Topic

Convolution and Correlation

Lecture by: Fatemeh Pishdadian

Outline

- Convolution
- Correlation

Convolution

- **convolution** is a mathematical operator which takes two functions *x* and *h* and produces a third function that represents the amount of overlap between *h* and a reversed and translated version of *x*.
- In signal processing, one of the functions (*h*) is taken to be a fixed filter *impulse response*, and the other (x) the input signal.

$$(h * x)(t) \equiv \int_{-\infty}^{\infty} h(\tau)\, x(t-\tau)\, d\tau$$

Convolution operator

Why do we care about convolution?

- You can digitally simulate having recorded something in a place you’ve never been by using convolution reverb.

- With a convolution reverb, you can record your voice in your bedroom and then make it sound like it was recorded in Carnegie Hall.

- The convolution operator makes convolution reverb possible (more on that next lecture).

Discrete Convolution

- **convolution** is a mathematical operator which takes two functions *x* and *h* and produces a third function that represents the amount of overlap between *h* and a reversed and translated version of *x*.
- In signal processing, one of the functions (h) is taken to be a fixed filter *impulse response*, and the other (x) the input signal.

$$(h * x)[n] \equiv \sum_{k=-\infty}^{\infty} h[k]x[n-k]$$

Convolution operator

Convolution In Python Code

```
import numpy as np

def convolution(A,B):

      lengthA = np.size(A)
      lengthB = np.size(B)

      C = np.zeros(lengthA + lengthB -1)

      for m in np.arange(lengthA):
           for n in np.arange(lengthB):
                C[m+n] = C[m+n] + A[m]*B[n]

      return C
```

Some things to note

- If you convolve two vectors of length ***n***, you end up with a new vector of length **2n+1**
- The algorithm on the previous slide takes **n^2** time.
- Although that algorithm embodies the definition of convolution in code, there is a faster way of doing it involving the fast Fourier transform.
- We'll get to that faster version later.

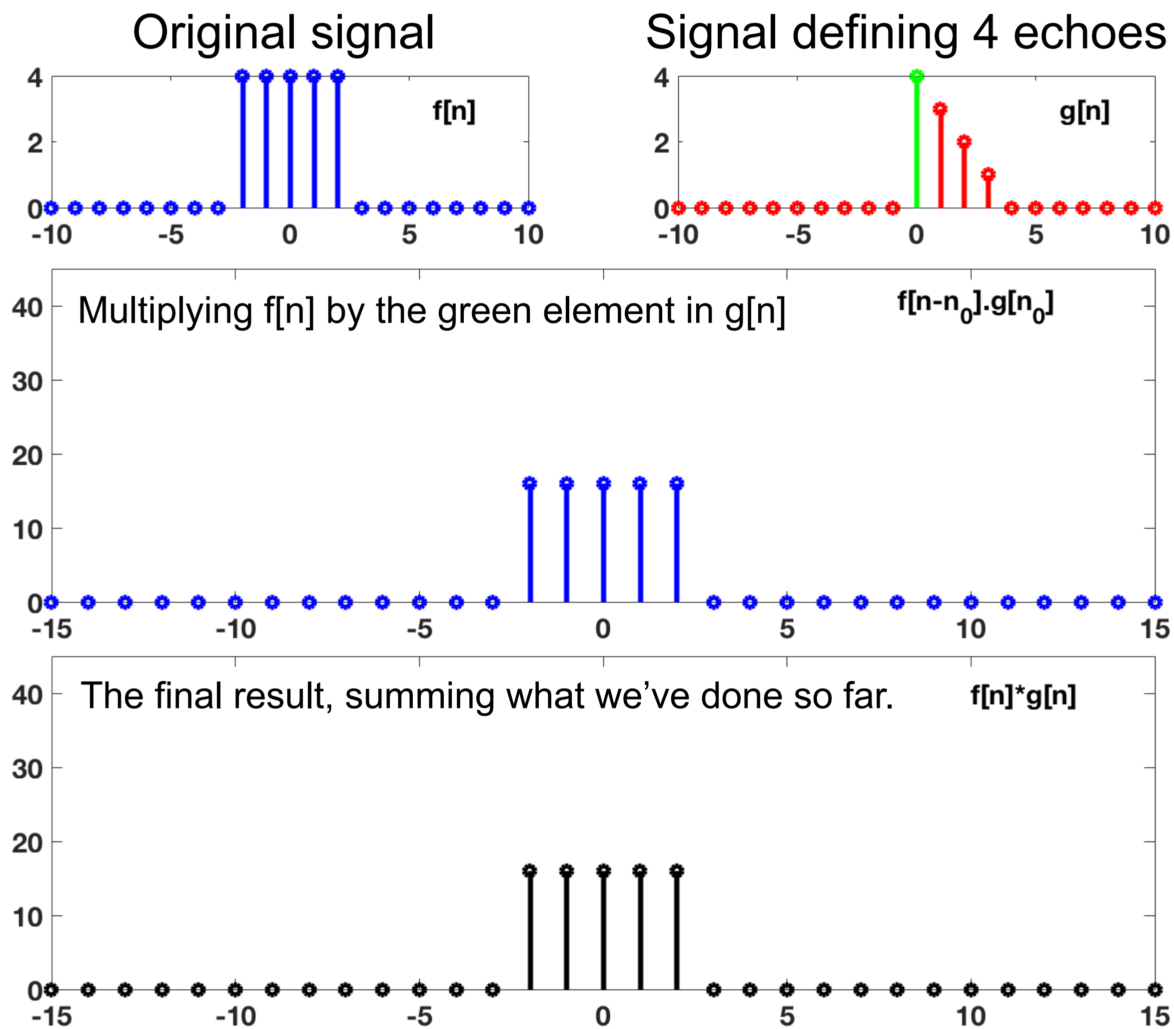
What is reverberation?

- Say I have a recording of some audio (e.g. my voice) that's been turned into a digital waveform (i.e. a time-series f[n])
- I want to add echoes (aka reverberation) to it. What would I do?
- Think of an echo as a delayed and diminished copy of the original f[n].
- I would make several delayed and diminished copies of f[n] and then add them all together.
- Voila, I've added reverb!

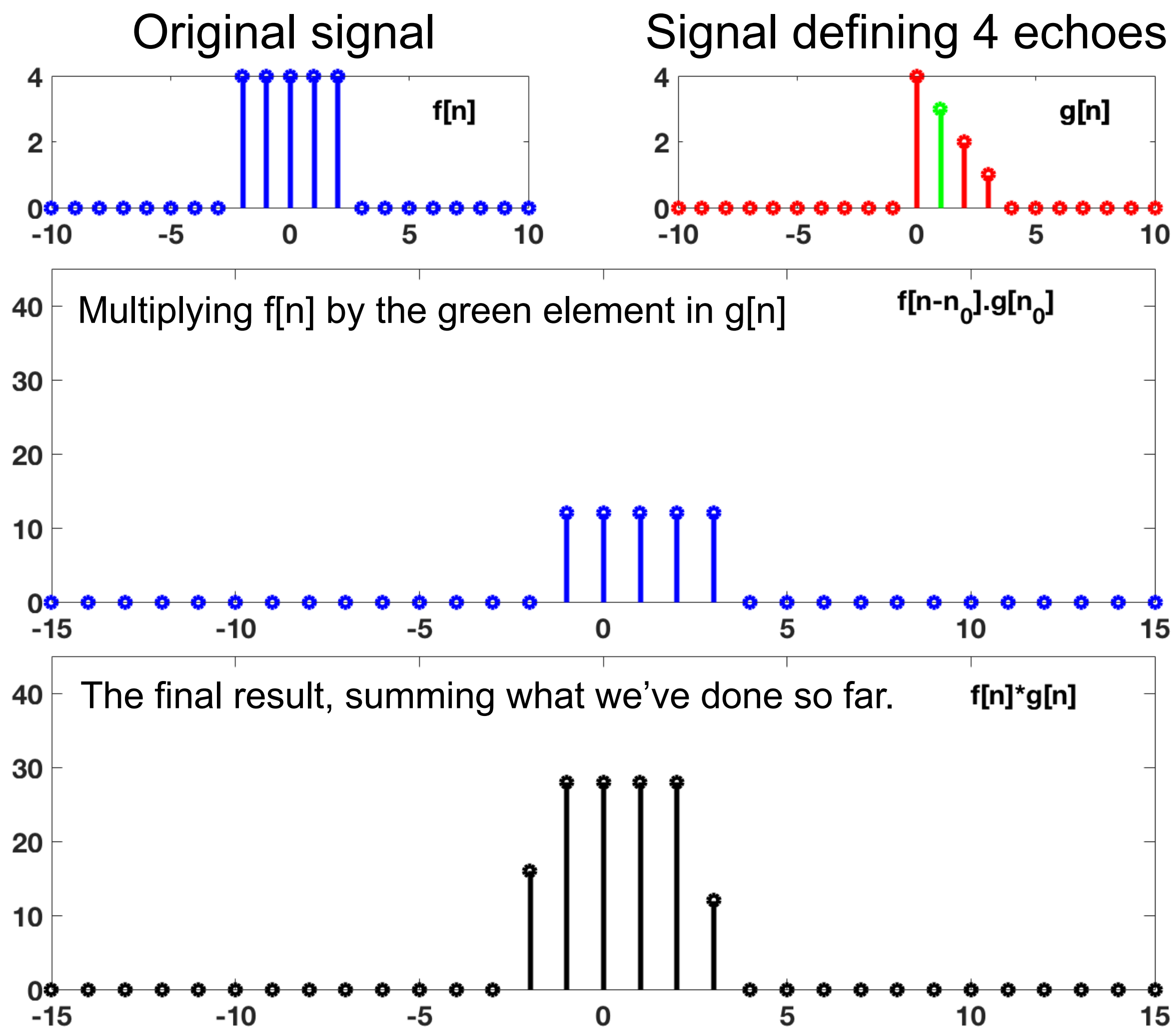
How does convolution relate?

- Let's say I want to make 4 copies of my signal f[n], each one delayed from the previous copy by 1 sample, and reduced in amplitude.
- Think of this as the original signal, plus three echoes.
- I can define a 2nd signal (call it g[n]) that looks like 4 impulses, each smaller than the previous one.
- Then I convolve f[n] and g[n].
- This makes 4 copies of f[n], with the kth copy being scaled and delayed by the kth element in g[n]. Then, they are all summed up.
- The result is a signal with 3 echoes added. Let's look!

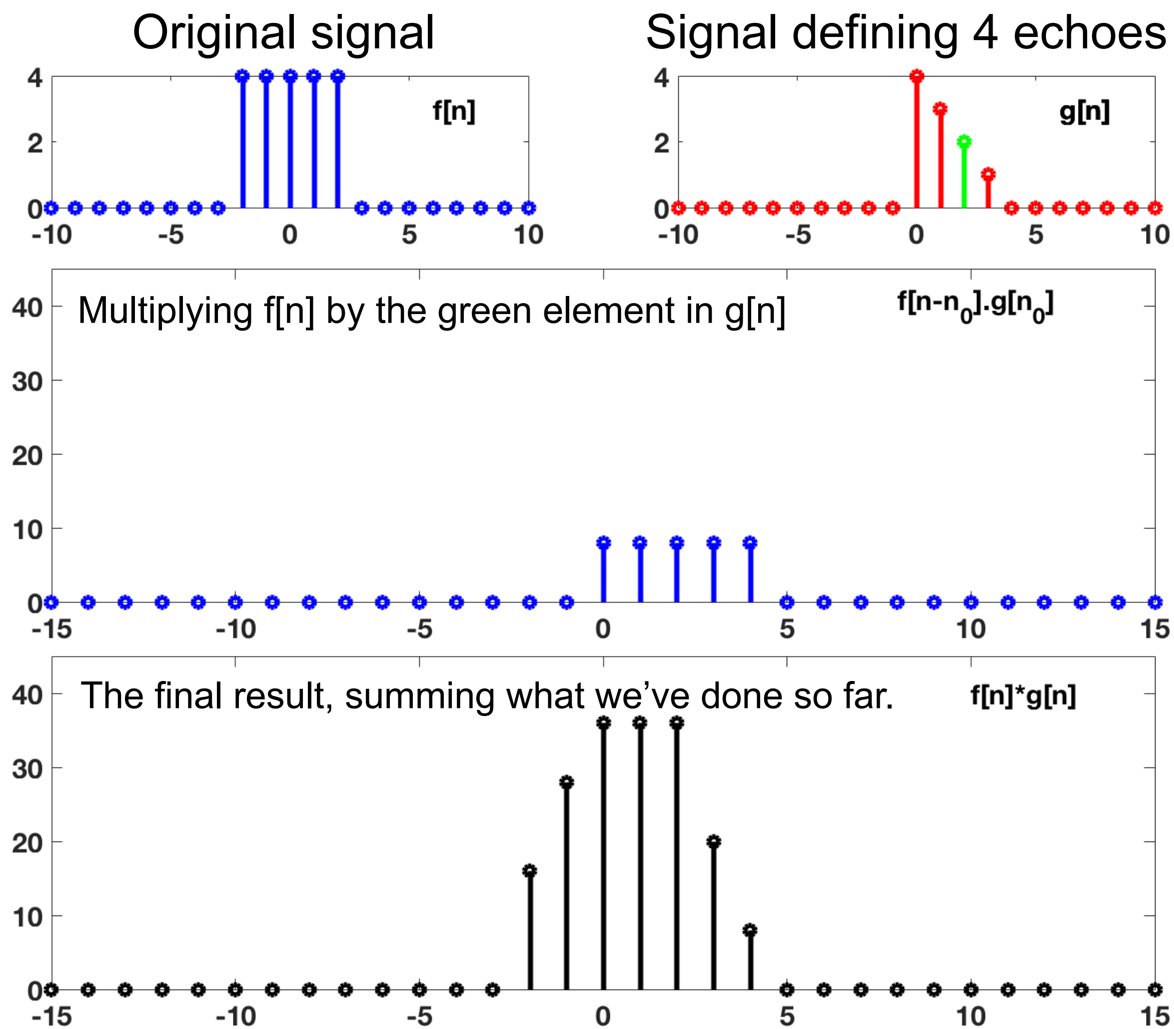
Let's look at Convolution



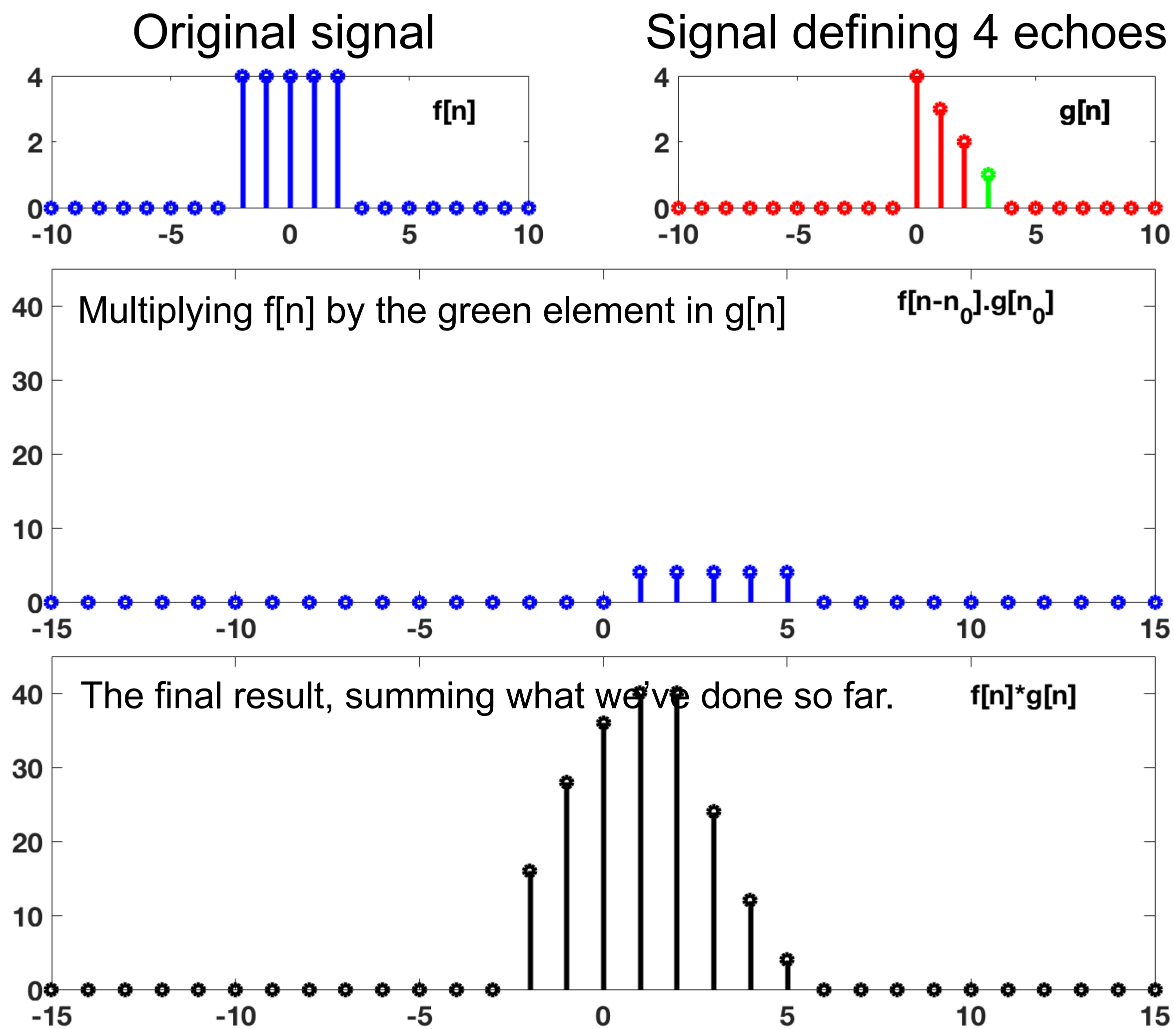
Let's look at Convolution



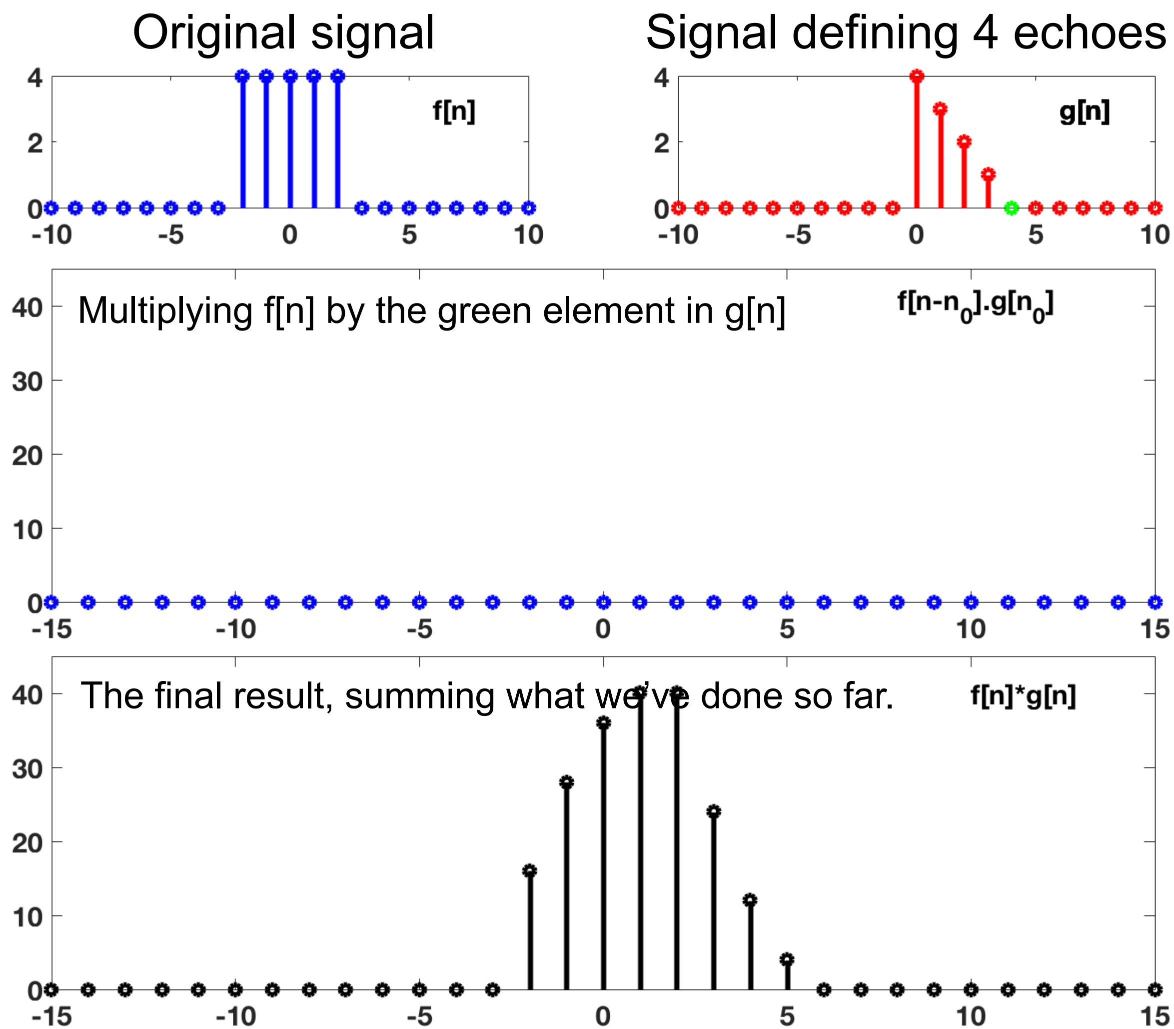
Let's look at Convolution



Let's look at Convolution



Let’s look at Convolution



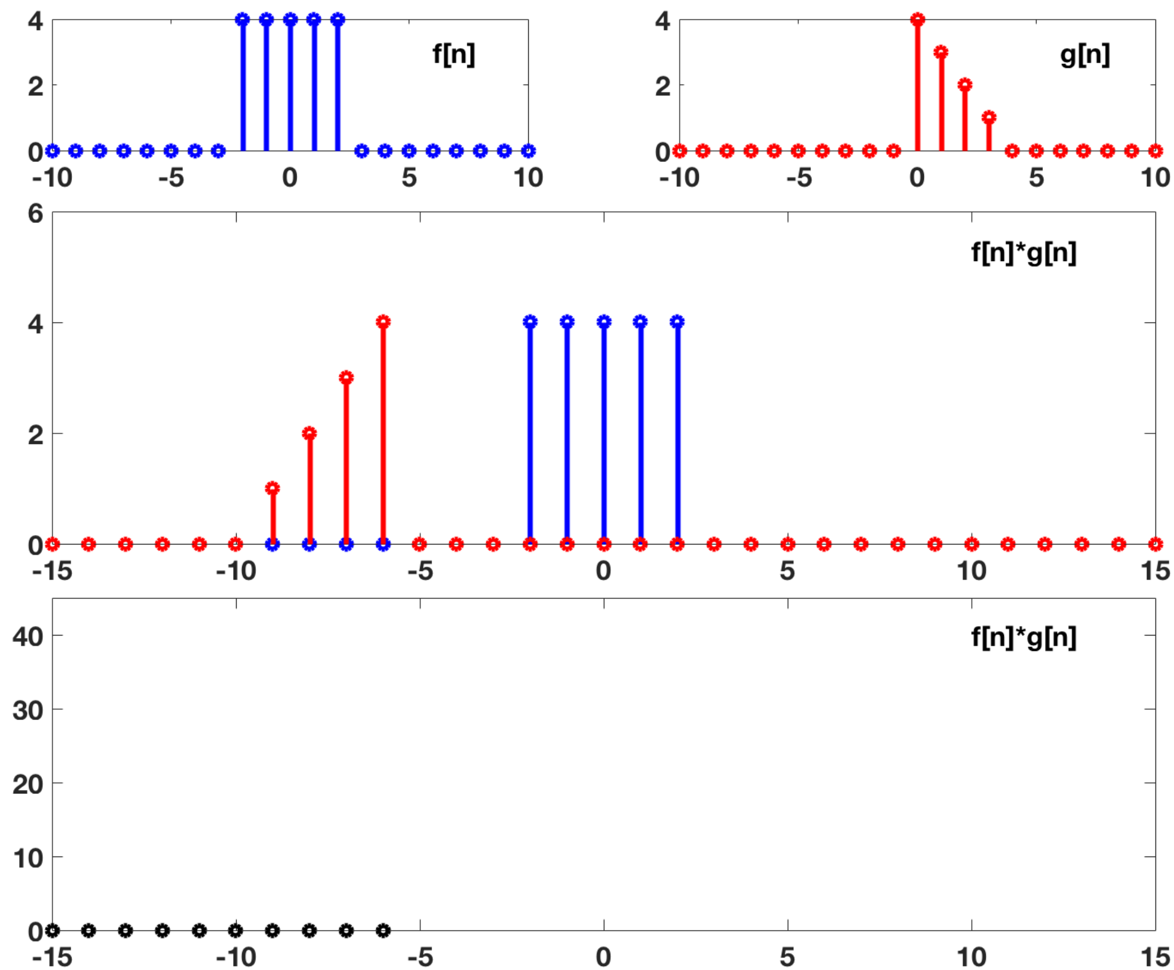
Another way of looking at it.

- The math of convolution is defined as flipping one of the signals in time and then moving it across the other signal, multiplying and summing (go back and look at that Python code).

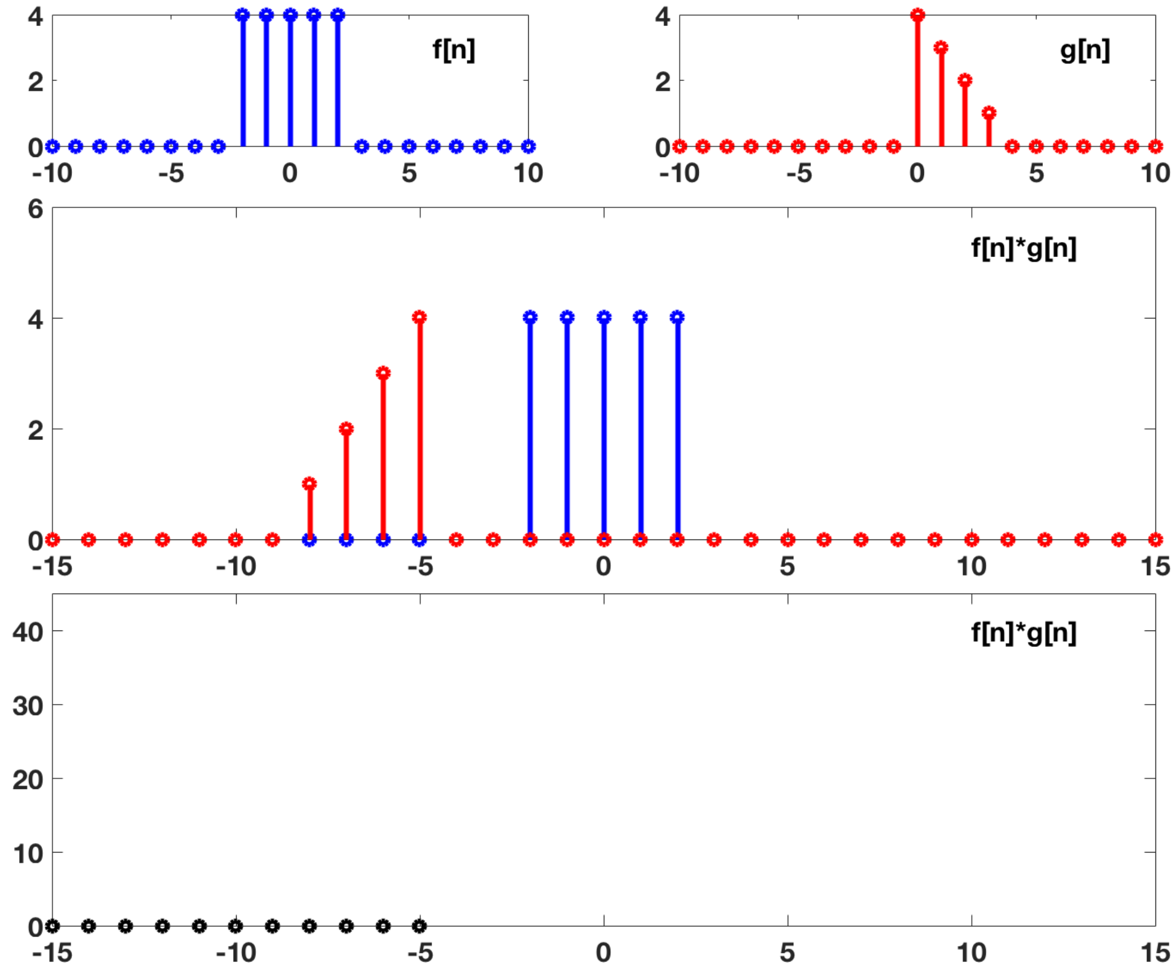
- On the next slides, you'll see convolution worked through as describe above.

- Note that the end result is the same as what you saw in the previous demo.

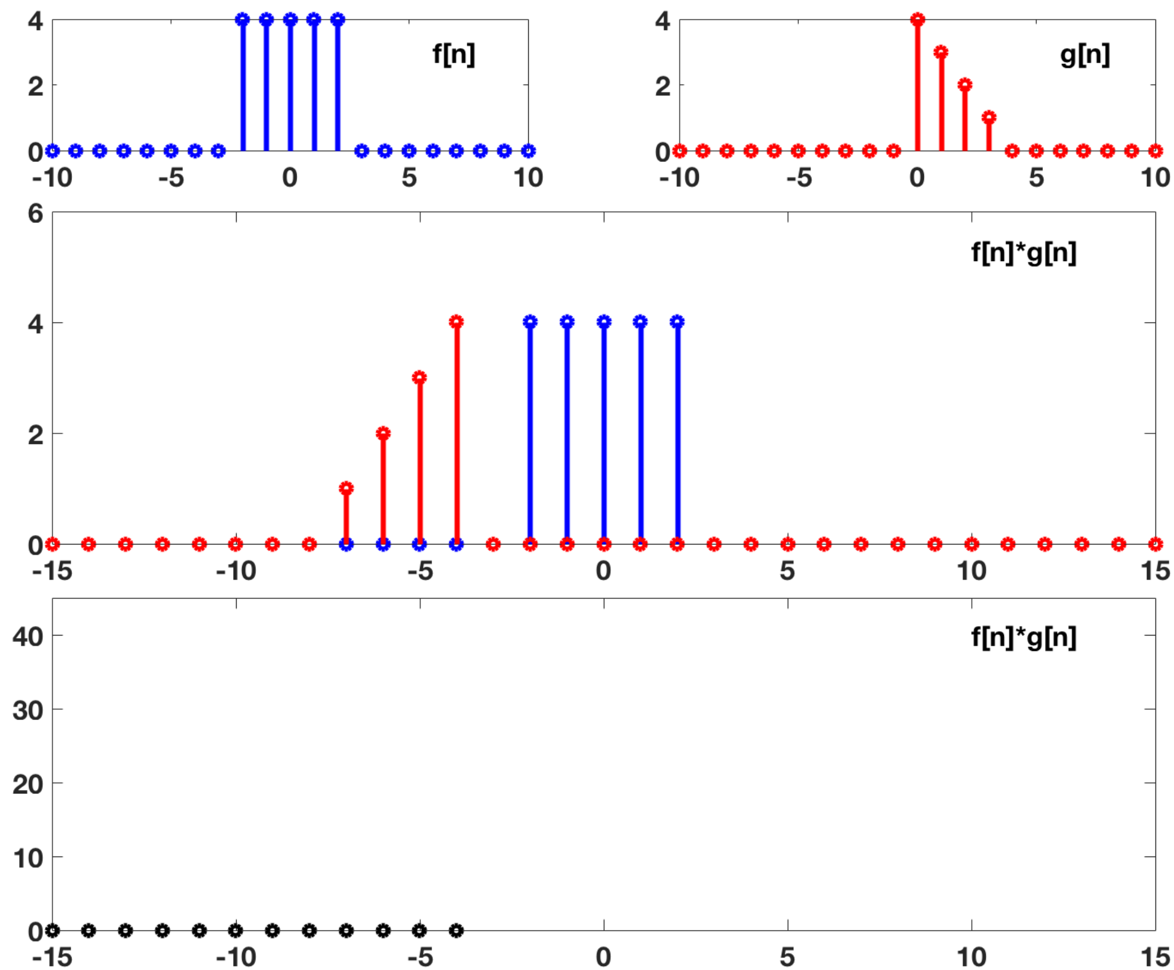
Let's look at Convolution



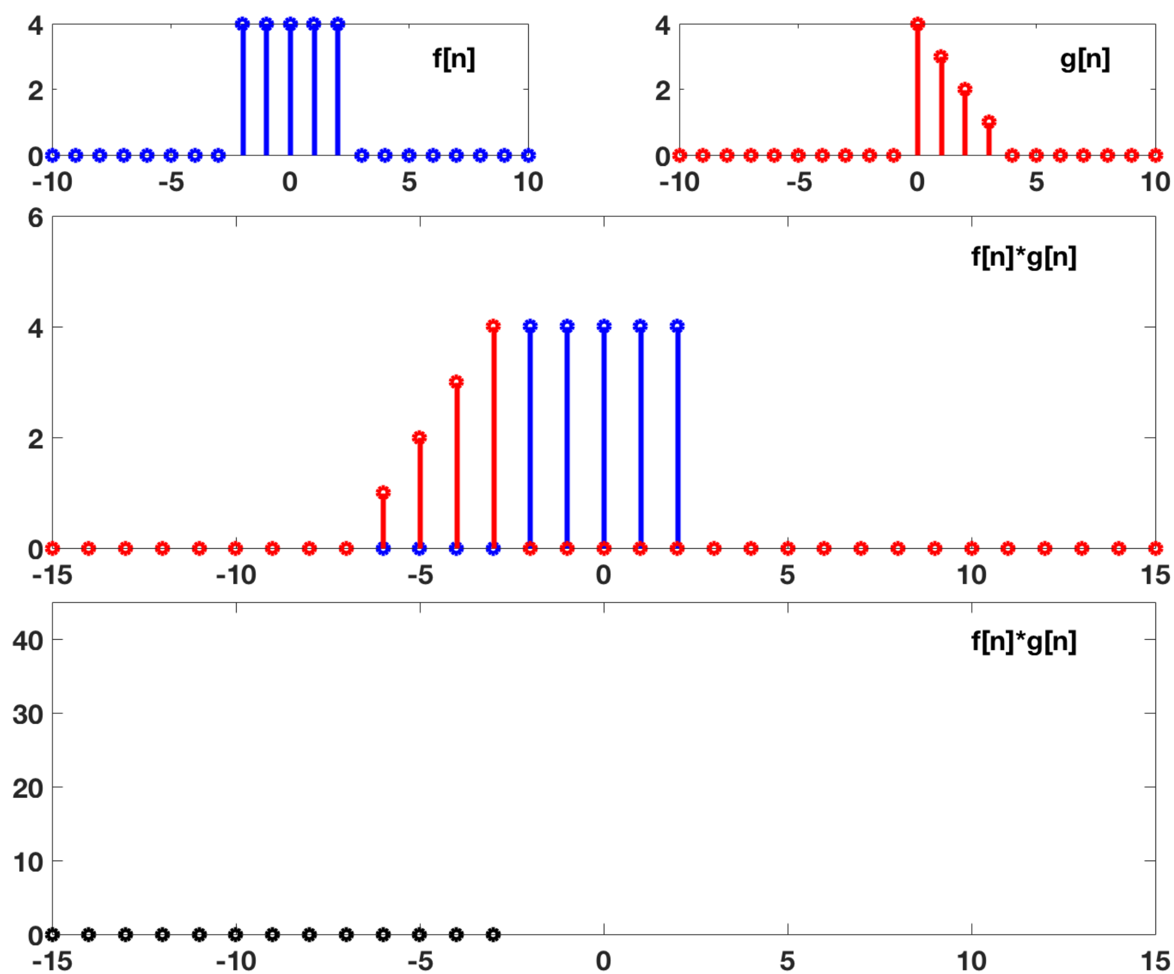
Let's look at Convolution



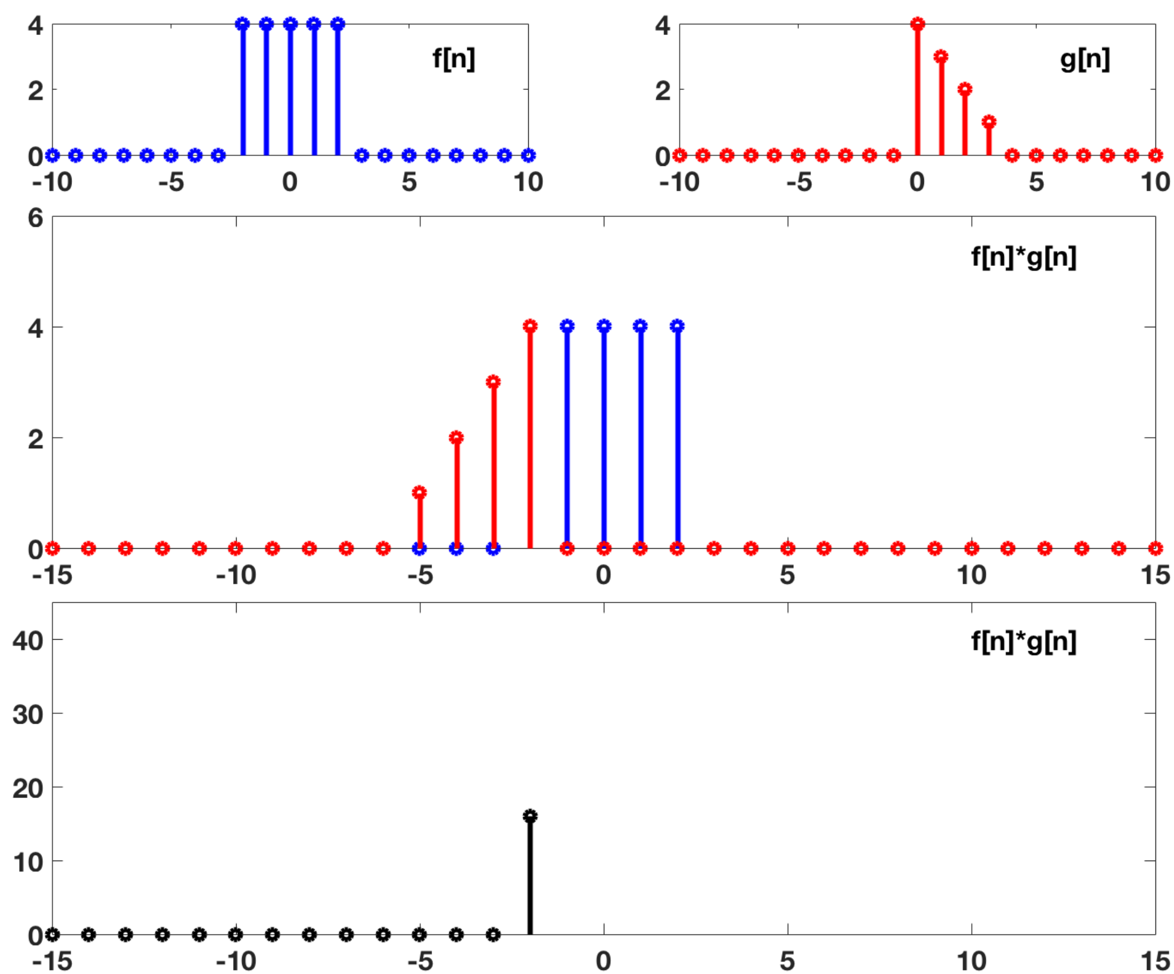
Let’s look at Convolution



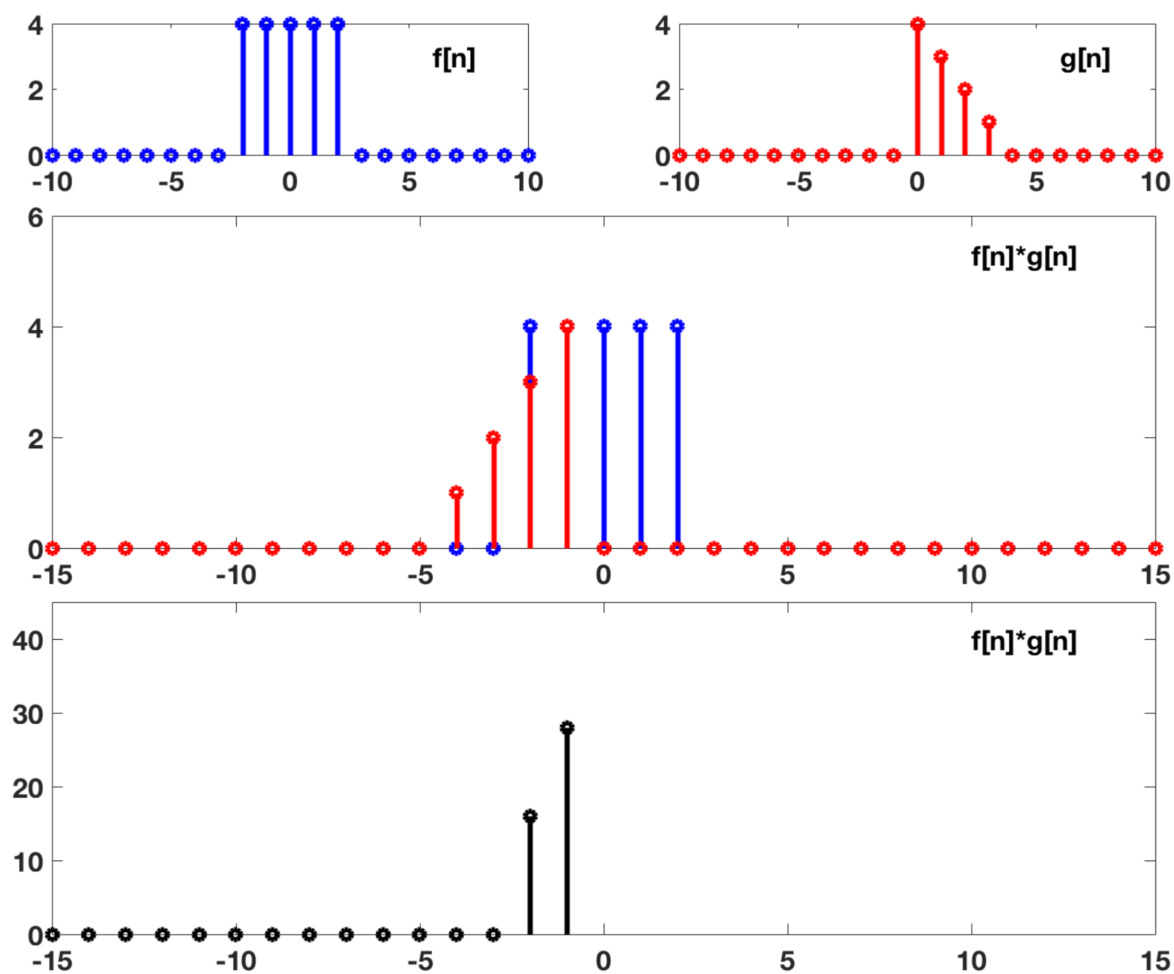
Let's look at Convolution



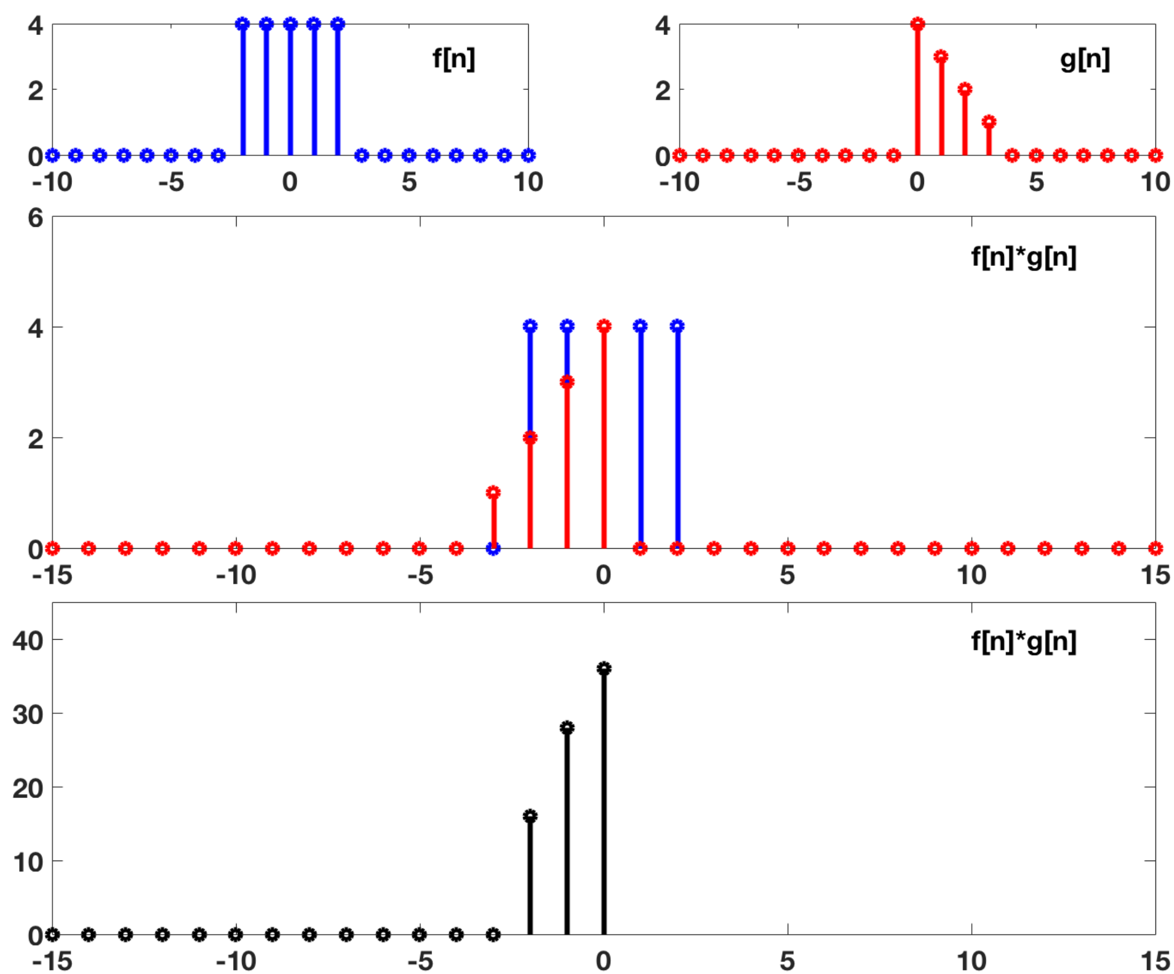
Let's look at Convolution



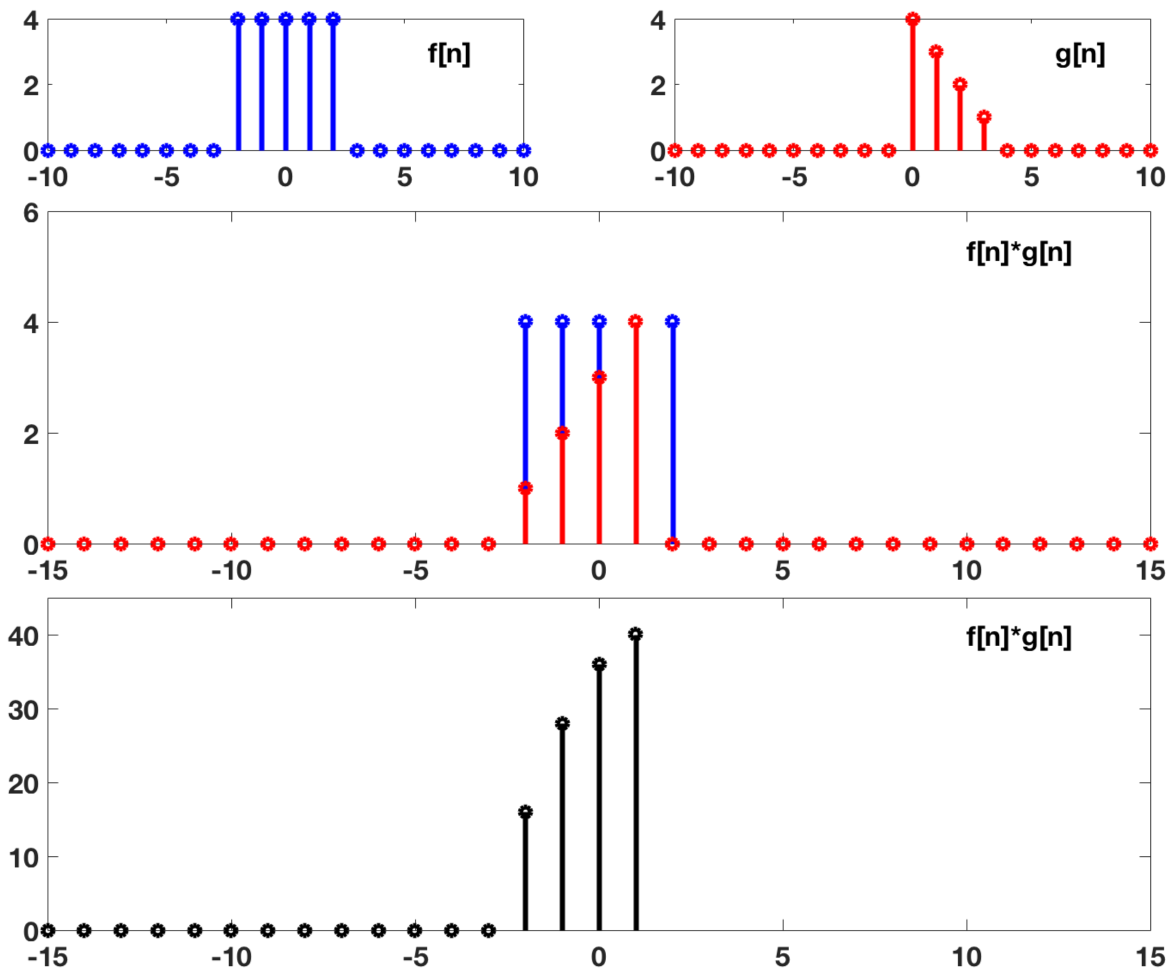
Let’s look at Convolution



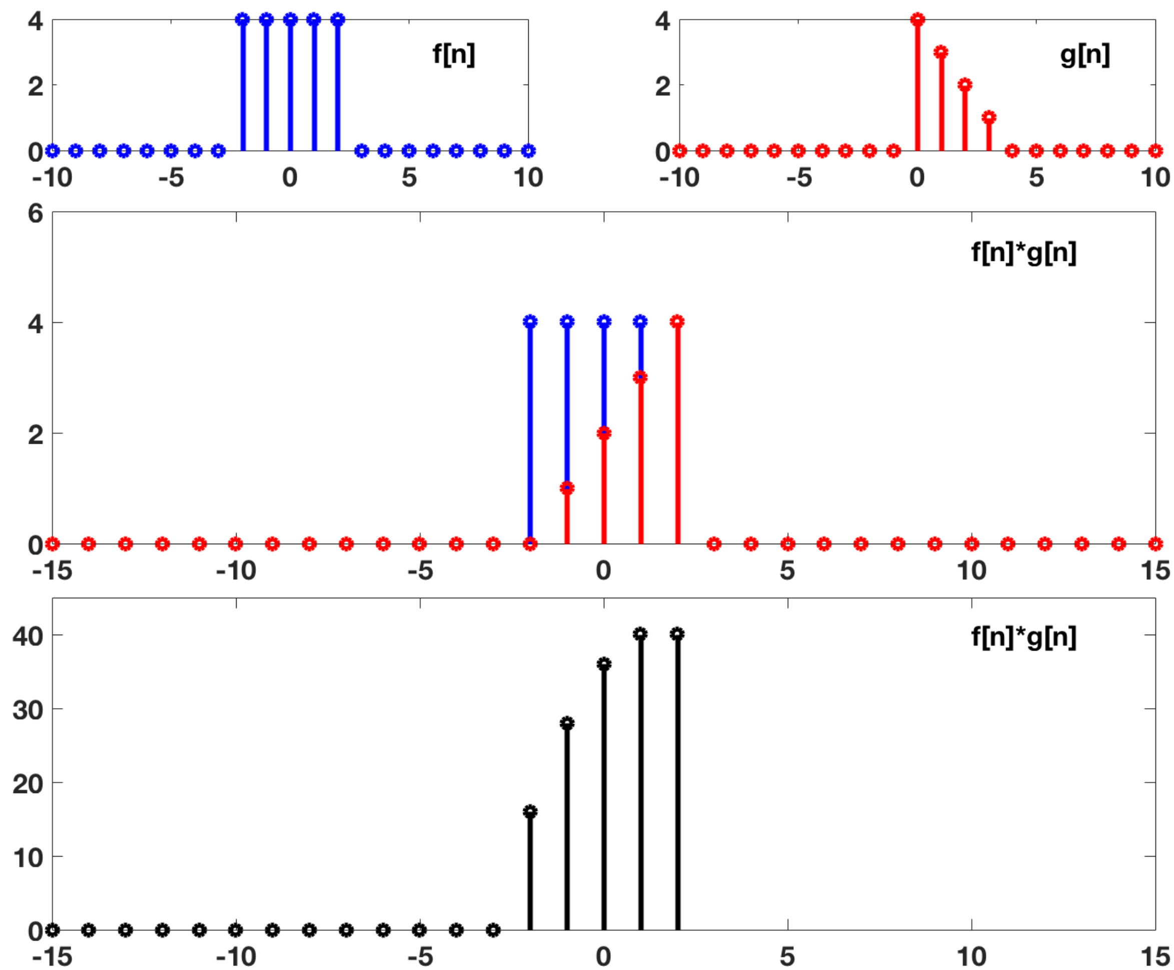
Let’s look at Convolution



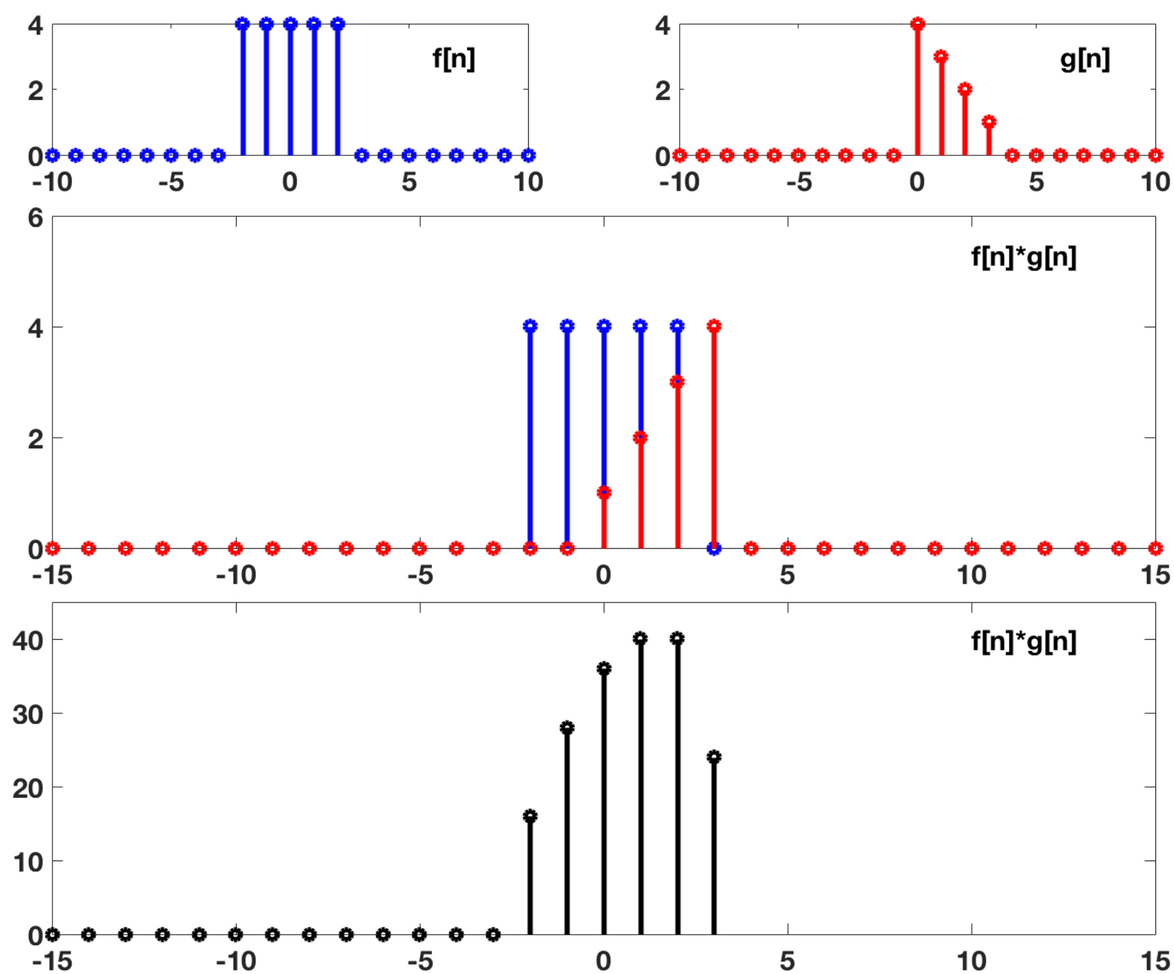
Let's look at Convolution



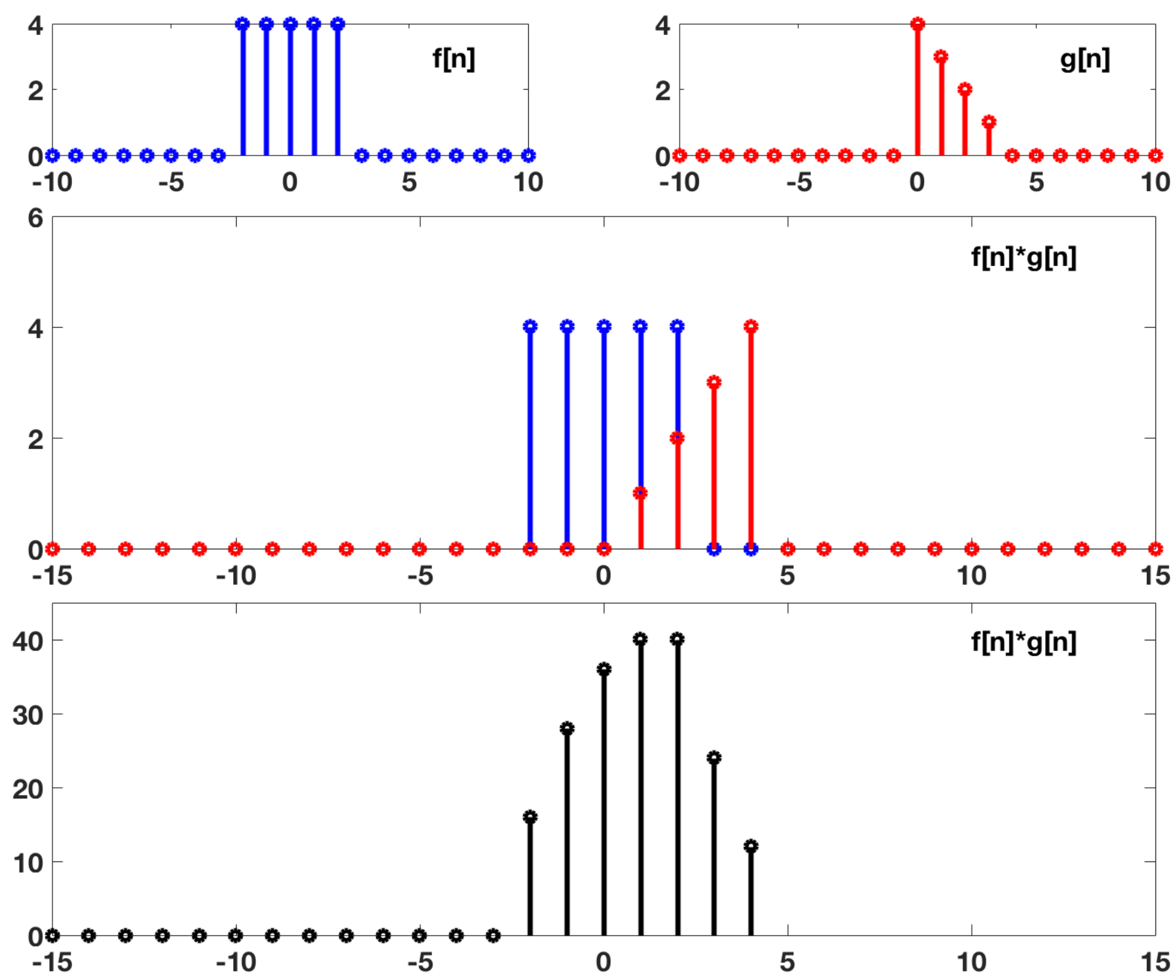
Let's look at Convolution



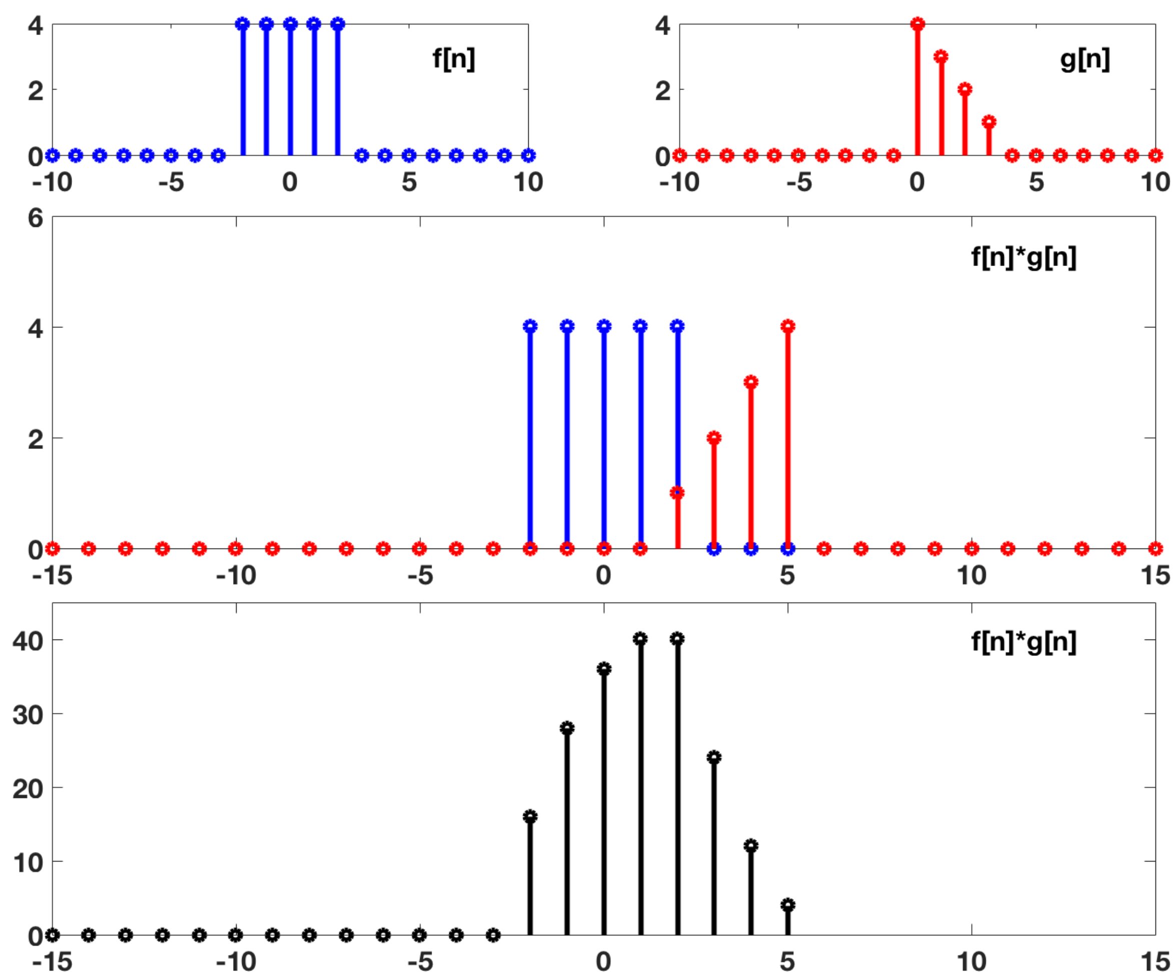
Let’s look at Convolution



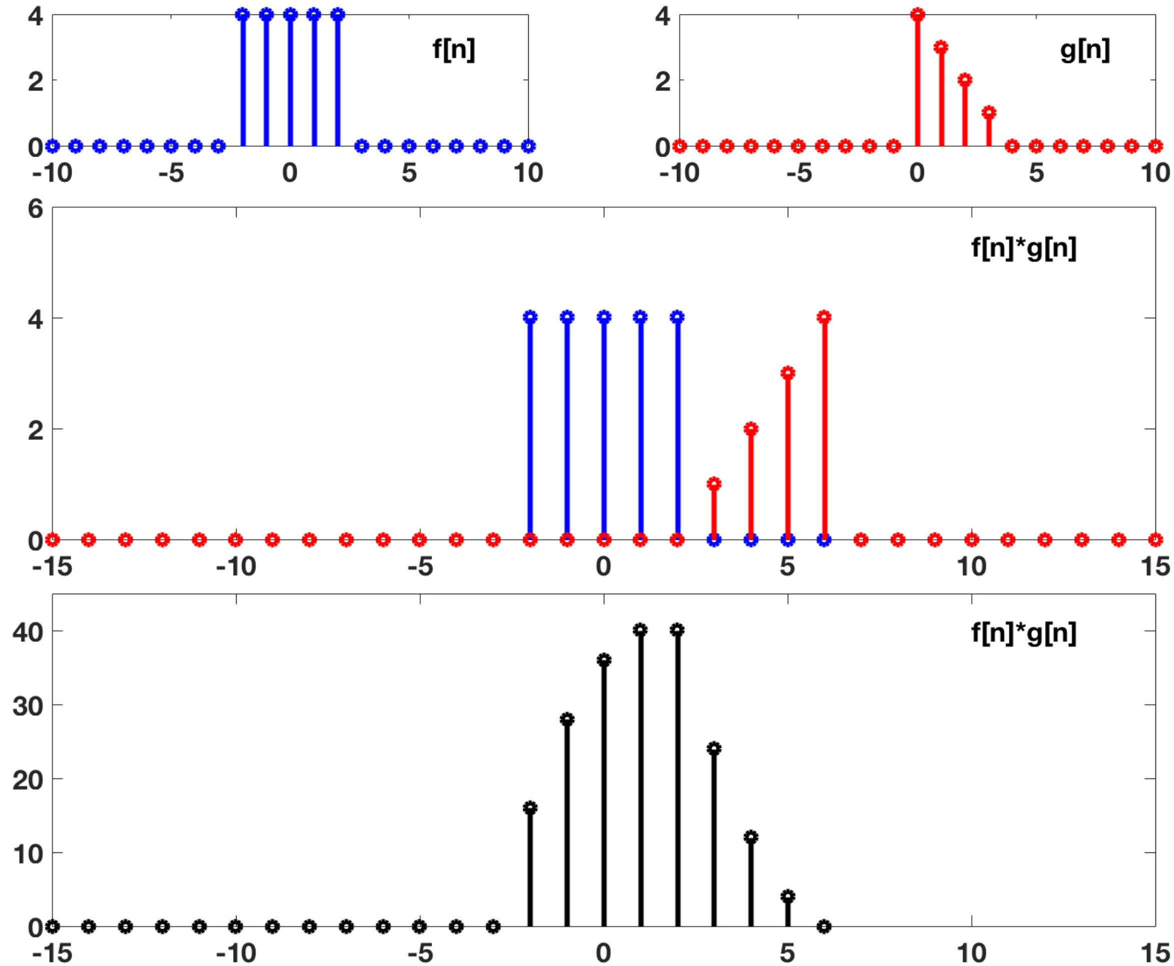
Let's look at Convolution



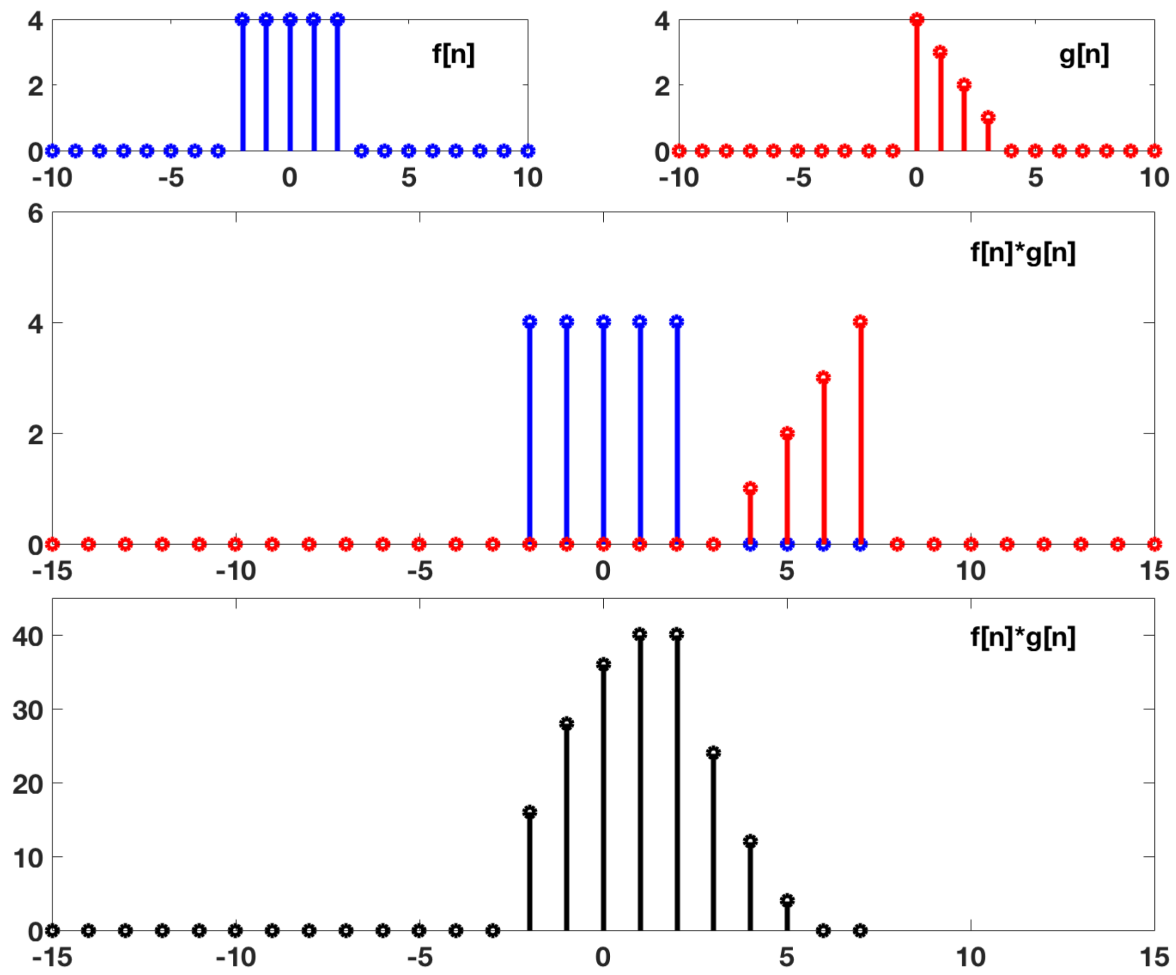
Let's look at Convolution



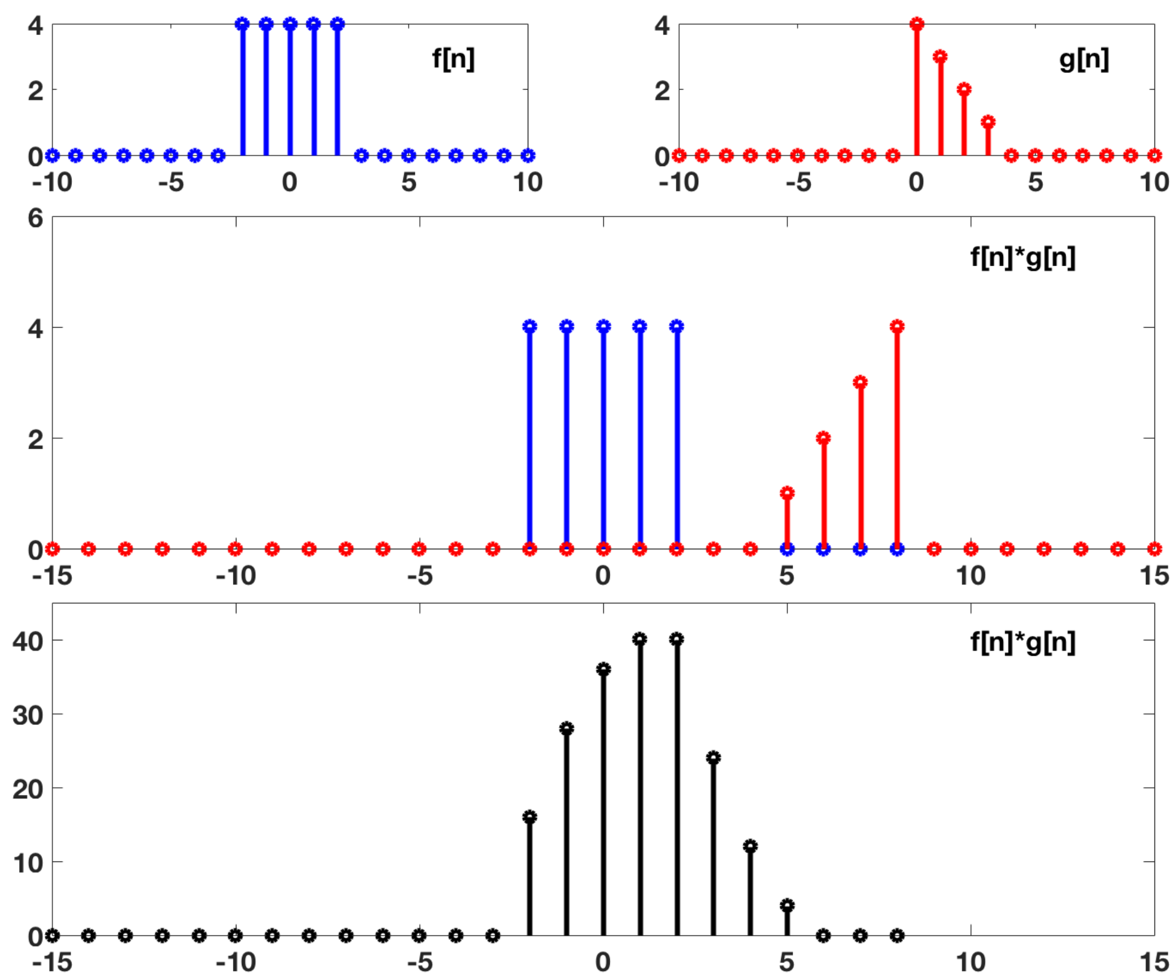
Let’s look at Convolution



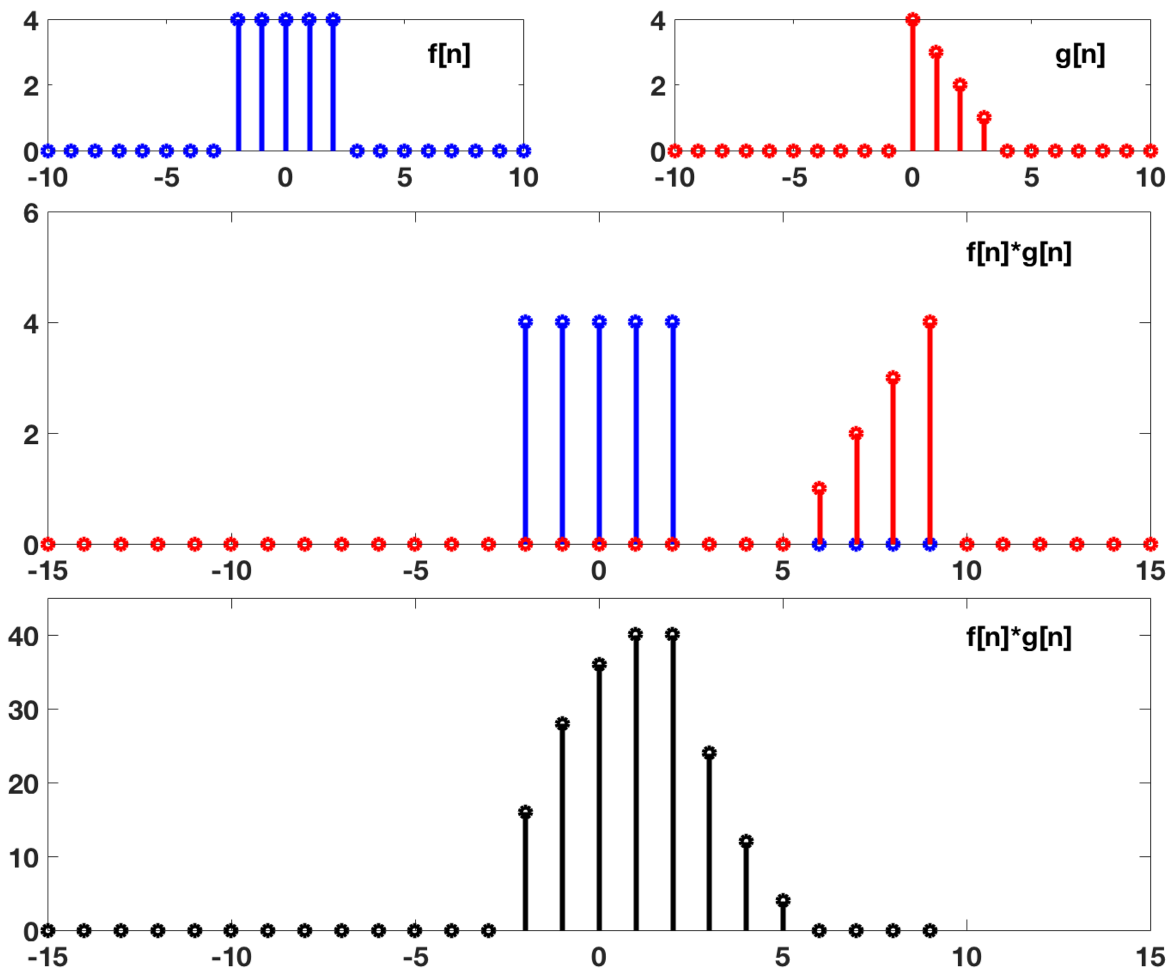
Let's look at Convolution



Let’s look at Convolution



Let’s look at Convolution



Cross-correlation

- **Cross-correlation** is a measure of similarity of two functions at time-lag ***t*** applied to one of them. It is a LOT like convolution…

Means “complex conjugate of h

$$(h \;\text{}\; \mathrm{x})(t) \equiv \int_{-\infty}^{\infty} h^{*}(\tau)\, x(t+\tau)\, d\tau$$

Cross-correlation operator
Should be a star
Couldn’t find “star” in my font

VERY Similar

- Convolution

$$(h * x)(t) \equiv \int_{-\infty}^{\infty} h(\tau)\, x(t-\tau)\, d\tau$$

- Cross-correlation

$$(h \star \mathrm{x})(t) \equiv \int_{-\infty}^{\infty} h^{*}(\tau)\, x(t+\tau)\, d\tau$$

Cross-correlation in Python Code

We can easily implement cross correlation with convolution as follows:

```
def crosscorrelation(A,B):
    return convolution(np.conj(A),B[::-1])
```

Better yet, use the built in Python functions…

```
np.convolve(A,B,"full")   # for convolution
np.correlate(A,B,"full")  # for cross correlation
```

Auto-correlation

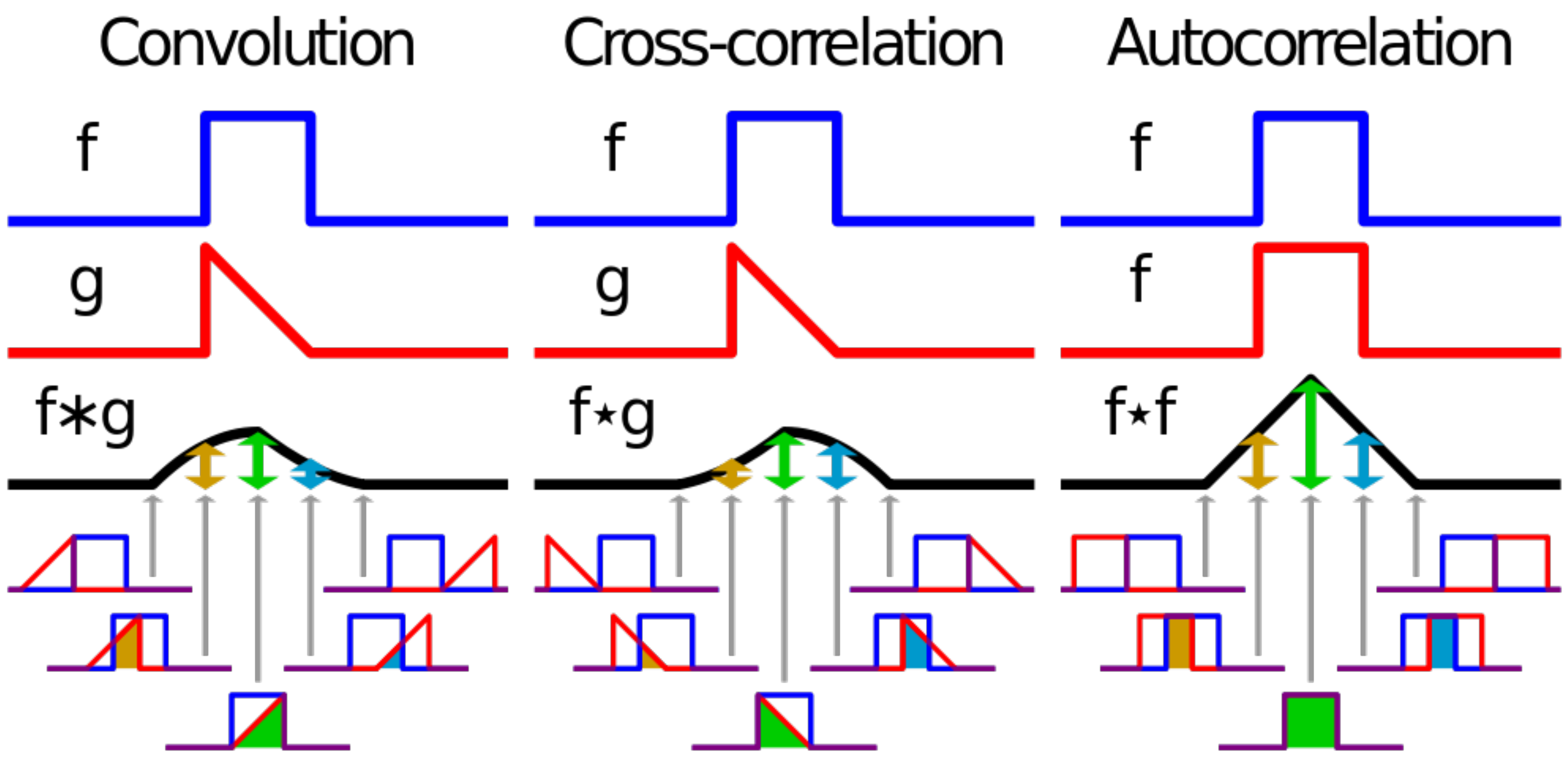
- **Auto-correlation** is a measure of similarity of a function to itself at time-lag ***t***. It is a special case of cross-correlation (cross-correlation of a function with itself).

Means "complex conjugate of f"

$$(x \;\text{}\; \mathrm{x})(t) \equiv \int_{-\infty}^{\infty} x^{*}(\tau)\, x(t+\tau)\, d\tau$$

Cross correlation
Should be a star
Couldn't find "star" in my font

Relating them all



Convolution and Fourier transform

- An important property of the Fourier transform: converts convolution in the time domain into multiplication in the frequency domain.

Convolution… $$y(t) = h(t) * x(t)$$

In the time domain: $$y(t) = \int h(\tau)x(t-\tau)\,d\tau$$

In frequency domain: $$Y(\omega) = H(\omega)X(\omega)$$

Frequency selective filters

- Frequency selective filters we use in this course are a subset of LTI system.
- The output of filters will be computed via convolution in the time domain, or equivalently, via multiplication in the frequency domain.
- Can you name a real world example of convolution as summing up attenuated and delayed copies of a signal?