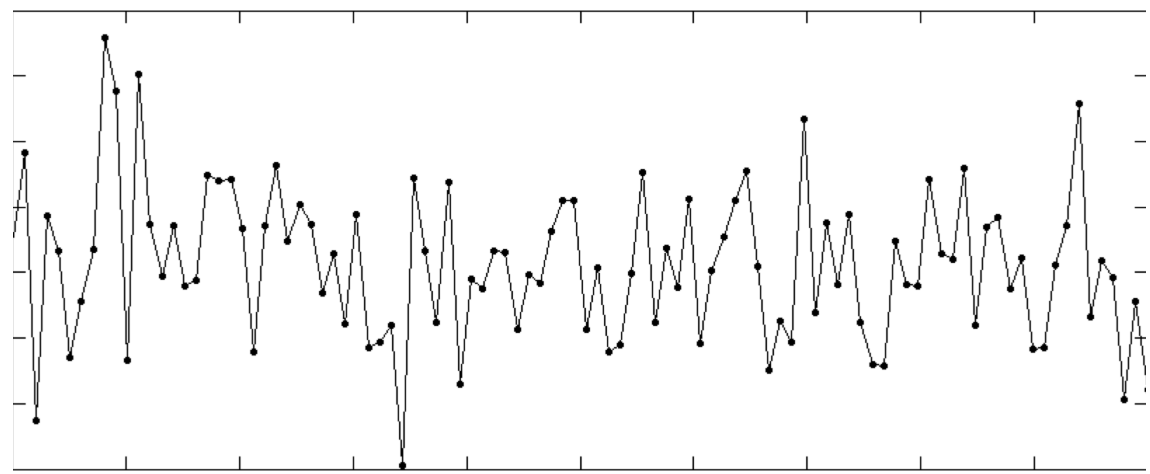


MATLAB: `abs(fft(signal))`

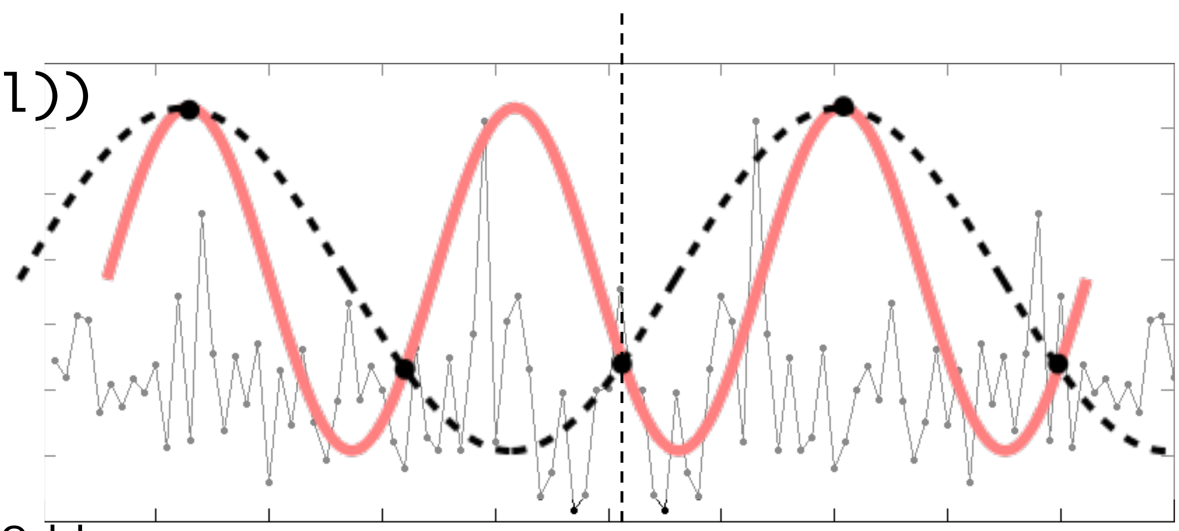
Amplitude spectrum example

random white noise
sampled at 10 Hz

```
randn(100,1)
linspace(0,10,100)
```




```
F     = abs(fft(signal))
F(1) = 12.3081
F(2) = 10.9257
F(3) = 15.6638
...
```



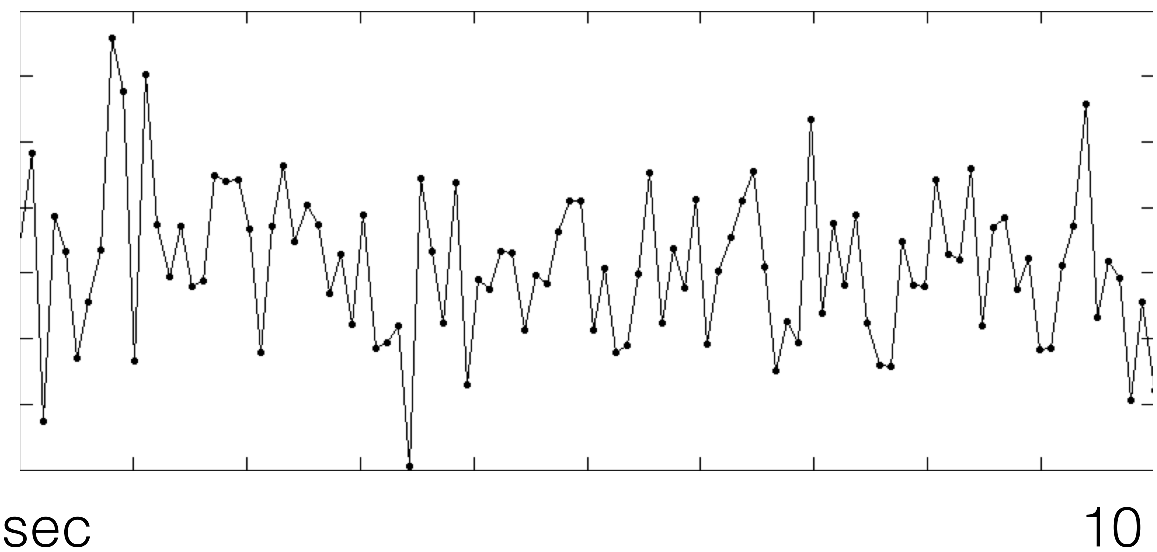
Nyquist frequency is 0.5 * sample frequency

MATLAB: `abs(fft(signal))`

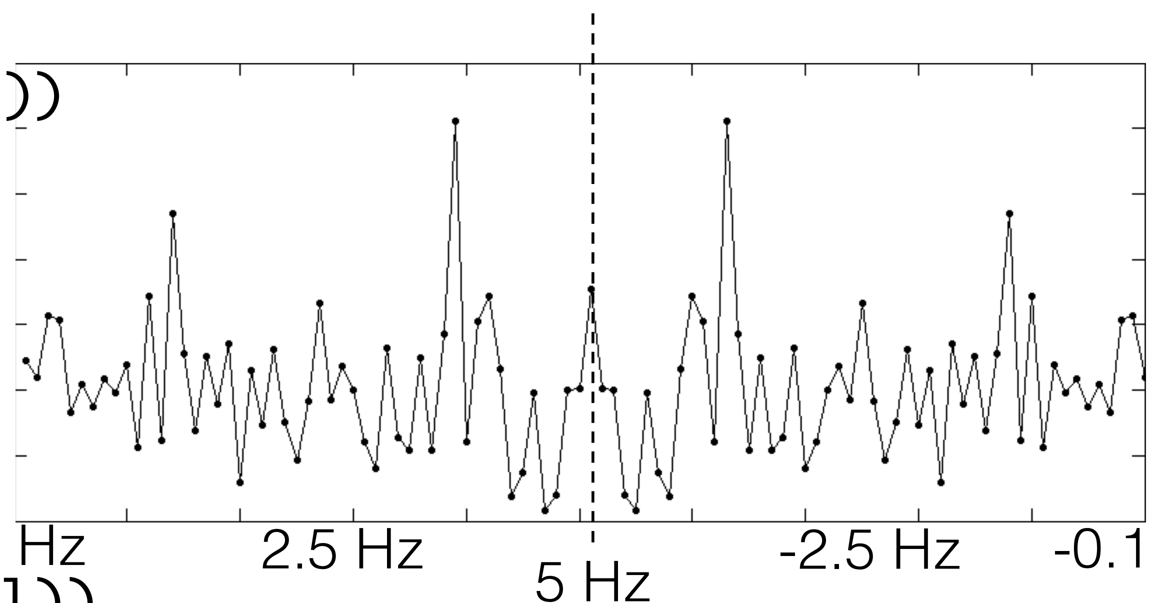
Amplitude spectrum example

random white noise
sampled at 10 Hz

`randn(100,1)`
`linspace(0,10,100)`




```
F    = abs(fft(signal))
F(1) = 12.3081
F(2) = 10.9257
F(3) = 15.6638
...
```

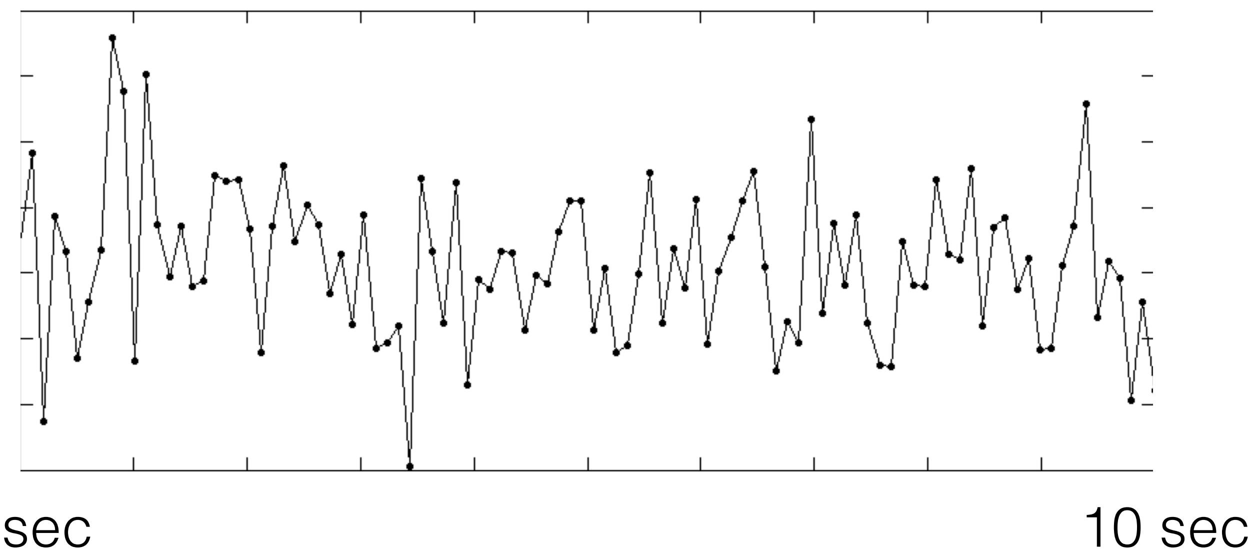


MATLAB: `abs(fft(signal))`

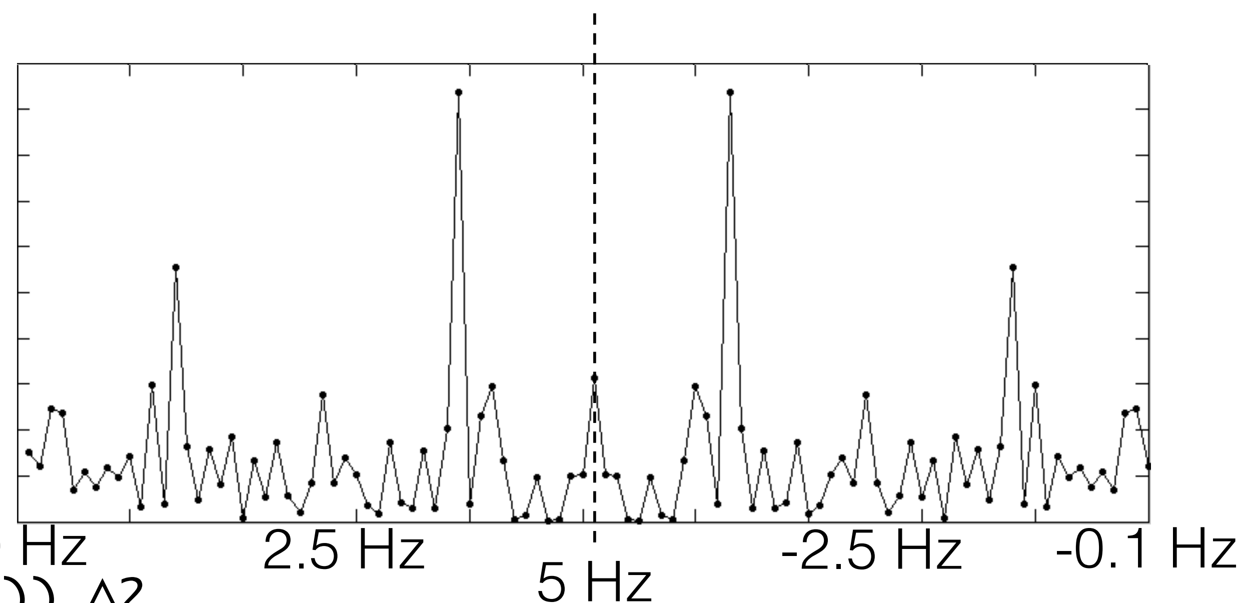
Power spectrum example

random white noise
sampled at 10 Hz

`randn(100,1)`
`linspace(0,10,100)`



power spectrum is just the squared amplitude spectrum

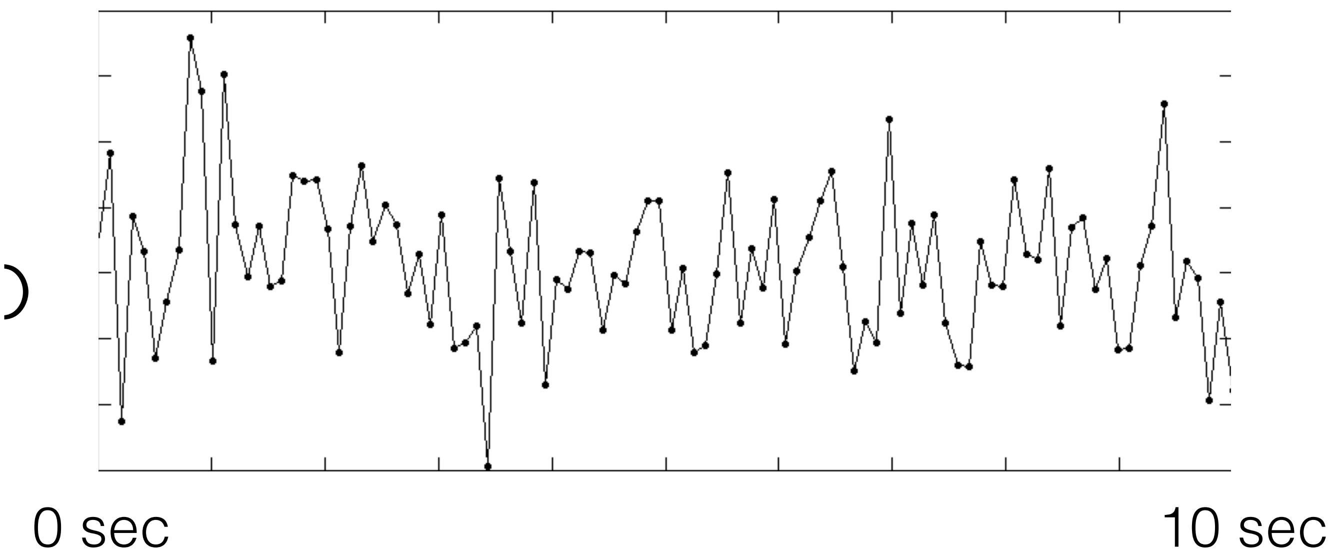


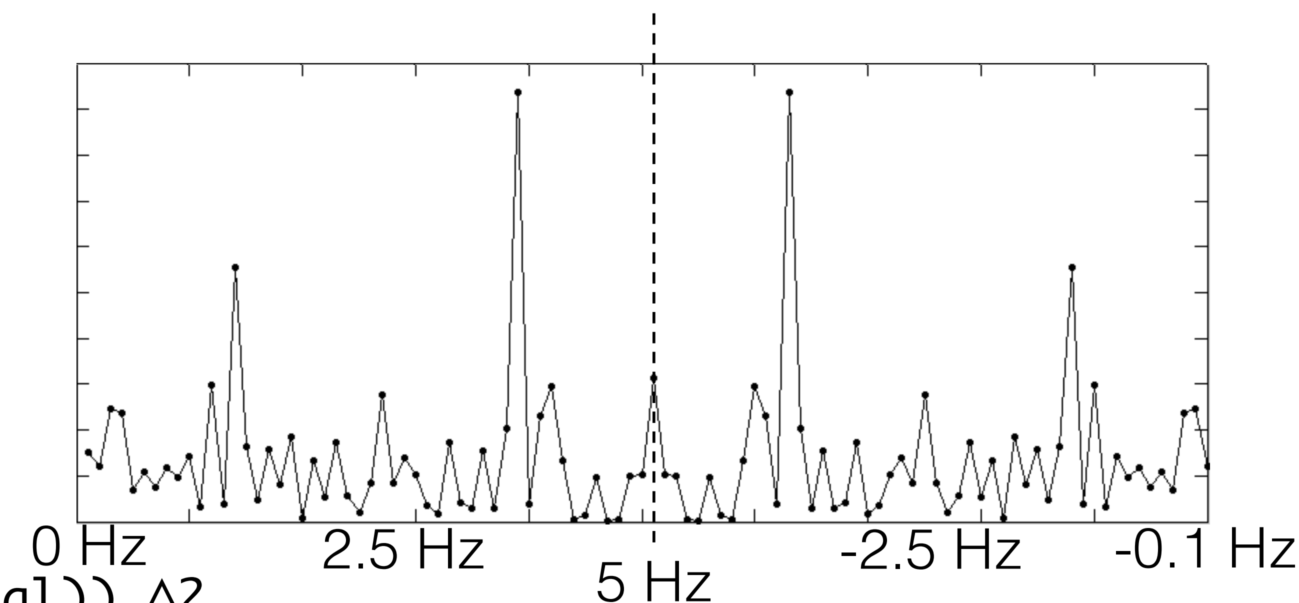
MATLAB: `abs(fft(signal)).^2`

What part do I actually care about?

random white noise
sampled at 10 Hz

`randn(100,1)`
`linspace(0,10,100)`



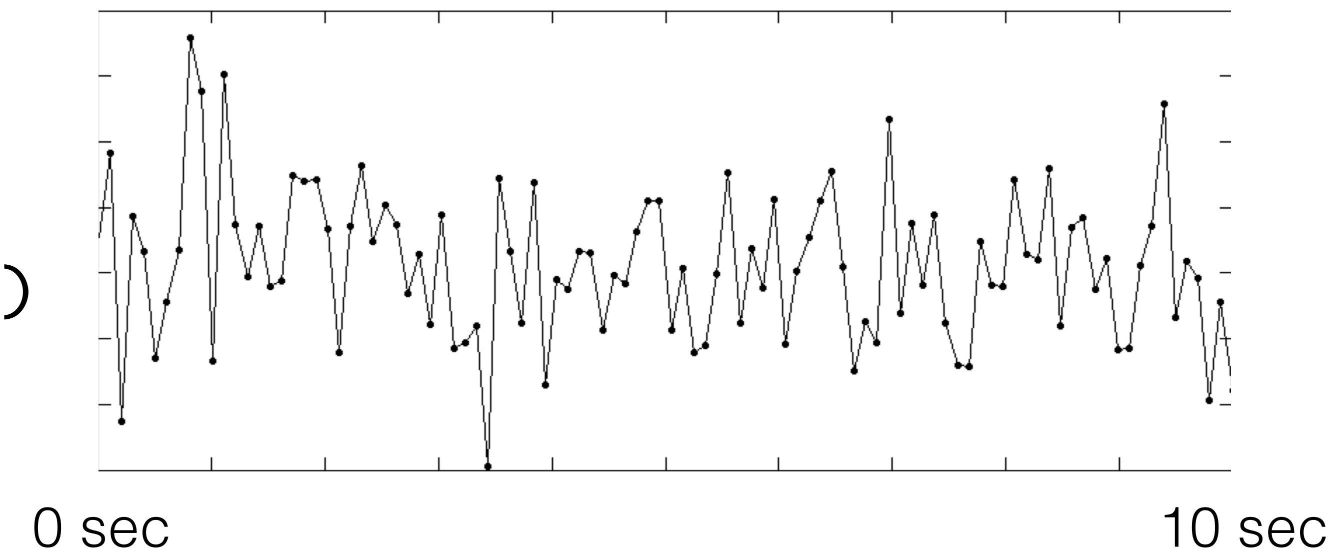


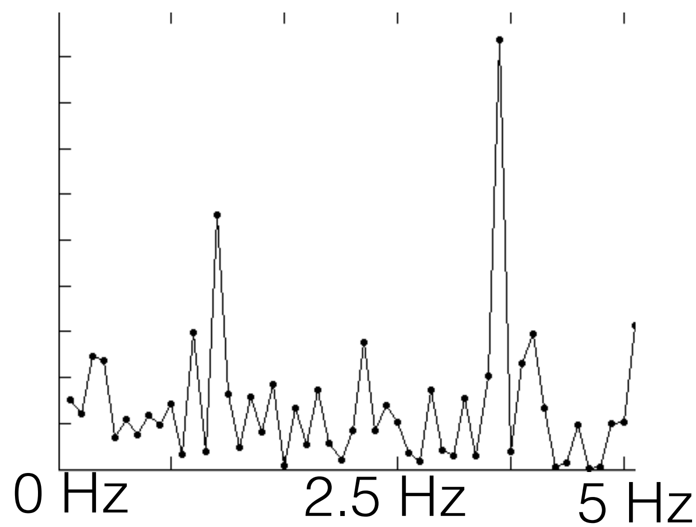
MATLAB: `abs(fft(signal)).^2`

What part do I actually care about?

random white noise
sampled at 10 Hz

```
randn(100,1)
linspace(0,10,100)
```





MATLAB (even length signal):

```
Ps = abs(fft(signal)).^2;
Ps = Ps(1:length(signal)/2+1)
```

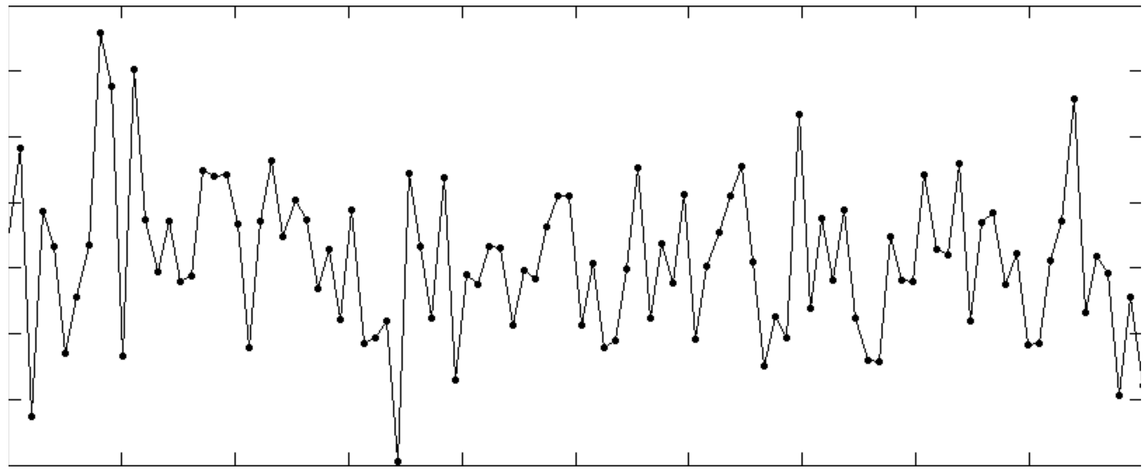
MATLAB (odd length signal):

```
Ps = abs(fft(signal)).^2;
Ps = Ps(1:(length(signal)-1)/2+1)
```

How to generate x-axis?

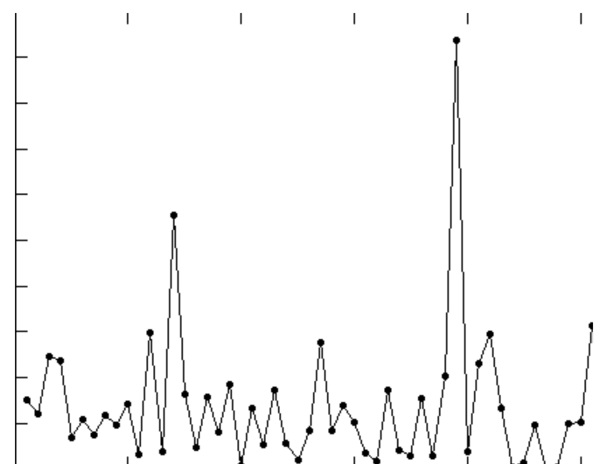
random white noise
sampled at 10 Hz

```
randn(100,1)
linspace(0,10,100)
```




```
%%      In general     %%
%% Fs is sample rate %%
freqs = linspace(0, Fs/2, length(Ps))

%% Our example %%
freqs = linspace(0, 5, 51)
```

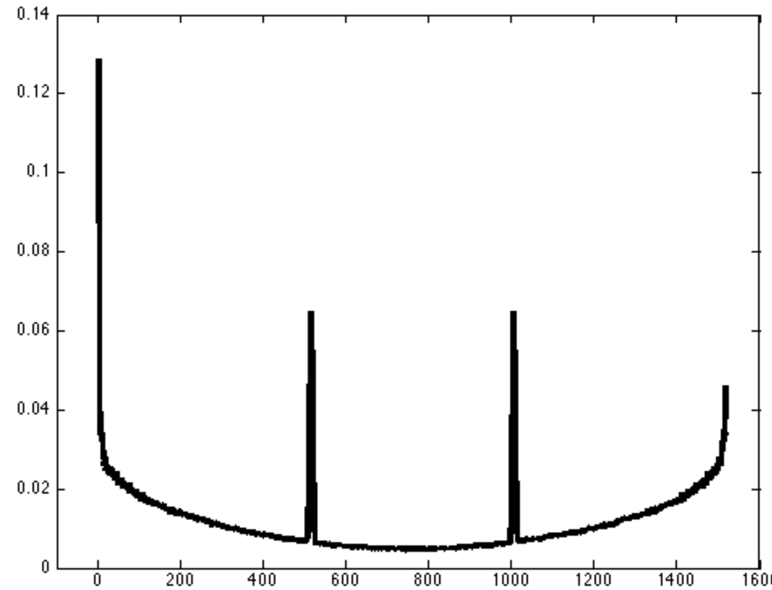


MATLAB (even length signal):

```
Ps = abs(fft(signal)).^2;
Ps = Ps(1:length(signal)/2+1)
```

Example: De-noising images

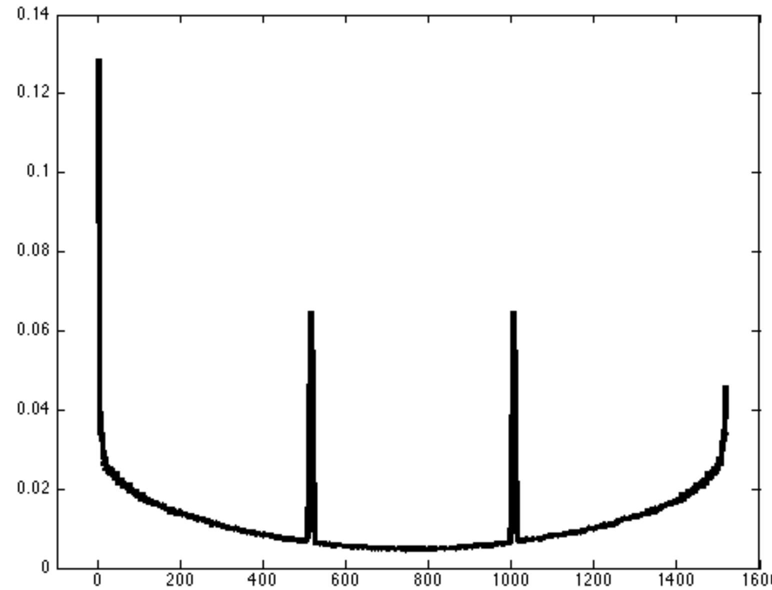
Example: De-noising images



MATLAB (odd length signal):

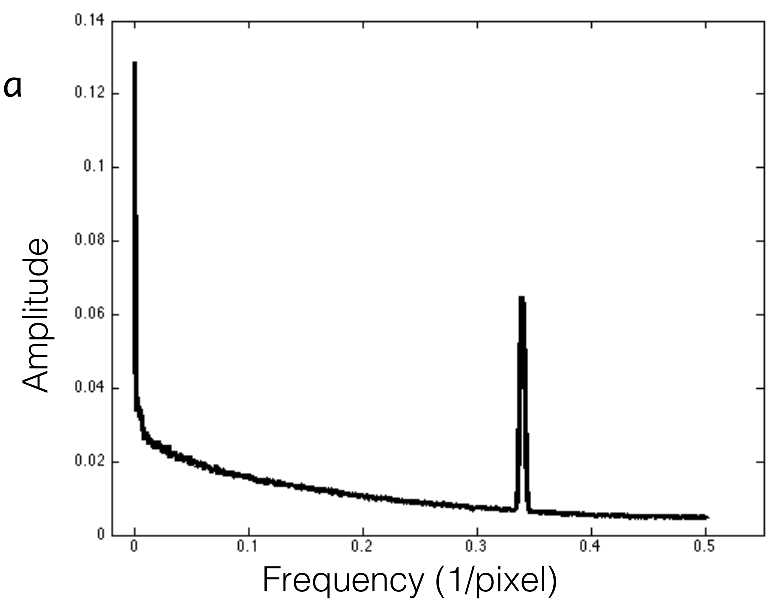
```
As_2d = (1/length(image(:)))*abs(fft2(image));
As_1d = mean(As_2d, 1); % average all horizontal spectra
```

Example: De-noising images

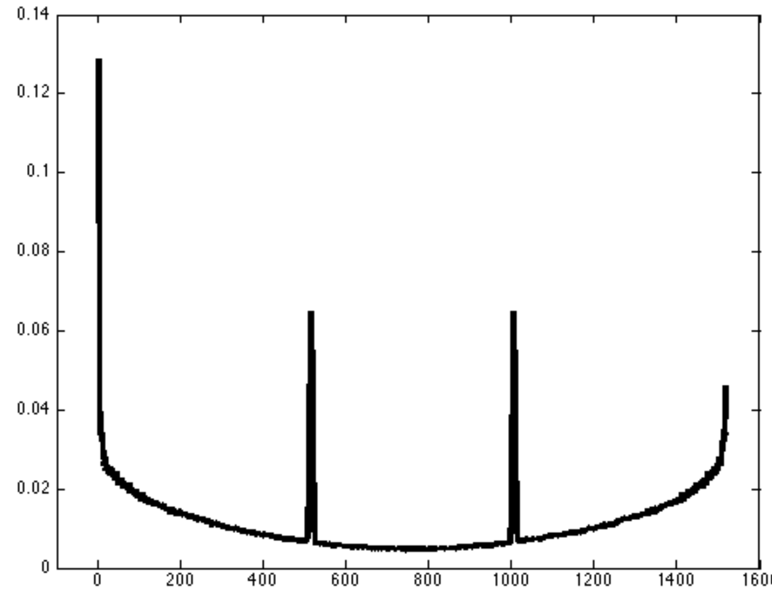


MATLAB (odd length signal):

```
As_2d = (1/length(image(:)))*abs(fft2(image));
As_1d = mean(As_2d, 1) % average all horizontal spectra
As    = As_1d(1:(length(As_1d)-1)/2+1)
freqs = linspace(0, 1/2, length(As))
```

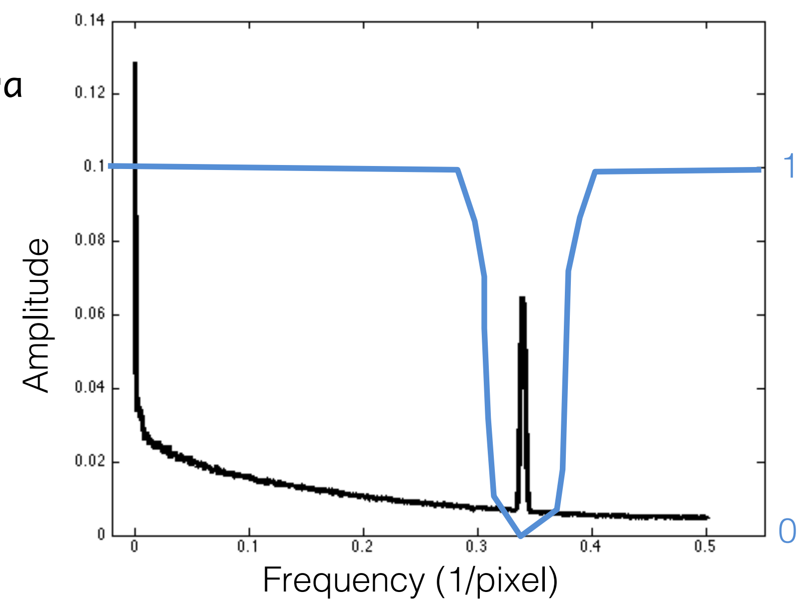


Example: notch filter

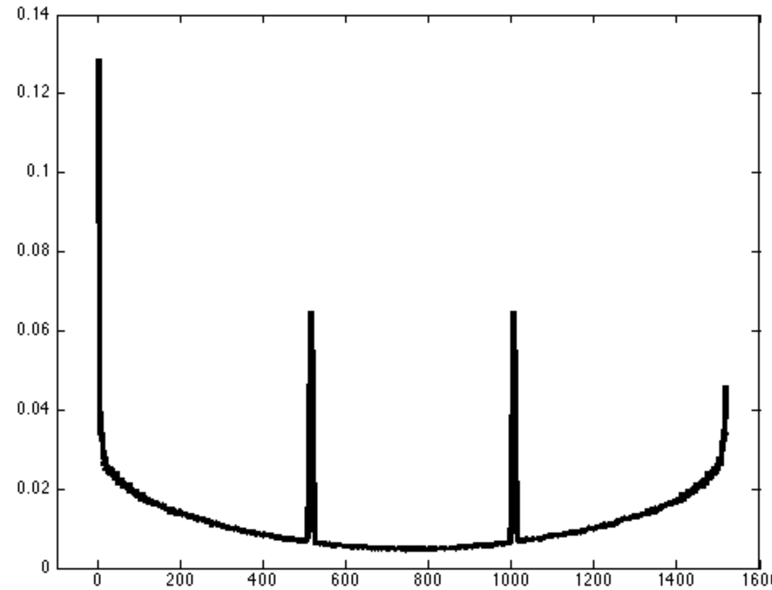


MATLAB (odd length signal):

```
As_2d = (1/length(image(:)))*abs(fft2(image));
As_1d = mean(As_2d, 1) % average all horizontal spectra
As    = As_1d(1:(length(As_1d)-1)/2+1)
freqs = linspace(0, 1/2, length(As))
```

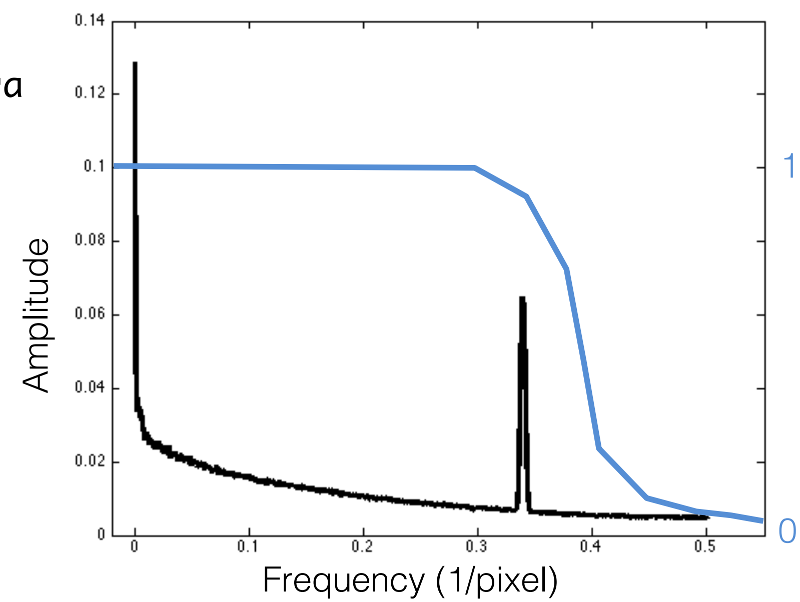


Example: low-pass filter

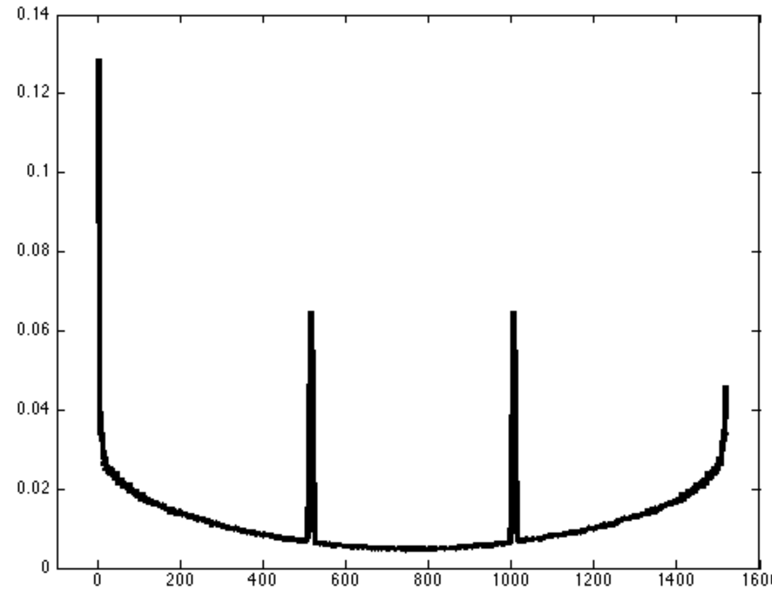


MATLAB (odd length signal):

```
As_2d = (1/length(image(:)))*abs(fft2(image));
As_1d = mean(As_2d, 1) % average all horizontal spectra
As    = As_1d(1:(length(As_1d)-1)/2+1)
freqs = linspace(0, 1/2, length(As))
```

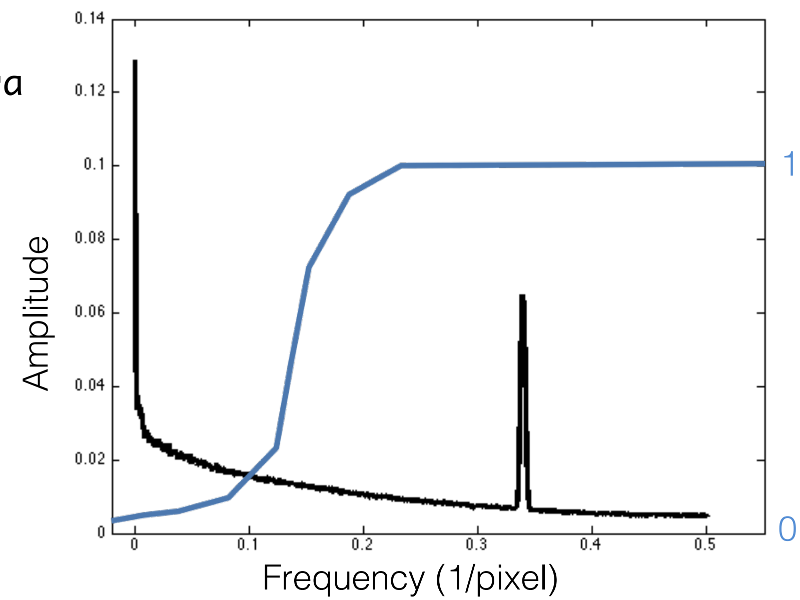


Example: high-pass filter

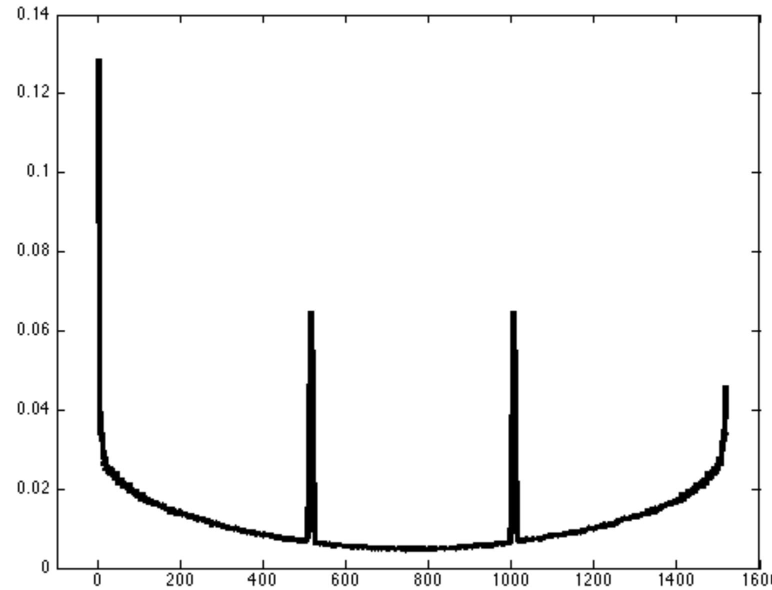


MATLAB (odd length signal):

```
As_2d = (1/length(image(:)))*abs(fft2(image));
As_1d = mean(As_2d, 1) % average all horizontal spectra
As    = As_1d(1:(length(As_1d)-1)/2+1)
freqs = linspace(0, 1/2, length(As))
```

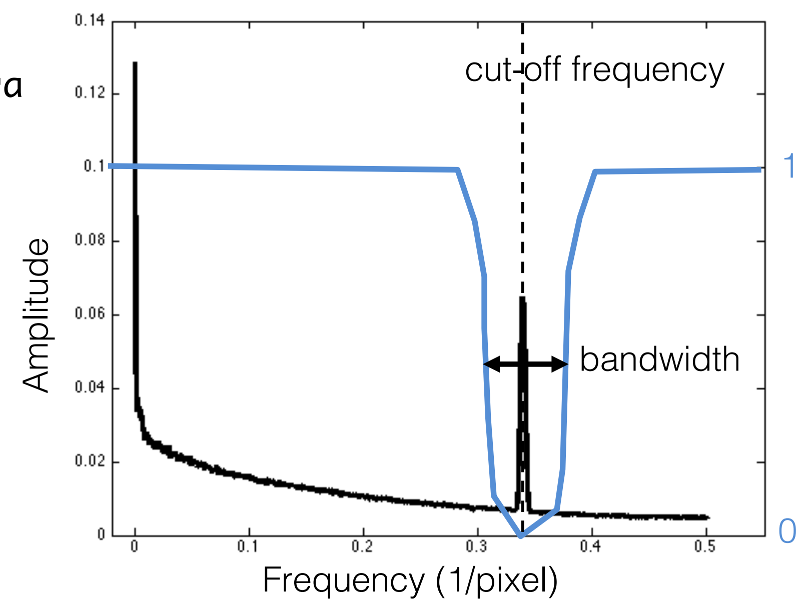


Example: notch filter

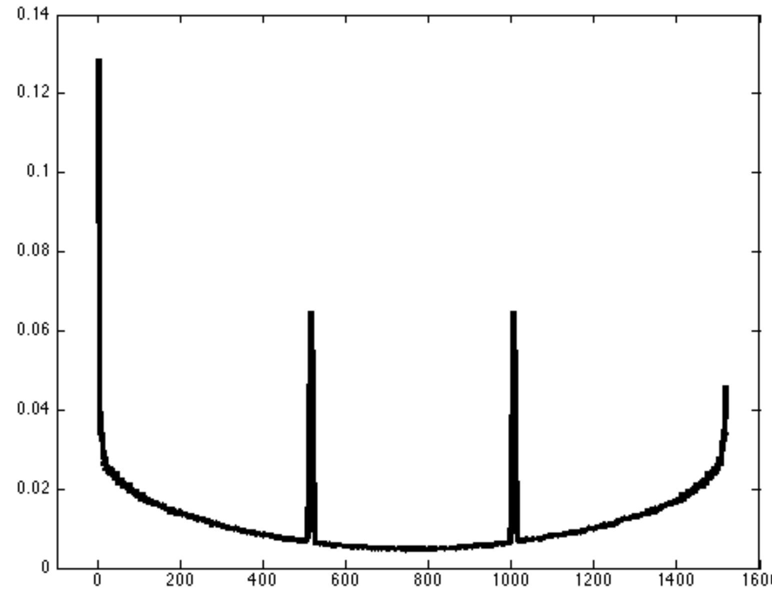


MATLAB (odd length signal):

```
As_2d = (1/length(image(:)))*abs(fft2(image));
As_1d = mean(As_2d, 1) % average all horizontal spectra
As    = As_1d(1:(length(As_1d)-1)/2+1)
freqs = linspace(0, 1/2, length(As))
```

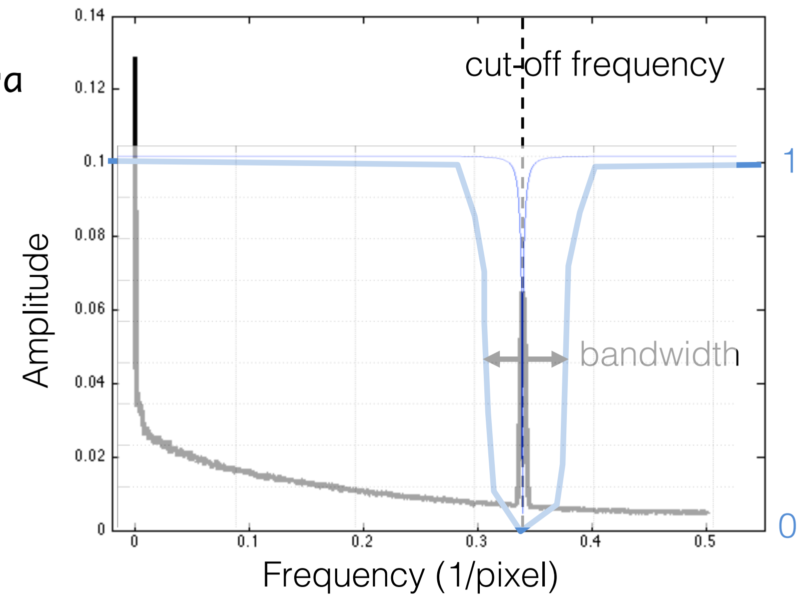


Example: notch filter



MATLAB (odd length signal):

```
As_2d = (1/length(image(:)))*abs(fft2(image));
As_1d = mean(As_2d, 1) % average all horizontal spectra
As    = As_1d(1:(length(As_1d)-1)/2+1)
freqs = linspace(0, 1/2, length(As))
```



MATLAB (notch filter design):

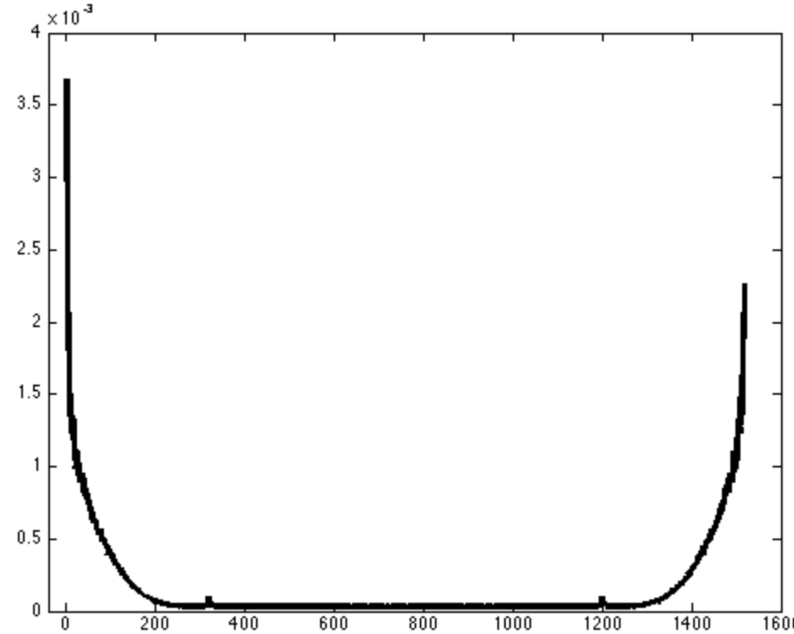
```
cutoff       = 0.34
bandwidth    = .05
[num, den]   = iirnotch(cutoff, bandwidth)
denoised_im = filter(num, den, image)
```

Example: De-noising images

Example: De-noising images

Example: De-noising images

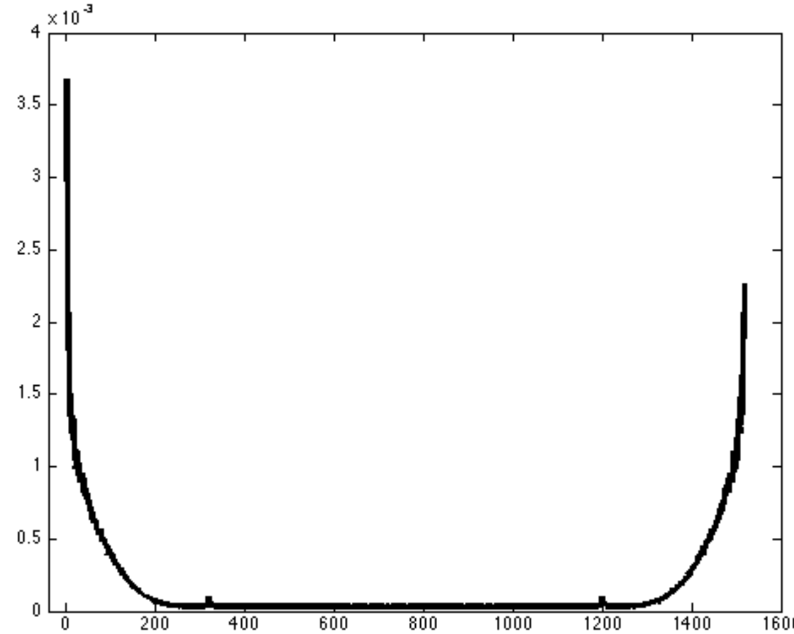
Example: De-noising images



MATLAB (odd length signal):

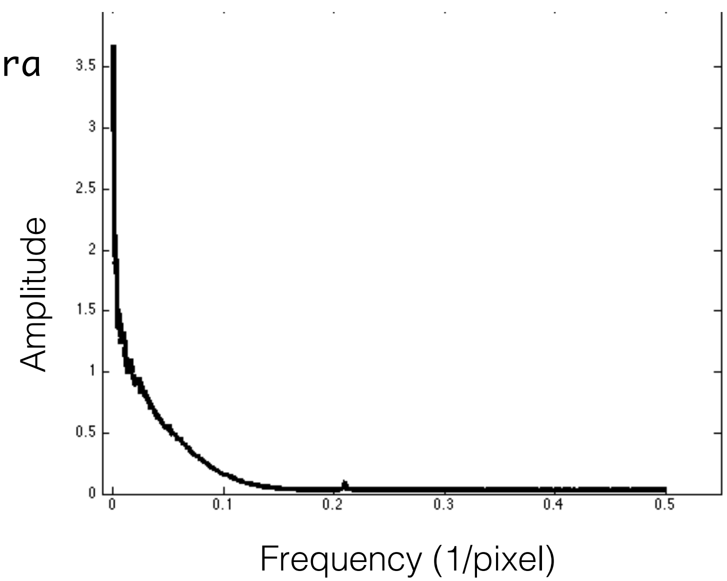
```
As_2d = (1/length(image(:)))*abs(fft2(image));
As_1d = mean(As_2d, 1); % average all horizontal spectra
```

Example: De-noising images

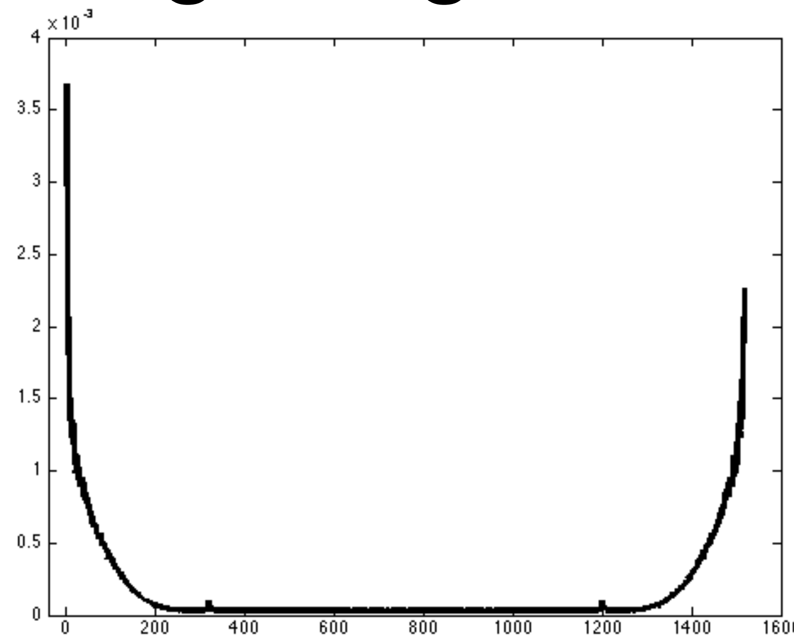


MATLAB (odd length signal):

```
As_2d = (1/length(image(:)))*abs(fft2(image));
As_1d = mean(As_2d, 1); % average all horizontal spectra
As    = As_1d(1:(length(As_1d)-1)/2+1);
freqs = linspace(0, 1/2, length(As));
```

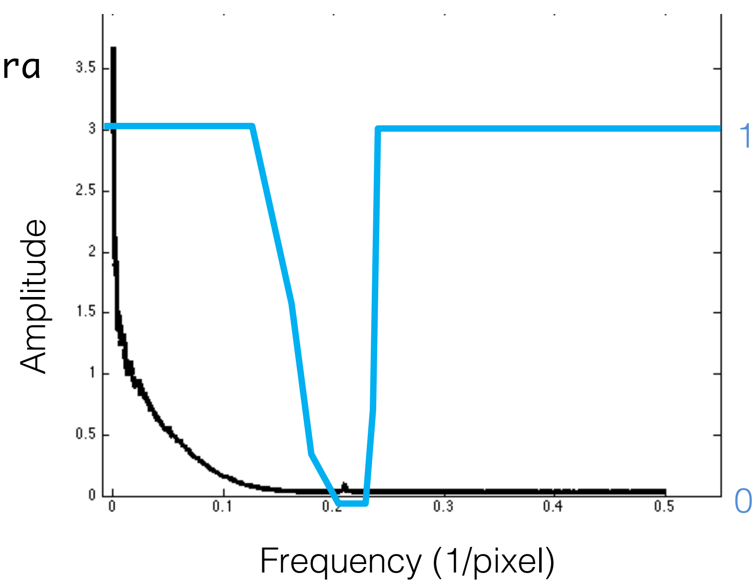


Example: De-noising images

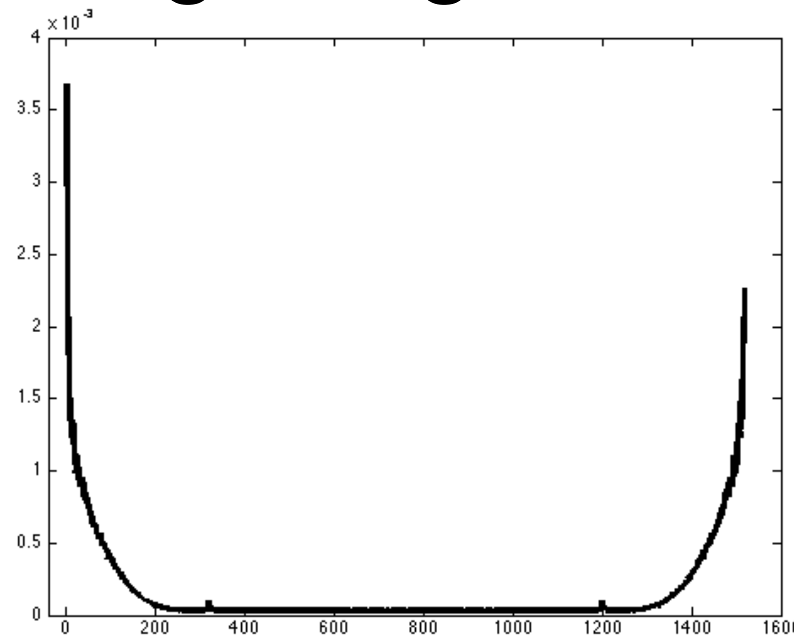


MATLAB (odd length signal):

```
As_2d = (1/length(image(:)))*abs(fft2(image));
As_1d = mean(As_2d, 1); % average all horizontal spectra
As    = As_1d(1:(length(As_1d)-1)/2+1);
freqs = linspace(0, 1/2, length(As));
```



Example: De-noising images

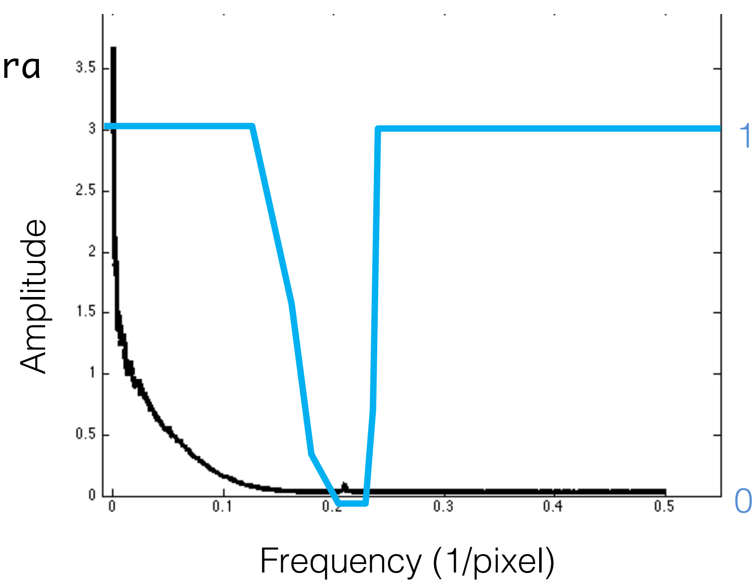


MATLAB (odd length signal):

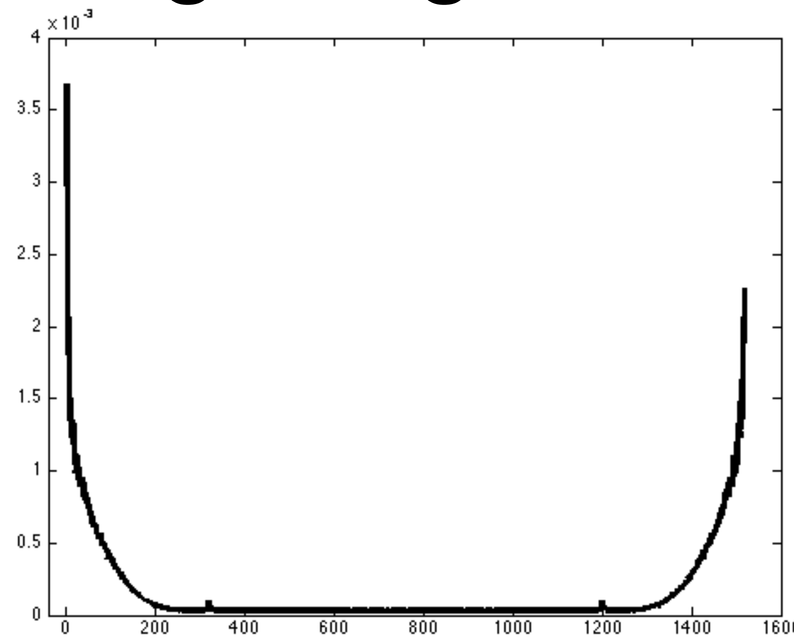
```
As_2d = (1/length(image(:)))*abs(fft2(image));
As_1d = mean(As_2d, 1); % average all horizontal spectra
As    = As_1d(1:(length(As_1d)-1)/2+1);
freqs = linspace(0, 1/2, length(As));
```

MATLAB (notch filter design):

```
cutoff       = 0.21
bandwidth    = .005
[num, den]   = iirnotch(cutoff, bandwidth);
denoised_im = filter(num, den, image);
```



Example: De-noising images

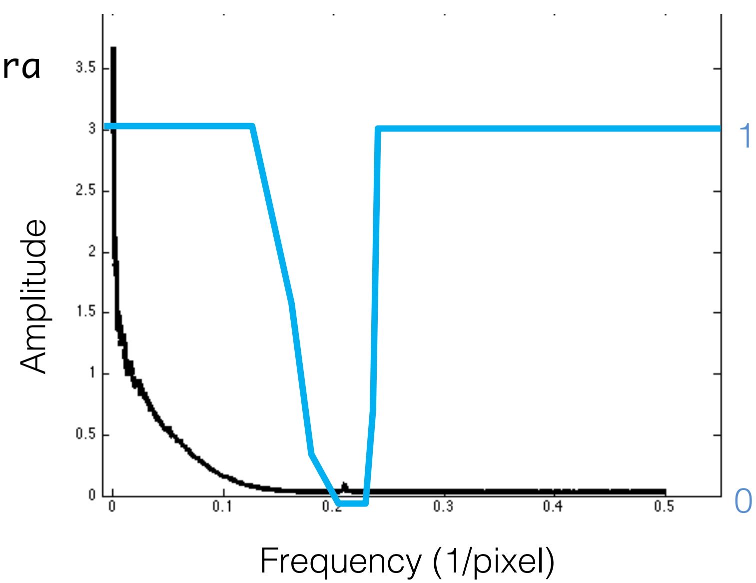


MATLAB (odd length signal):

```
As_2d = (1/length(image(:)))*abs(fft2(image));
As_1d = mean(As_2d, 1); % average all horizontal spectra
As    = As_1d(1:(length(As_1d)-1)/2+1);
freqs = linspace(0, 1/2, length(As));
```

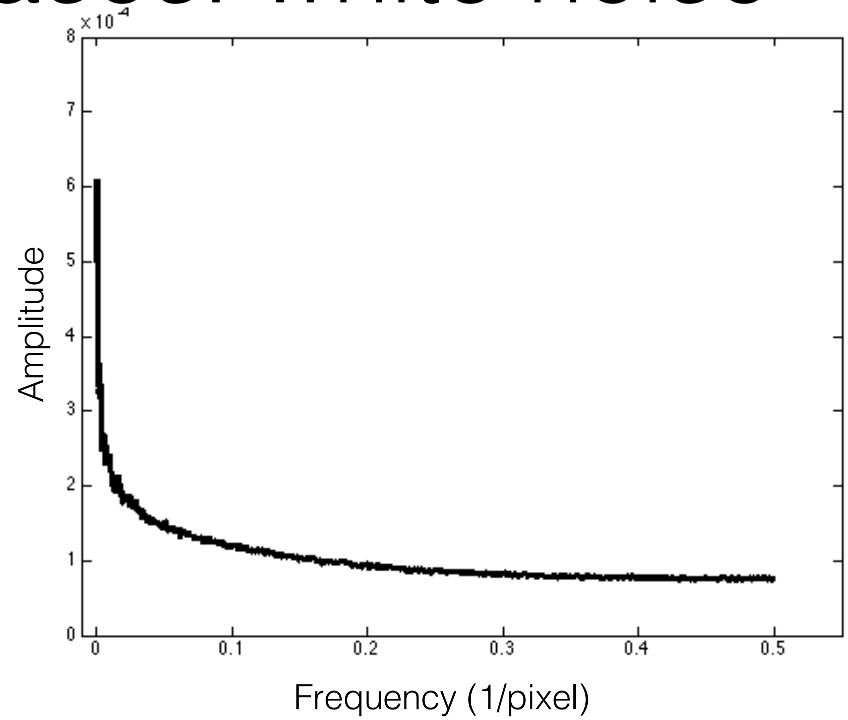
MATLAB (notch filter design):

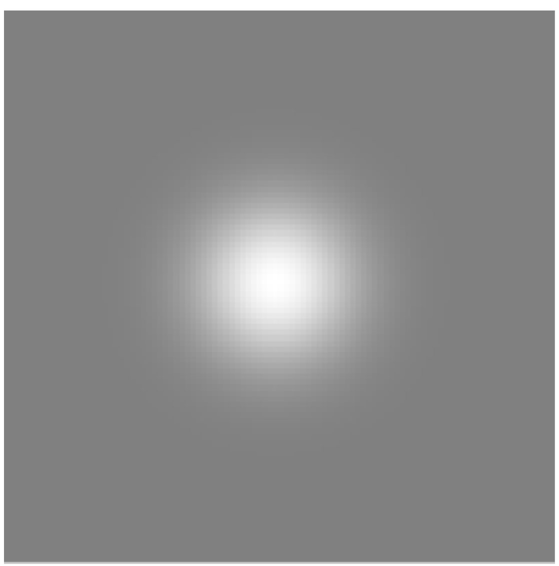
```
cutoff       = 0.15
bandwidth    = .5
[num, den]   = iirnotch(cutoff, bandwidth);
denoised_im = filter(num, den, image);
```



More difficult cases: white noise

More difficult cases: white noise

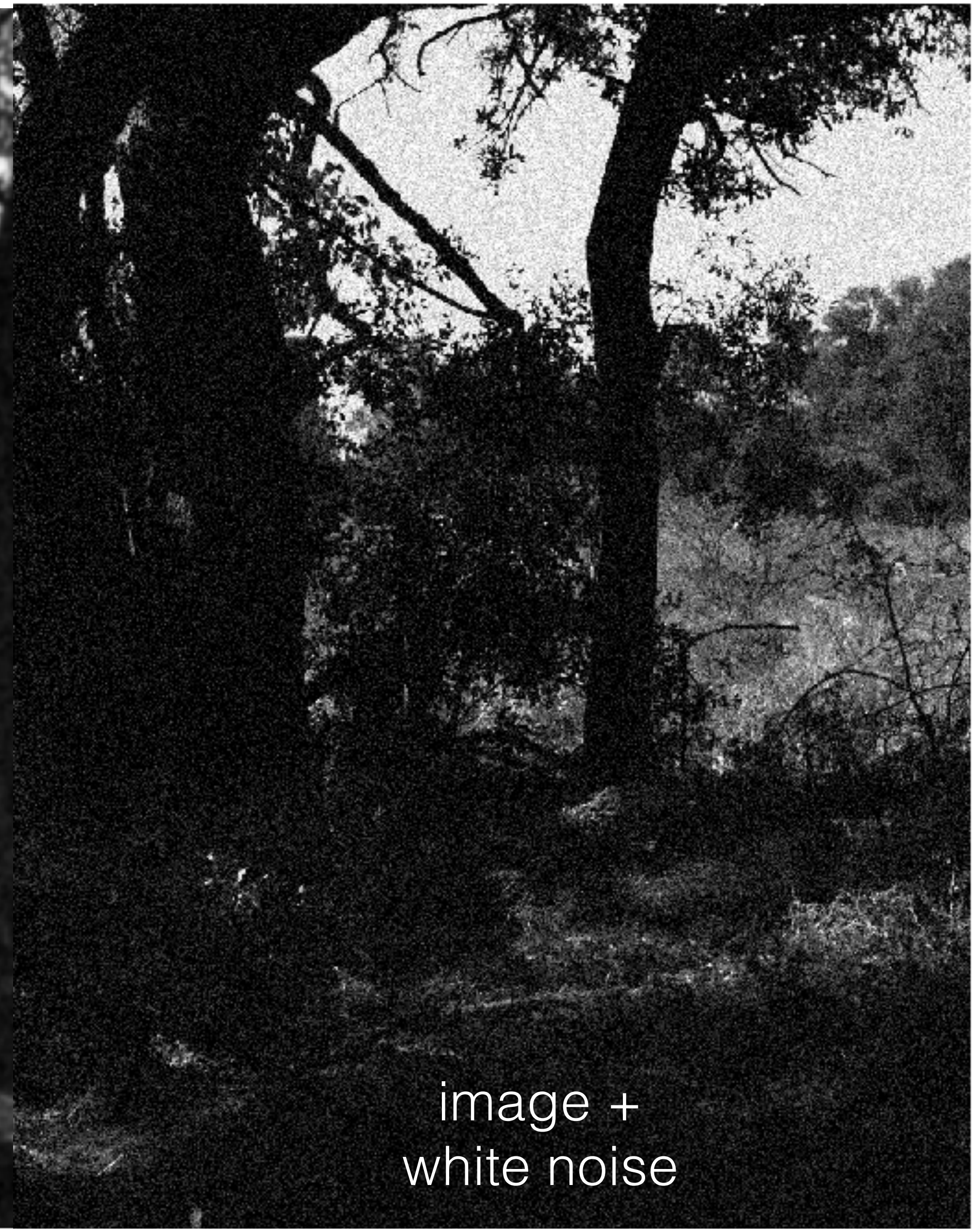




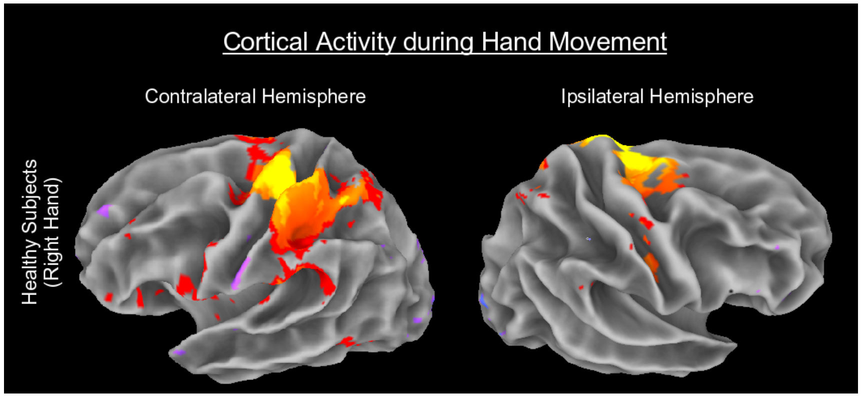
Gaussian filter

More difficult cases: white noise

More difficult cases: white noise

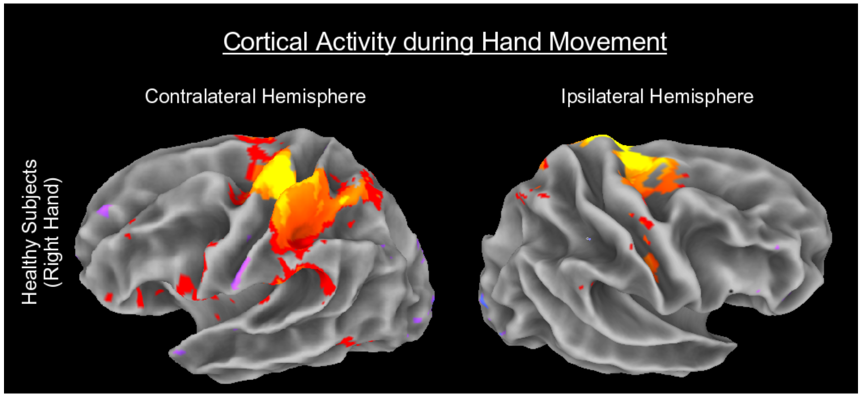


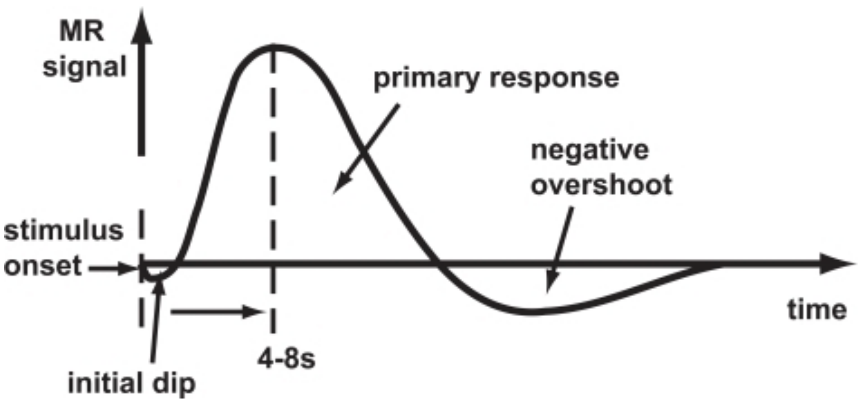
Example: Deconvolution and fMRI



fMRI

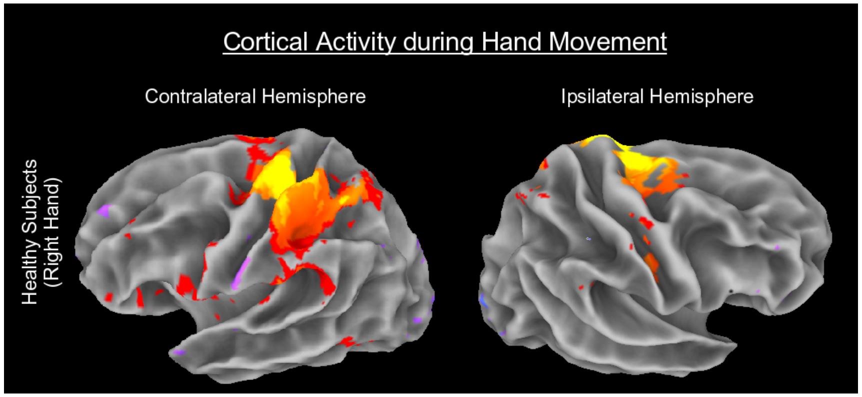
Example: Deconvolution and fMRI

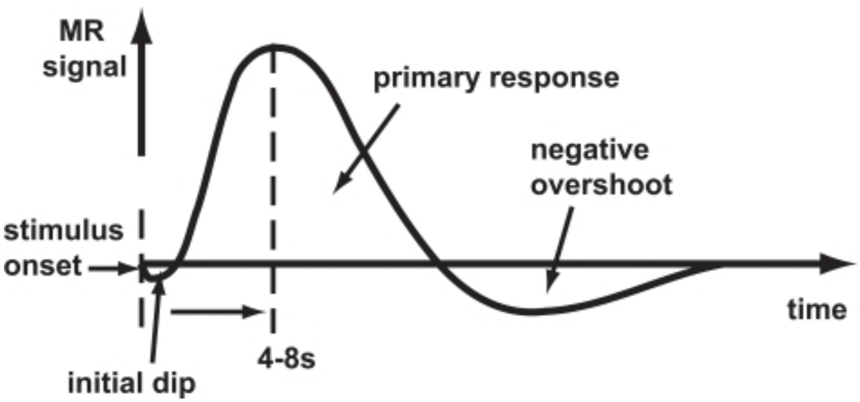




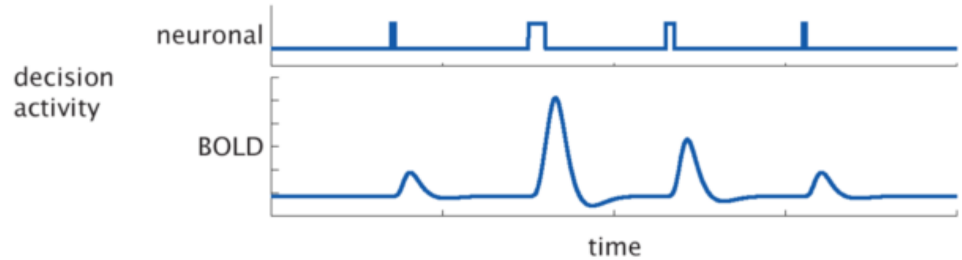
average BOLD response

Example: Deconvolution and fMRI





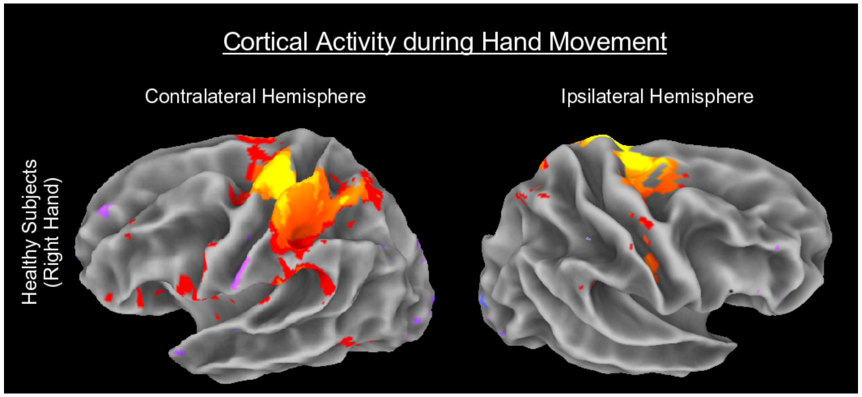
average BOLD response

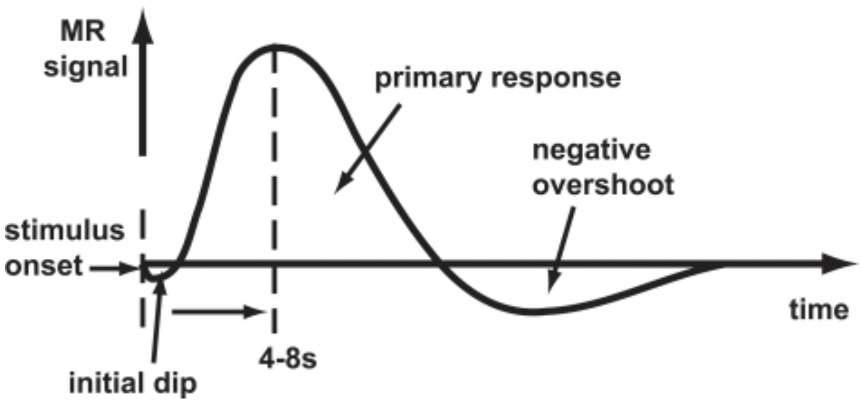


BOLD signal is the convolution of the original neuronal response and the average BOLD response

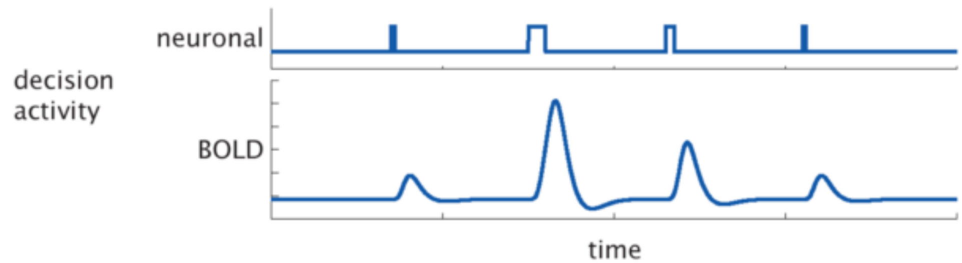
How do we get back the original neuronal response?

Example: Deconvolution and fMRI





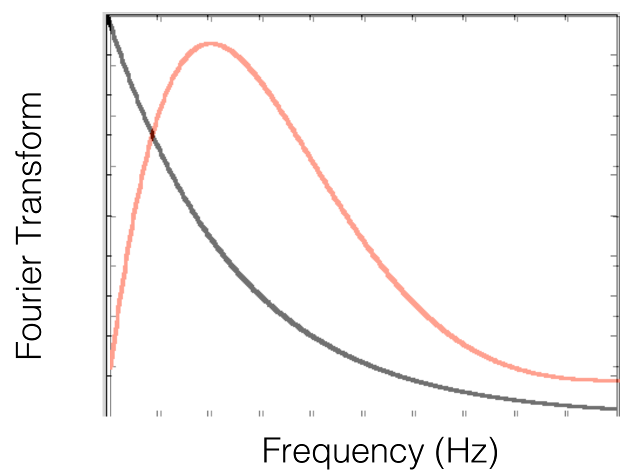
average BOLD response



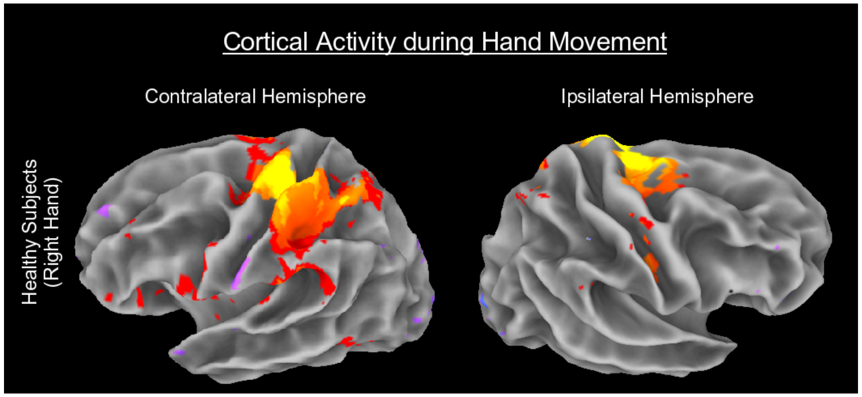
BOLD signal is the convolution of the original neuronal response and the average BOLD response

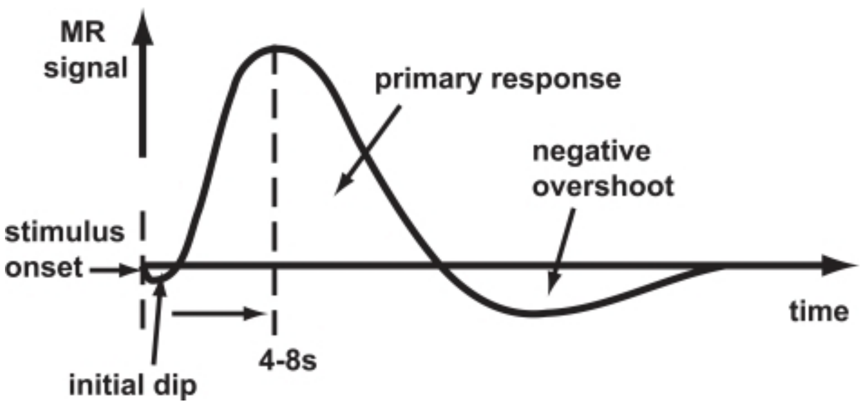
How do we get back the original neuronal response?

BOLD signal, average BOLD response

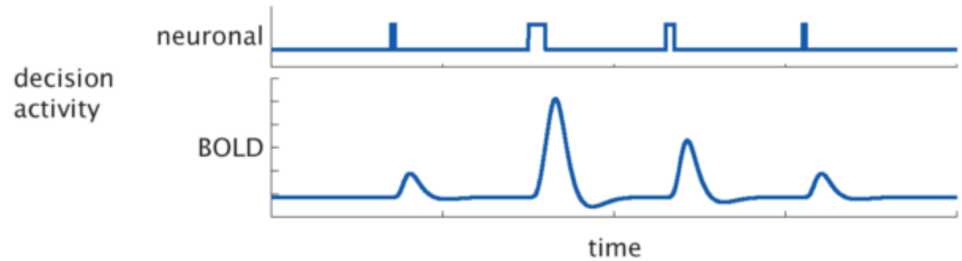


Example: Deconvolution and fMRI





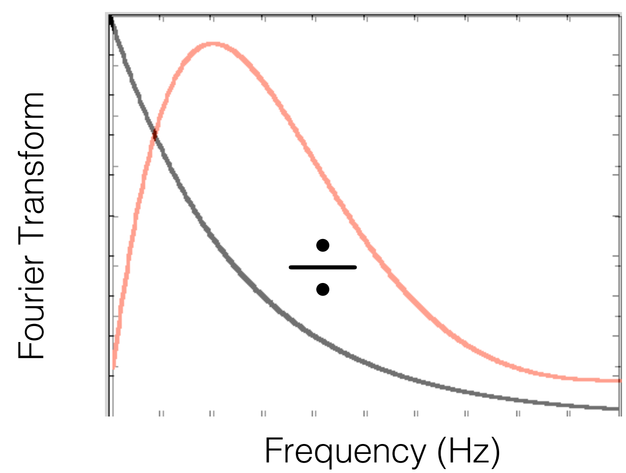
average BOLD response



BOLD signal is the convolution of the original neuronal response and the average BOLD response

How do we get back the original neuronal response?

BOLD signal, average BOLD response



MATLAB:

```
neuronal_fft  = signal_fft ./ avg_response_fft
neuronal      = ifft(neuronal_fft)

% alternatively, just do this:
neuronal = deconv(signal, avg_response)
```

Further Reading

Analyzing behavior

Long and Fee, 2008. “Using temperature to analyse temporal dynamics in the songbird motor pathway” *Nature*

Cortical circuits

Giocomo et al. 2007. “Temporal frequency of subtreshold oscillations scales with entorhinal grid cell field spacing” *Science*

Uhlhaas and Singer, 2010. “Abnormal neural oscillations and synchrony in schizophrenia.” *Nature Neuro Reviews*

Coding

Pollen and Ronner, 1982. “Visual cortical neurons as localized spatial frequency filters.” *IEEE*

van Hateren, 1992. “Theoretical predictions of spatiotemporal receptive fields of fly LMCs, and experimental validation” *Journal of Comp. Physiol.*

Acknowledgements

Some of the slides on Fourier transforms were adapted from the original Math Tools powerpoint slides