Для доказательства эквивалентности интегральной и дискретной свертки рассмотрим произвольную непрерывную функцию $f(t)$ и её дискретное представление $f[n]$, где n - целочисленное значение. Предположим также, что у нас есть непрерывная функция $g(t)$ и её дискретное представление $g[n]$. Мы хотим доказать, что интегральная и дискретная свертки $f(t)$ и $g(t)$ эквивалентны.

Интегральная свертка определяется следующим образом:
$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$

Дискретная свертка определяется следующим образом:
$(f * g)[n] = \sum_{k=-\infty}^{\infty} f[k]g[n-k]$

Для доказательства эквивалентности мы можем сначала дискретизировать интегральную свертку и затем продемонстрировать, что она равна дискретной свертке. Для этого давайте дискретизируем функцию $f(\tau)$ и $g(\tau)$ с шагом $\Delta\tau$, где $\tau = k\Delta\tau$, а также заменим интеграл на сумму. Получим:

$(f * g)(t) = \lim_{\Delta\tau \to 0} \sum_{k=-\infty}^{\infty} f(k\Delta\tau)g(t-k\Delta\tau)\Delta\tau$

Это похоже на определение дискретной свертки. Поскольку $\Delta\tau \to 0$, то $\Delta\tau = \frac{1}{N}$, где N - количество точек дискретизации. Таким образом, можно записать:

$(f * g)(t) = \lim_{N \to \infty} \sum_{k=-\infty}^{\infty} f\left(\frac{k}{N}\right) g\left(t-\frac{k}{N}\right) \frac{1}{N}$

По предельному переходу получаем:

$(f * g)(t) = \sum_{n=-\infty}^{\infty} f\left(\frac{n}{N}\right) g\left(t-\frac{n}{N}\right)$

Это идентично дискретной свертке. Таким образом, мы доказали эквивалентность интегральной и дискретной сверток.

Что такое свертка и для чего она нужна?
Свертка это способ объединить два временных ряда (или изображения для двумерных массивов) чтобы создать третий временной ряд.

Сигнал -> Временной ряд, который необходимо обработать.
Ядро -> Фильтр.
Результат свертки: смесь функций сигнала и ядра.

Таким образом на вход “Свертки” подаются два сигнала – непосредственно сигнал и ядро свертки.
Такой подход подобен способу передачи на вход “Фичей” сигнала и свертки для выделения ключевых характеристик сигнала.

Ключевым моментом свертки является идея о том, что у вас есть сигнал и ядро. При этом ядро может быть использовано как фильтр.

Для доказательства эквивалентности интегральной и дискретной свертки рассмотрим интегральную свертку функций $f(x)$ и $g(x)$ на интервале $[a, b]$ и дискретную свертку их дискретных аналогов $f[n]$ и $g[n]$ на дискретном множестве индексов n.

Интегральная свертка определяется как:
$(f * g)(x) = \int_a^b f(t) \cdot g(x - t)\, dt$

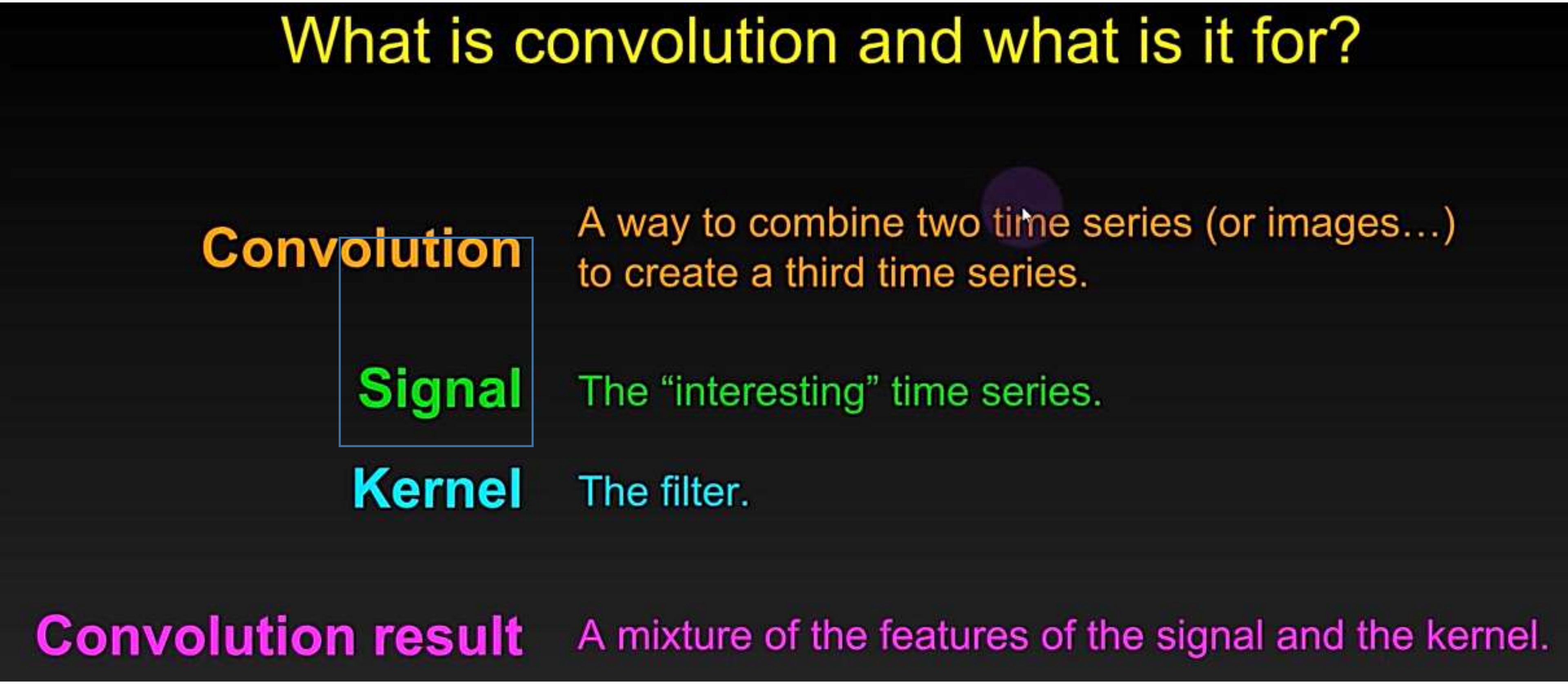
Дискретная свертка определяется как:
$(f * g)[n] = \sum_k f[k] \cdot g[n - k]$

Для доказательства эквивалентности мы можем применить дискретизацию к интегральной свертке. Предположим, что интервал $[a, b]$ разбивается на N равных частей, и обозначим шаг дискретизации как $\Delta x = \frac{b-a}{N}$. Тогда мы можем аппроксимировать функции $f(x)$ и $g(x)$ дискретными последовательностями $f[n]$ и $g[n]$, где $n = 0, 1, 2, \ldots, N - 1$, используя значения функций на равномерно распределенных точках $x_i = a + i \cdot \Delta x$.

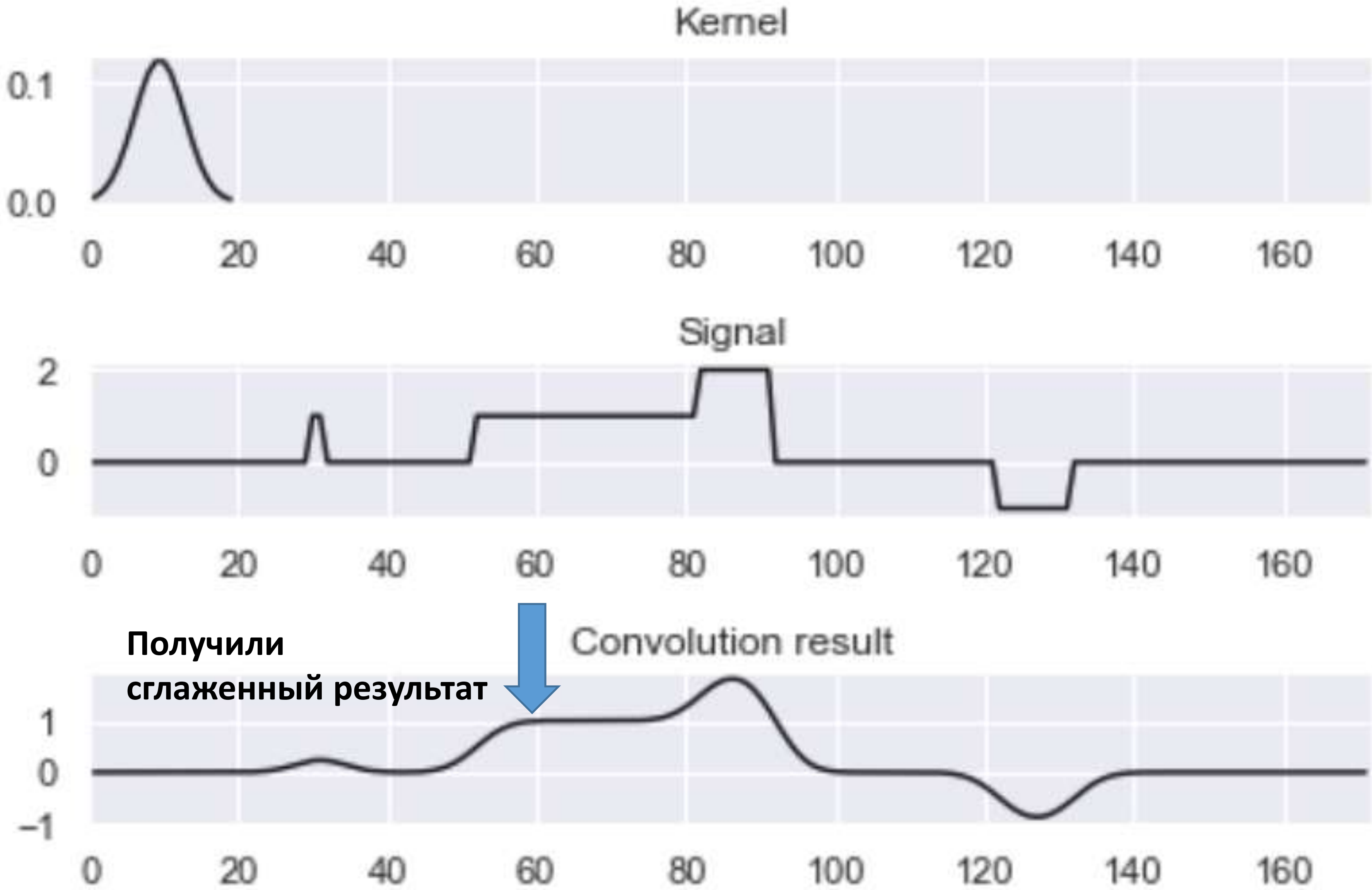
Теперь мы можем переписать интегральную свертку как сумму Римана:
$(f * g)(x) \approx \sum_{i=0}^{N-1} f(x_i) \cdot g(x - x_i) \cdot \Delta x$

Используя свойство аппроксимации интеграла суммой Римана при бесконечно малом Δx, мы видим, что выражение для интегральной свертки становится похожим на выражение для дискретной свертки.

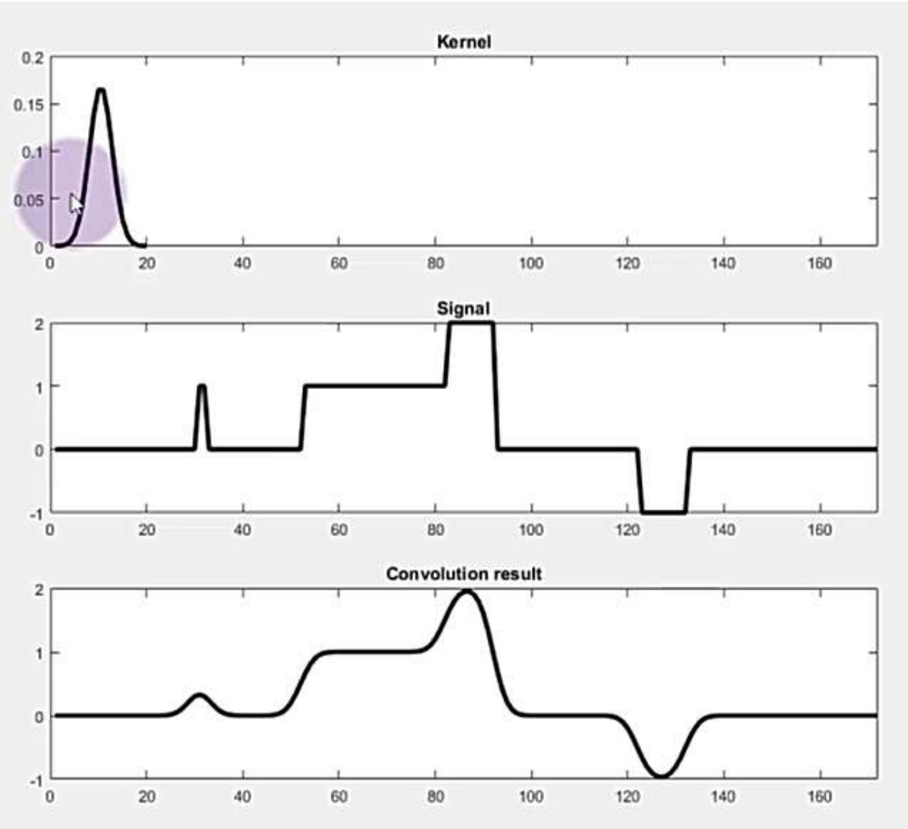
Таким образом, мы показали, что при дискретизации функций и аппроксимации интеграла суммой Римана интегральная и дискретная свертки становятся эквивалентными.

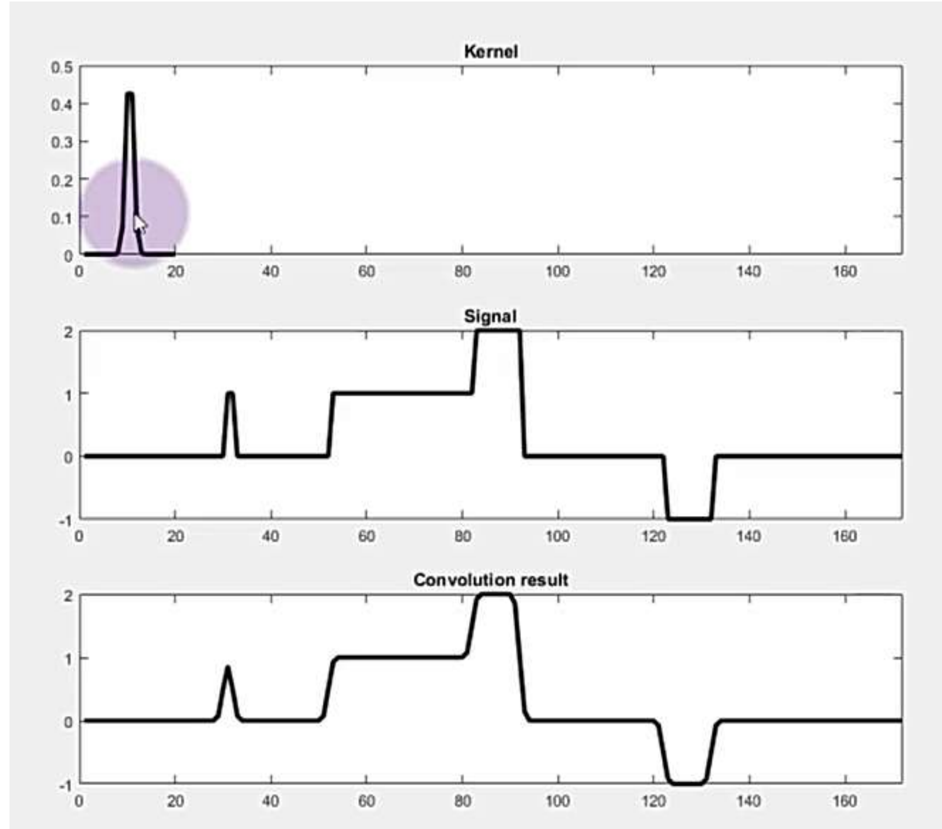

What is convolution and what is it for?
Convolution
A way to combine two time series (or images…) to create a third time series.
Signal
The "interesting" time series.
Kernel
The filter.
Convolution result
A mixture of the features of the signal and the kernel.

Пример свертки во временном пространстве.

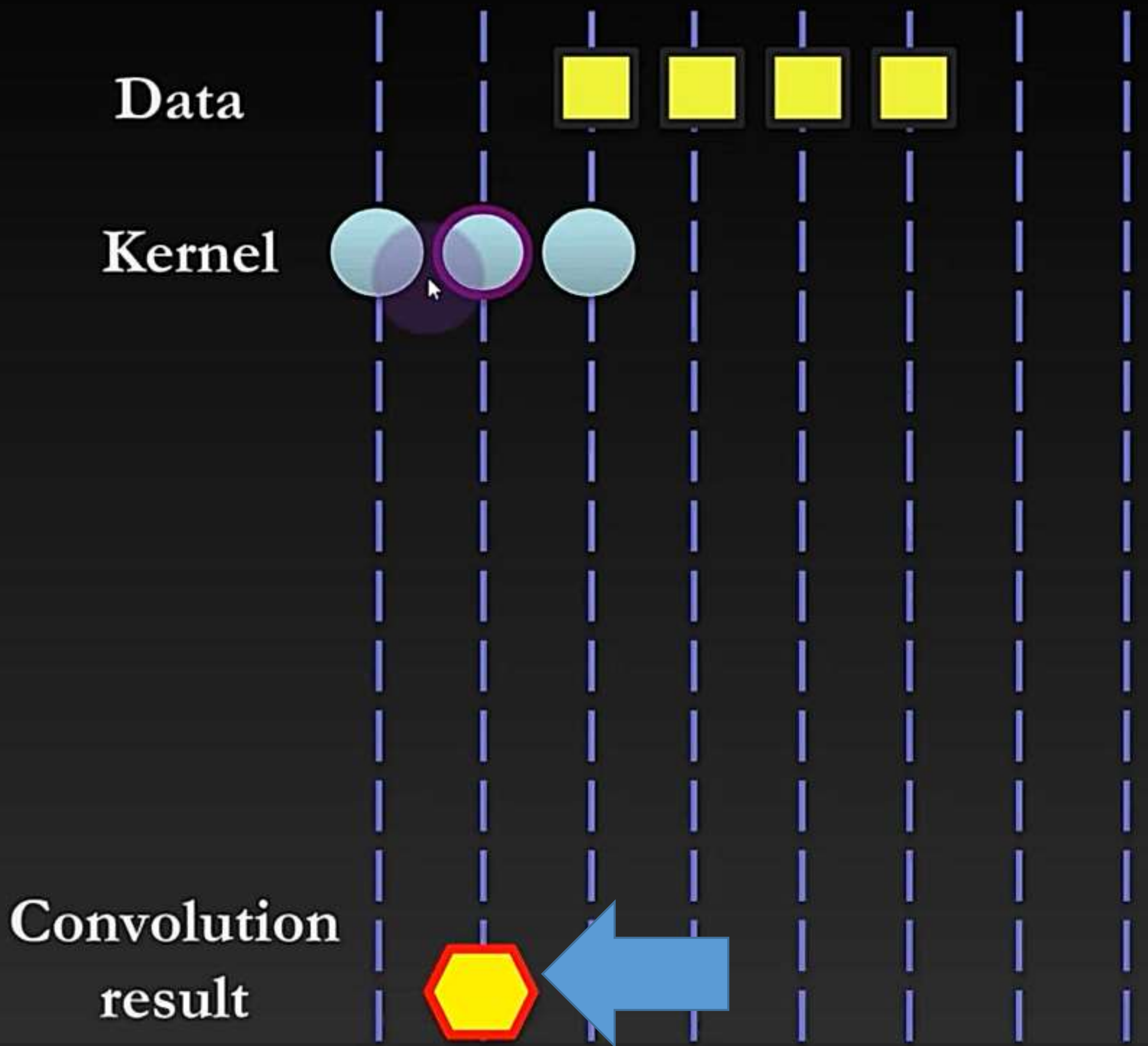


Применение различных “ядер”:

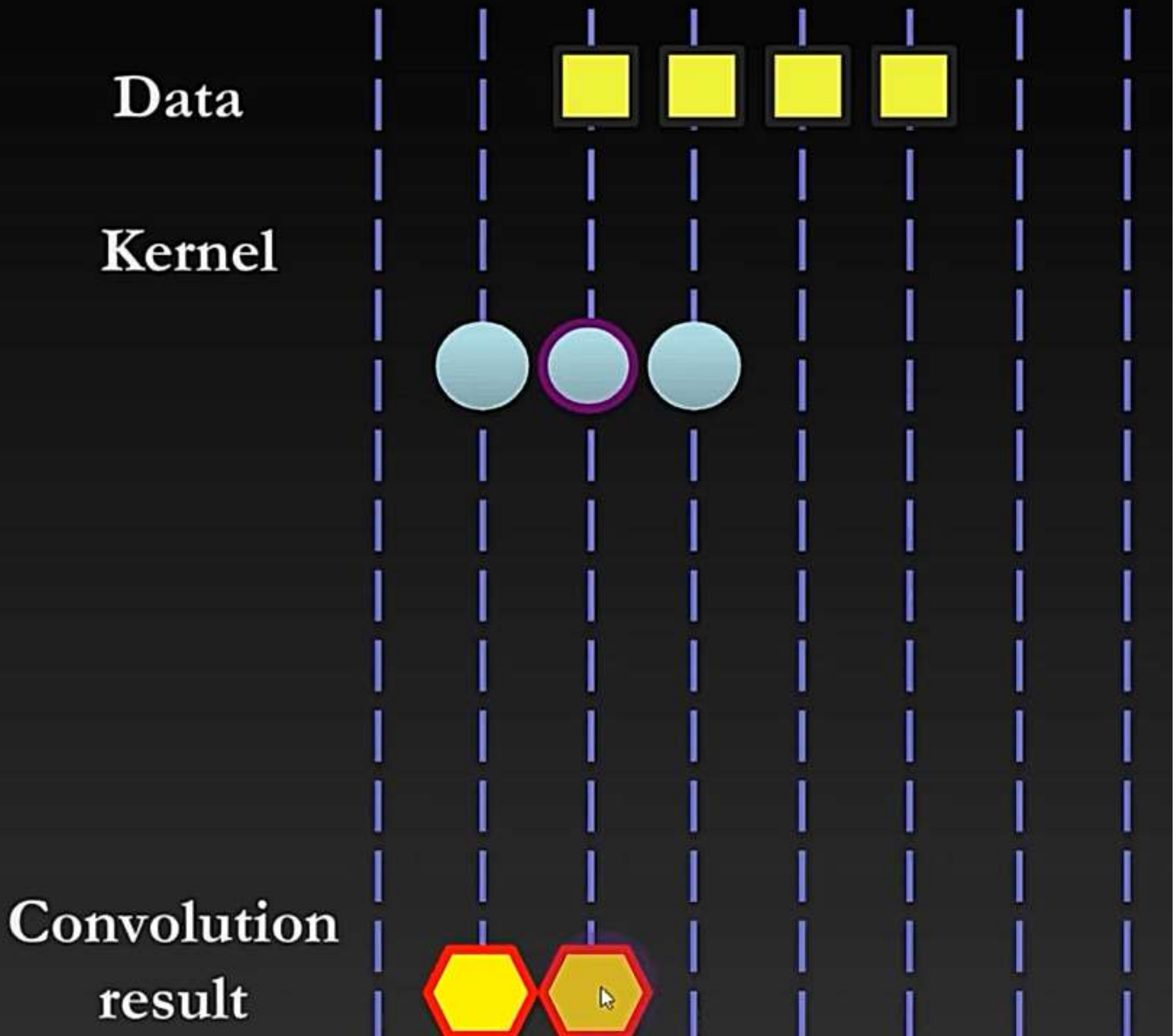




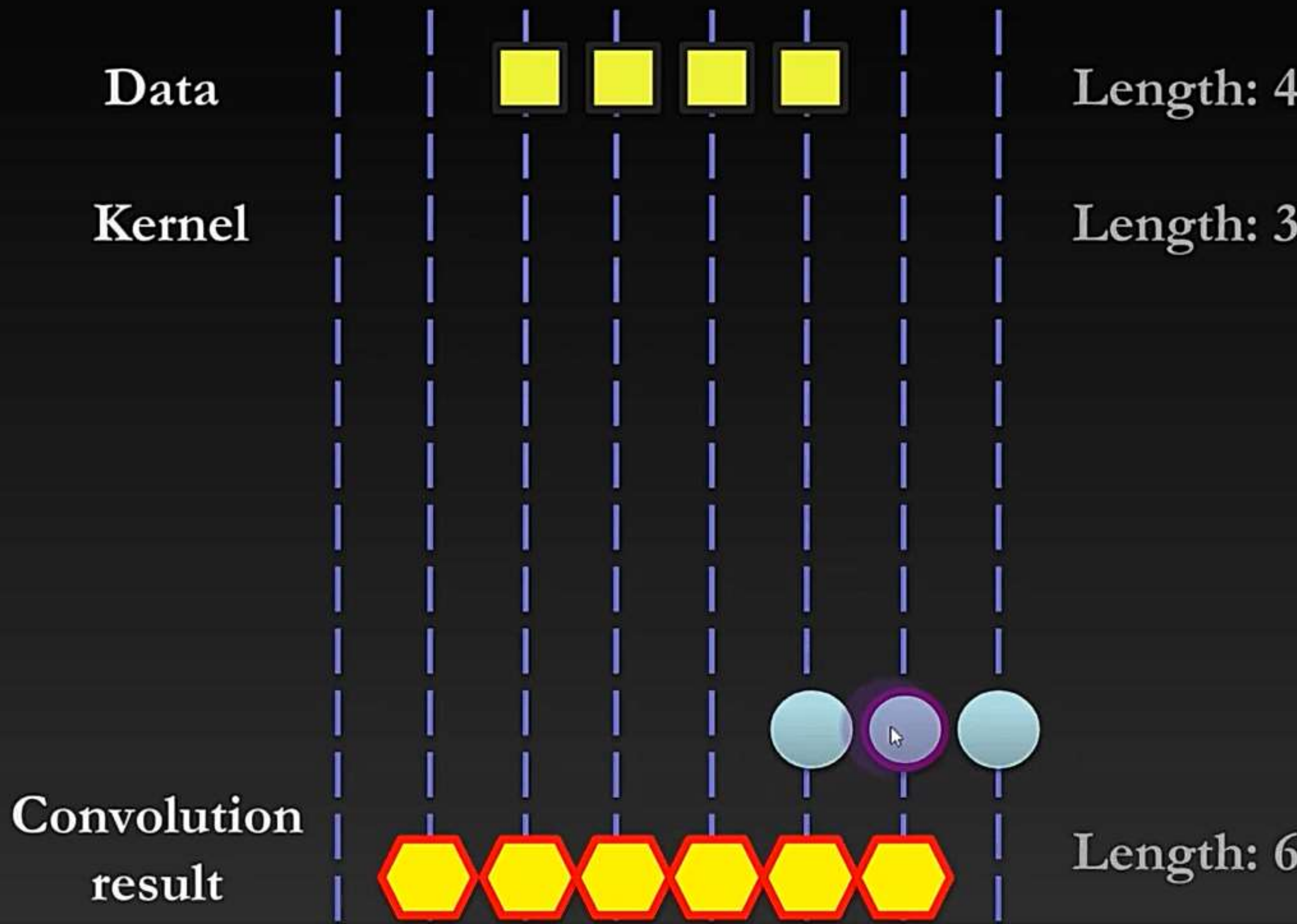
How convolution works



How convolution works



How convolution works

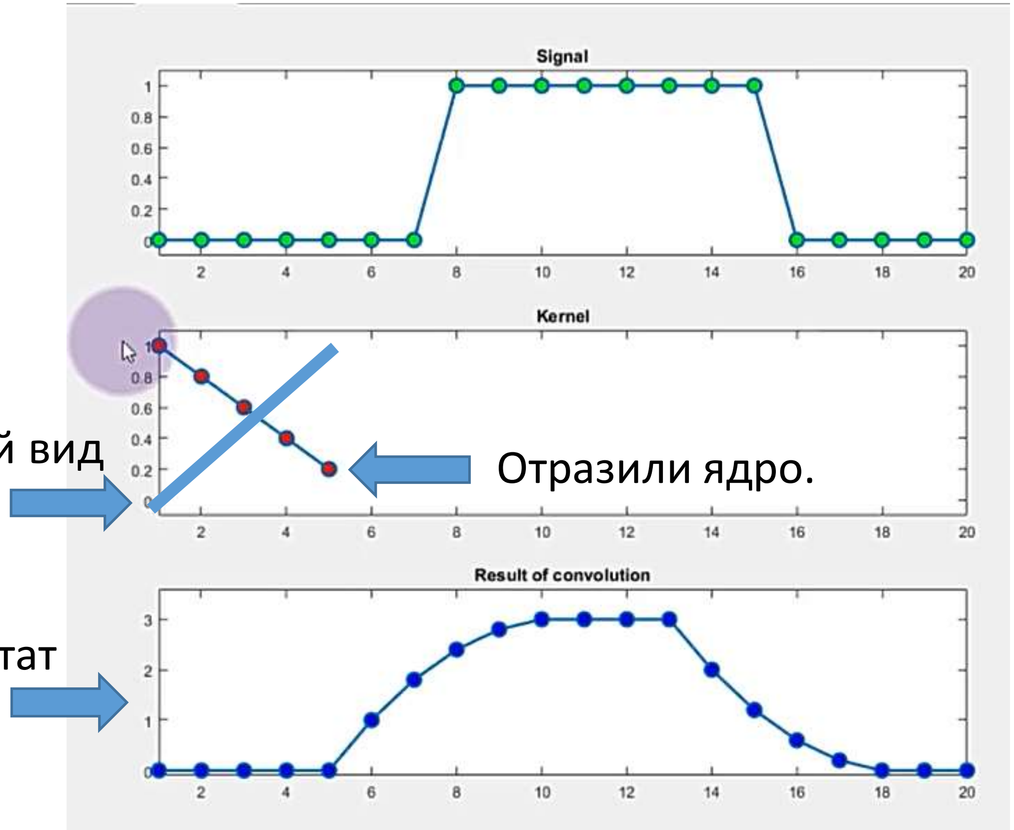


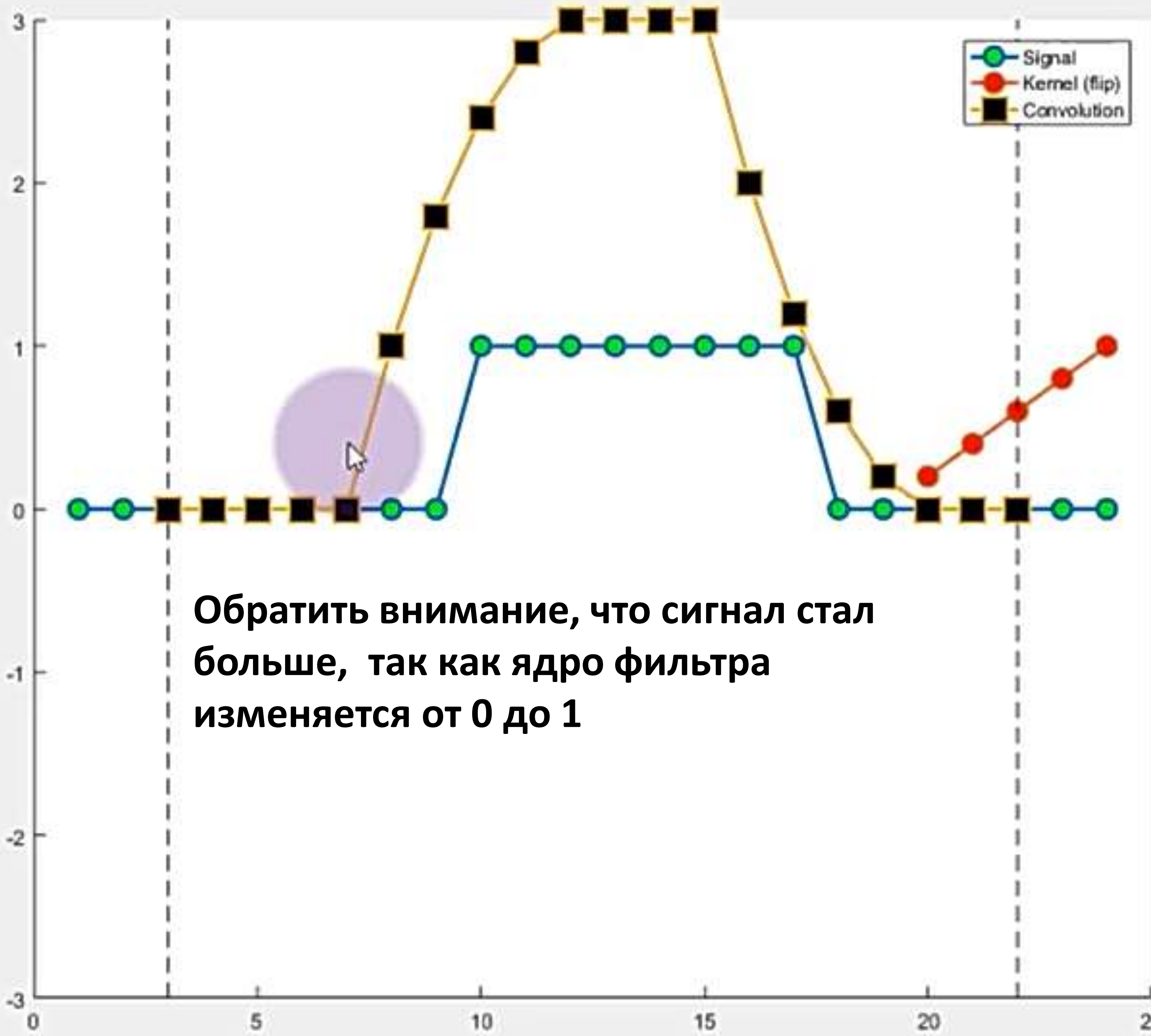
Есть одна дополнительная особенность свертки, которая заключается в том, что ядро переворачивается таким образом, чтобы первая точка была последней и т.д.

Ядро в первоначальном виде.

Ядро в свертке.

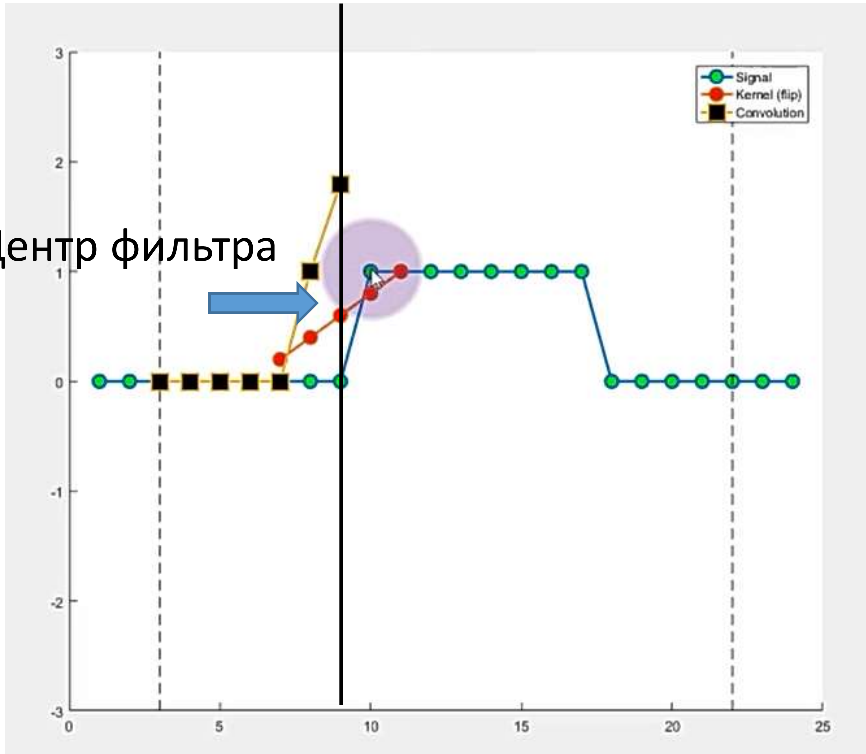
Ядро всегда отражено слева направо при выполнении свертки!

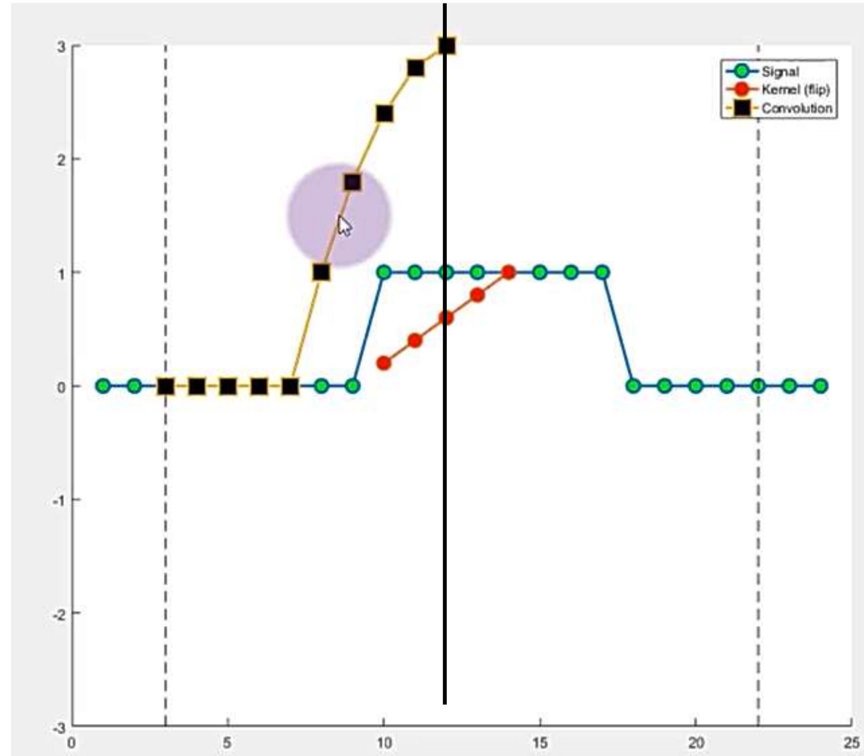




Обратить внимание, что сигнал стал больше, так как ядро фильтра изменяется от 0 до 1

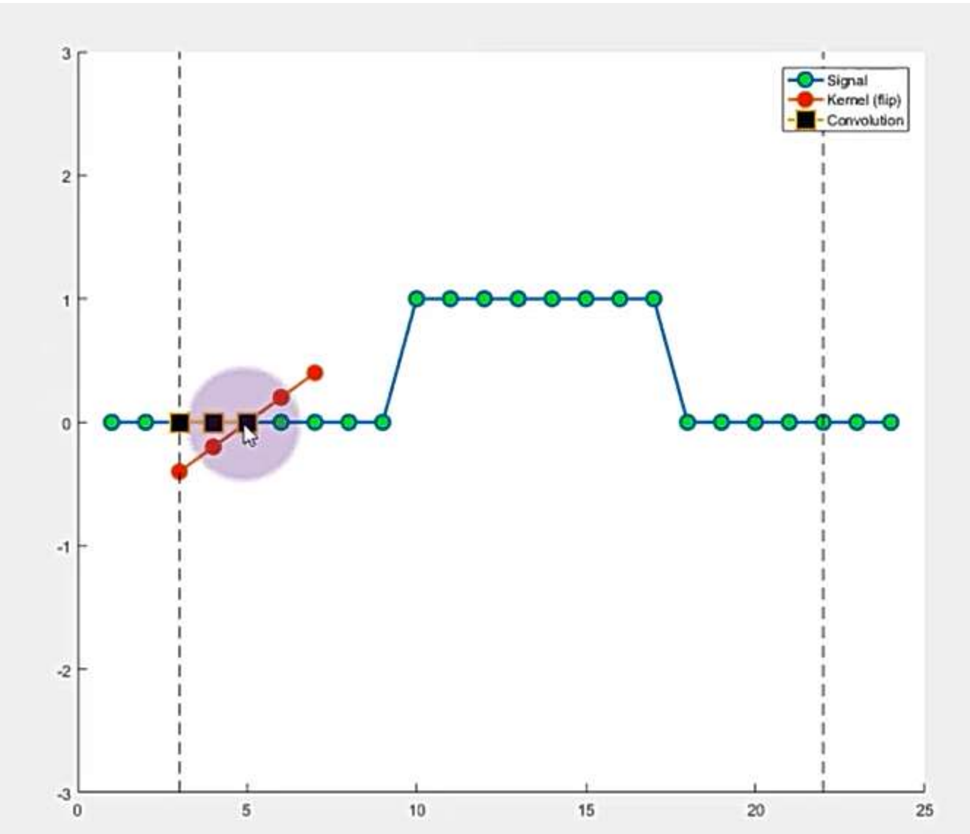
Центр фильтра

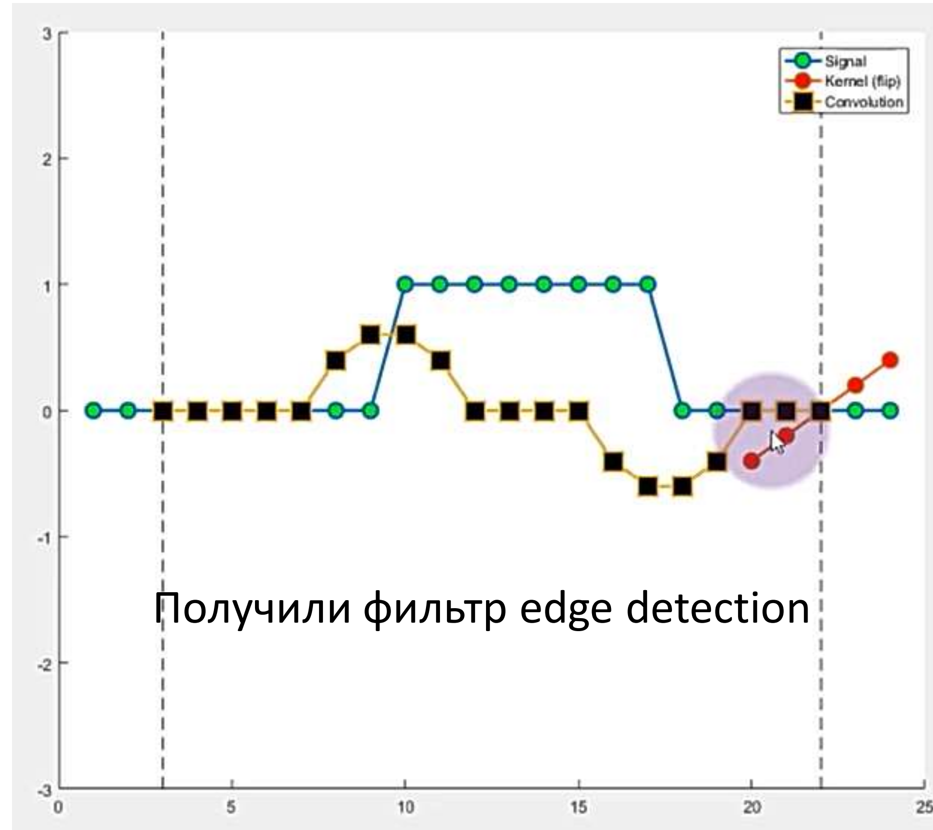




```
half_kern = floor(nKern/2);

% flipped version of kernel
kflip = kernel(end:-1:1)-mean(kernel);

% zero-padded data for convolution
dat4conv = [ zeros(1,half_kern) signal zeros(1,half_kern) ];
```

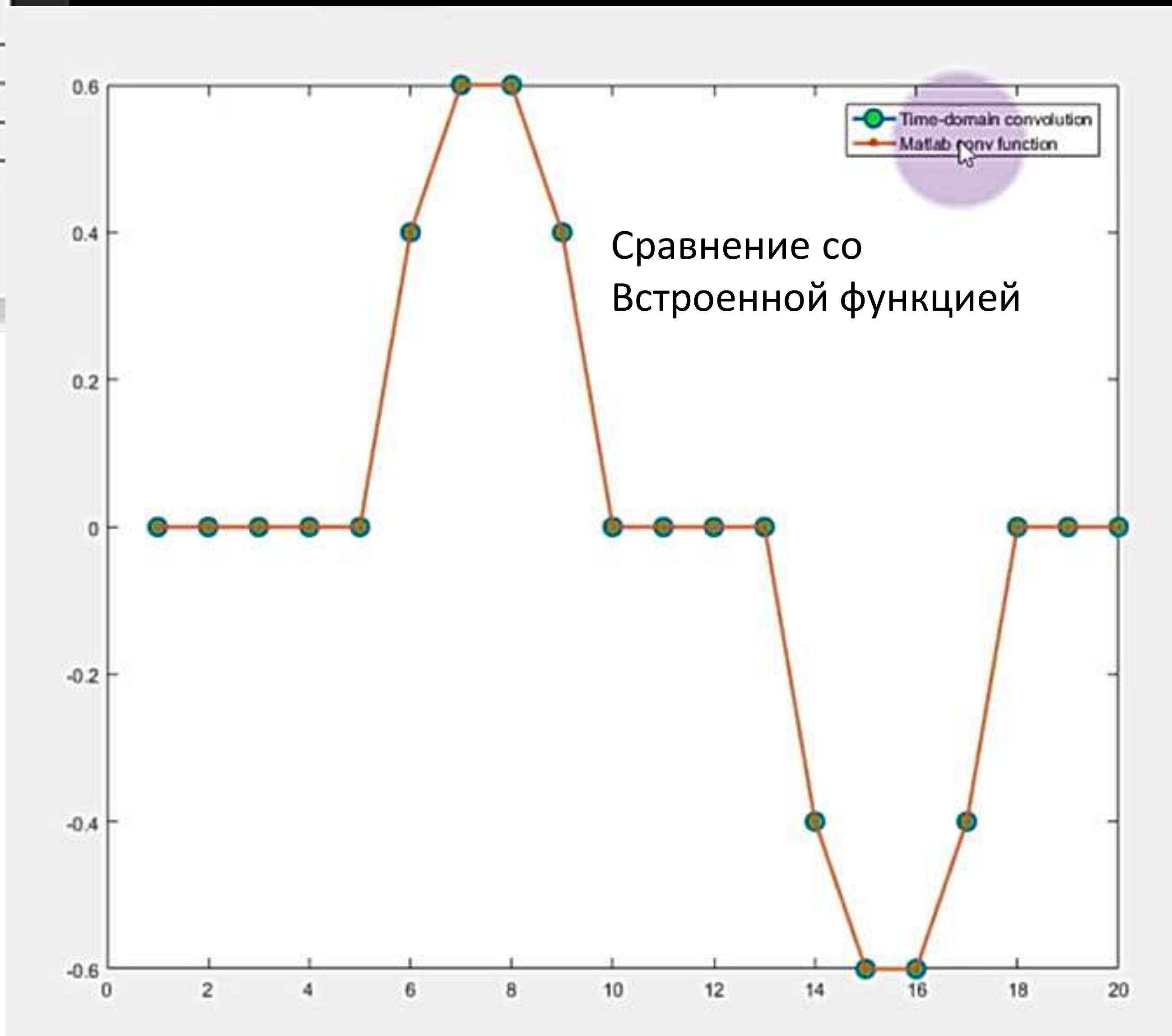
Отняли среднее значения для ядра



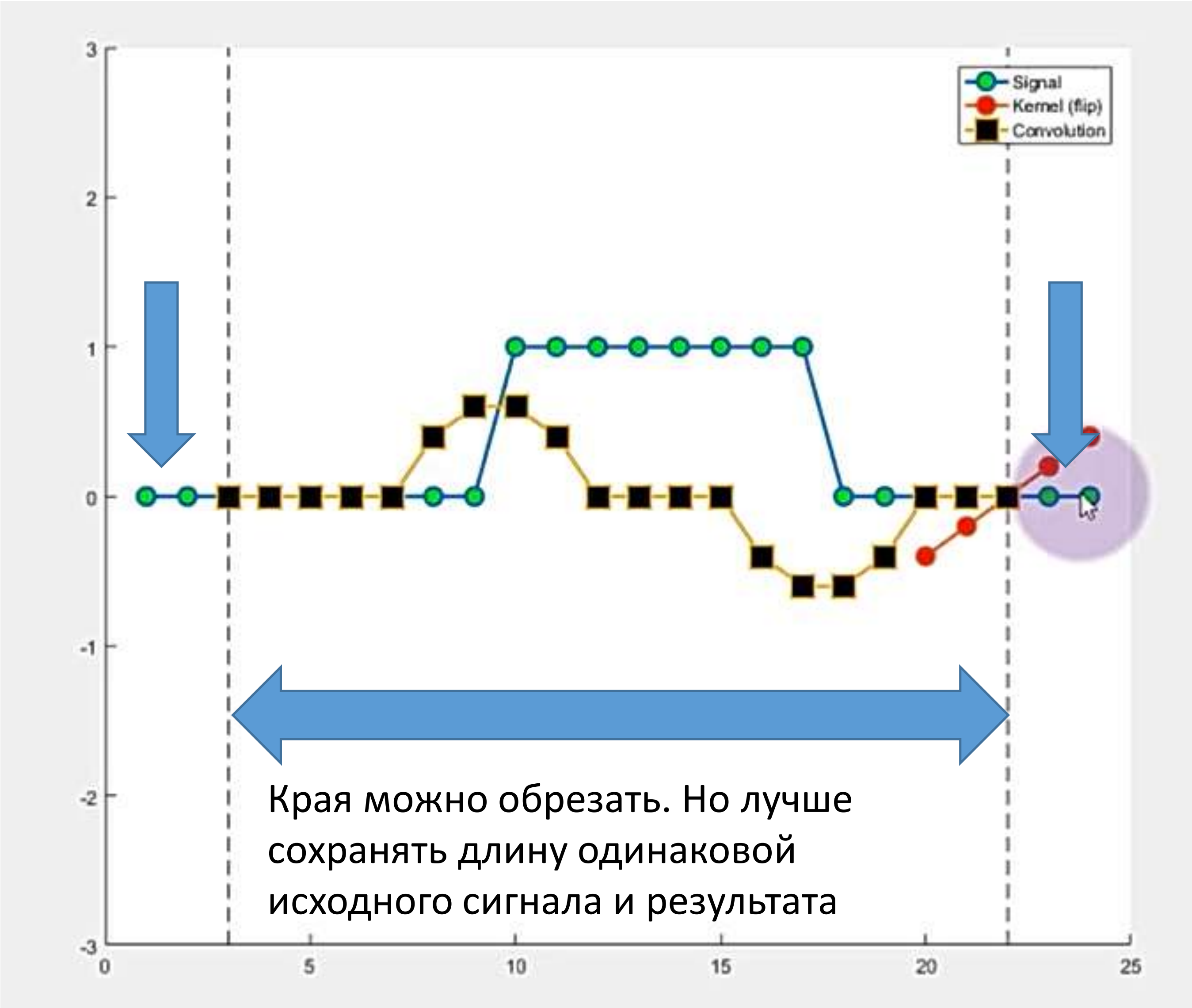


Получили фильтр edge detection

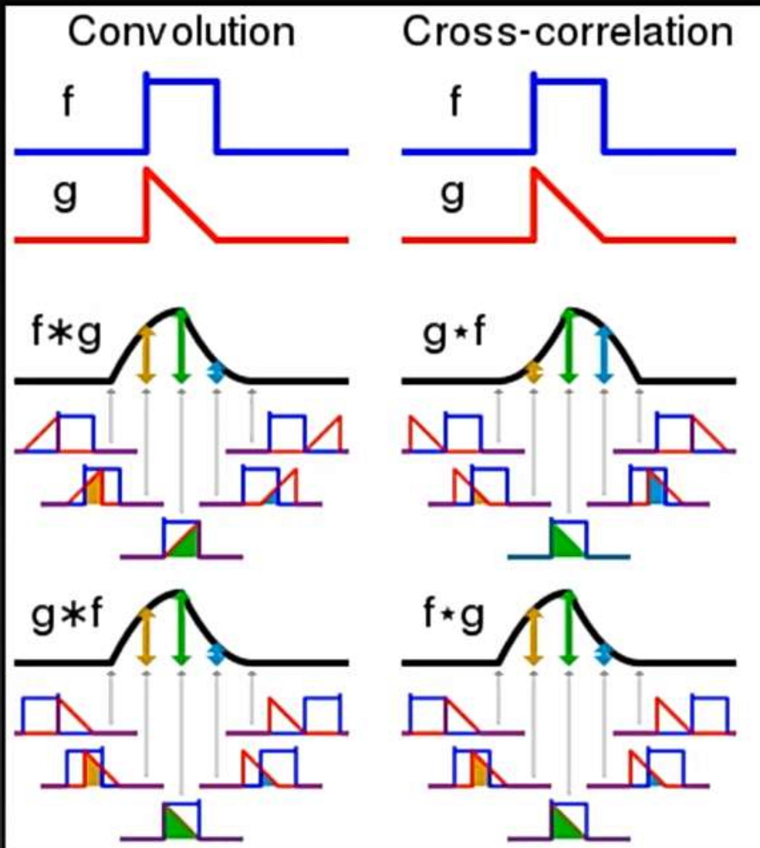
```
%% compare with MATLAB function

figure(4), clf

matlab_conv = conv(signal,kernel-mean(kernel),'same');
```



Сравнение со Встроенной функцией


Signal
Kernel (flip)
Convolution
Края можно обрезать. Но лучше
сохранять длину одинаковой
исходного сигнала и результата
3
2
1
0
-1
-2
-3
0
5
10
15
20
25

Why is the kernel flipped?



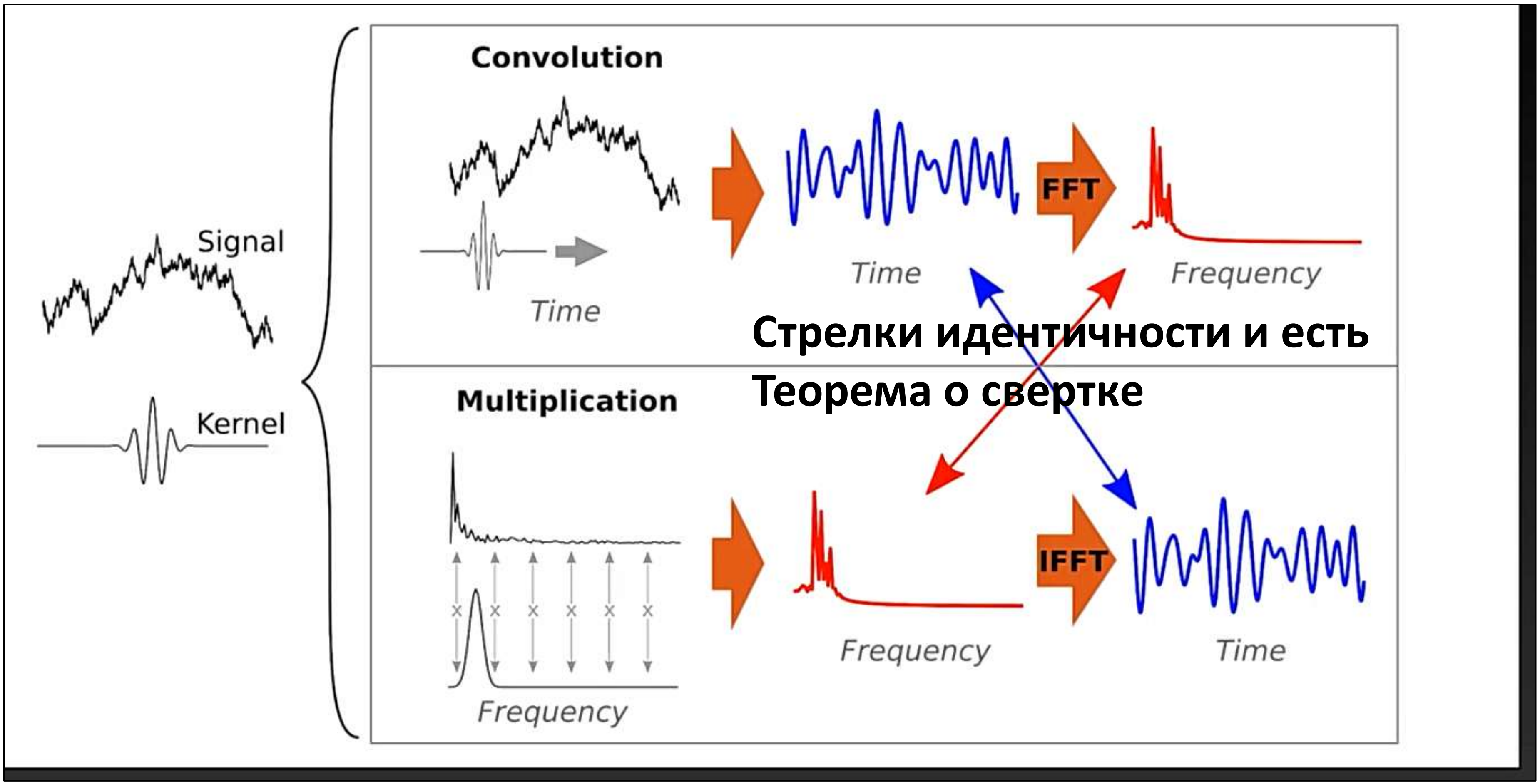
https://en.wikipedia.org/wiki/Convolution

Таким образом:
Для свертки мы ядро - переворачиваем, а
Для кросс корреляции - не переворачиваем!

Why is the kernel flipped?

Because the convolution theorem says so.

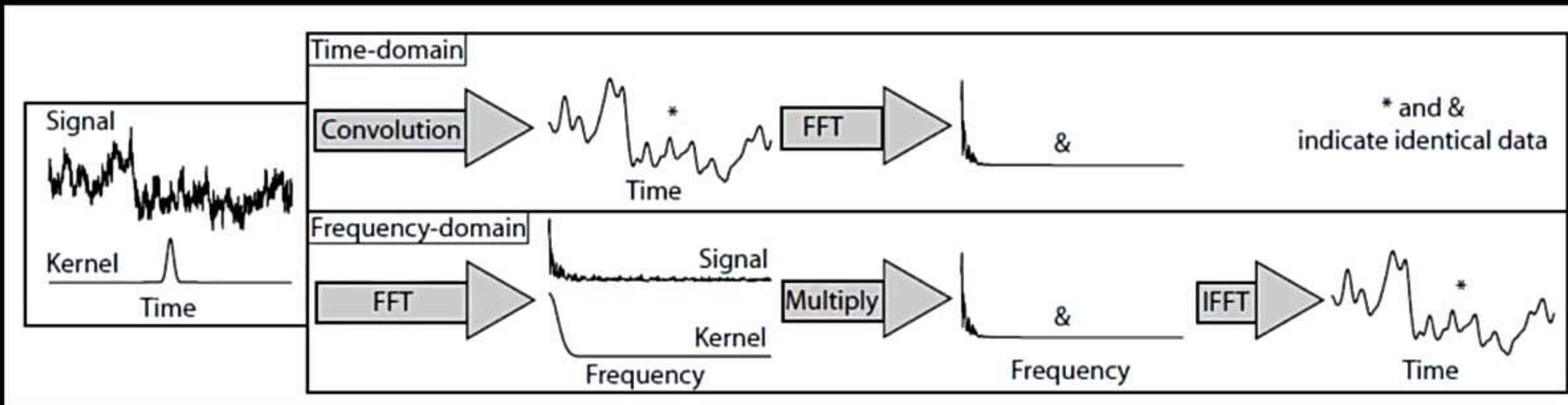
The convolution theorem



<u>Свертка полезна при обработке сигналов, в частности применяется для узкополосной фильтрации.</u>

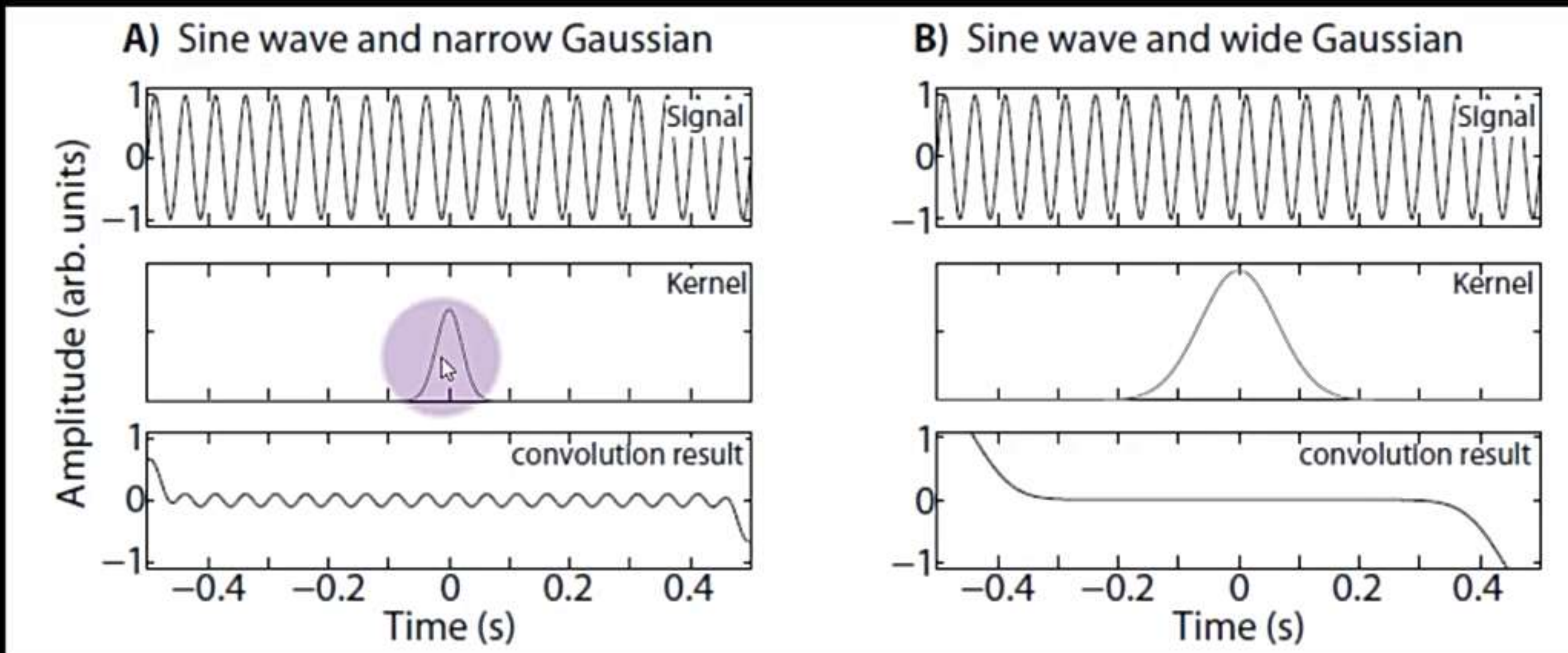
Почему теорема о свертке такая полезная?

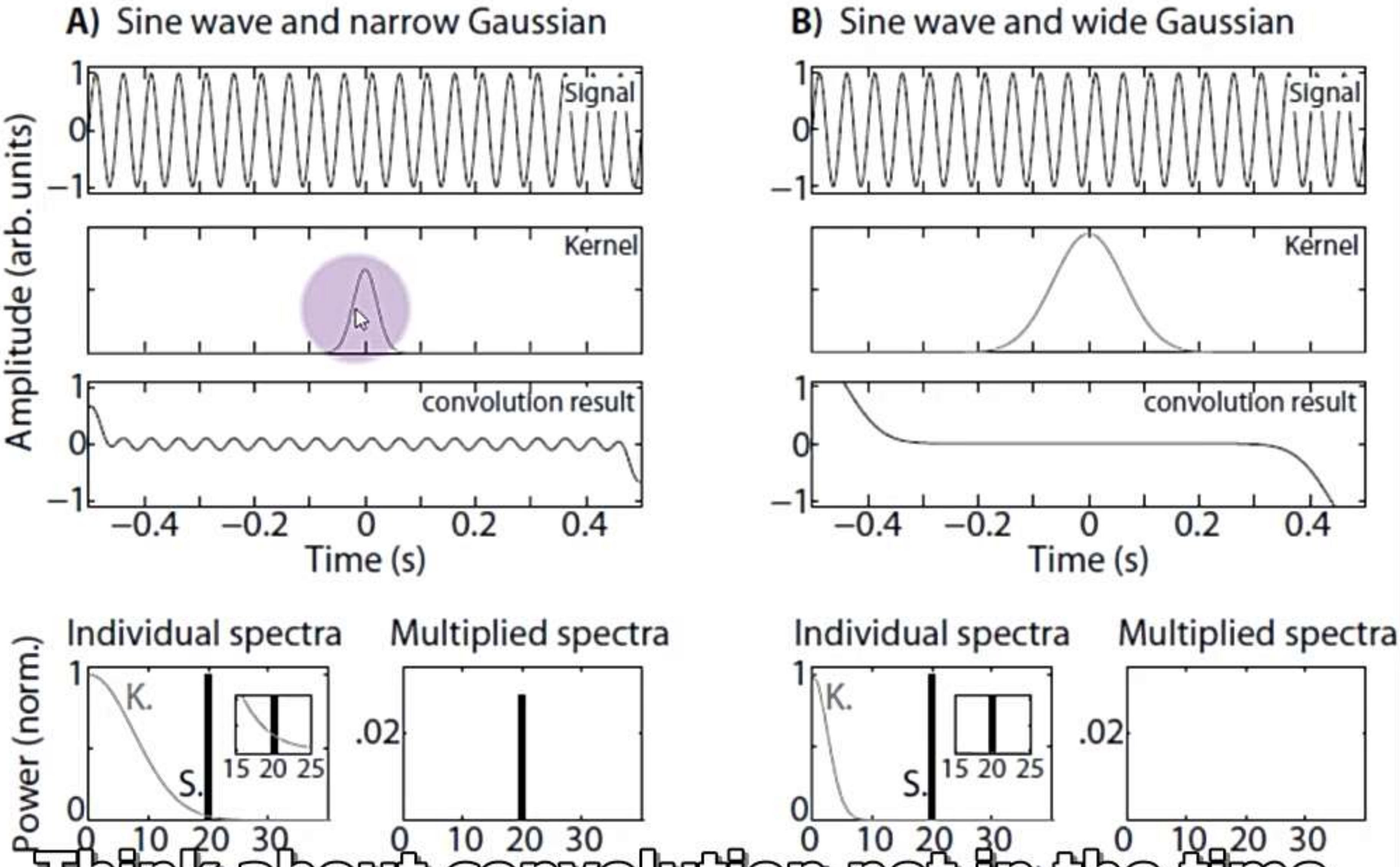
Lightning fast convolution in the frequency domain



Convolution in the time domain is equivalent to multiplication in the frequency domain.

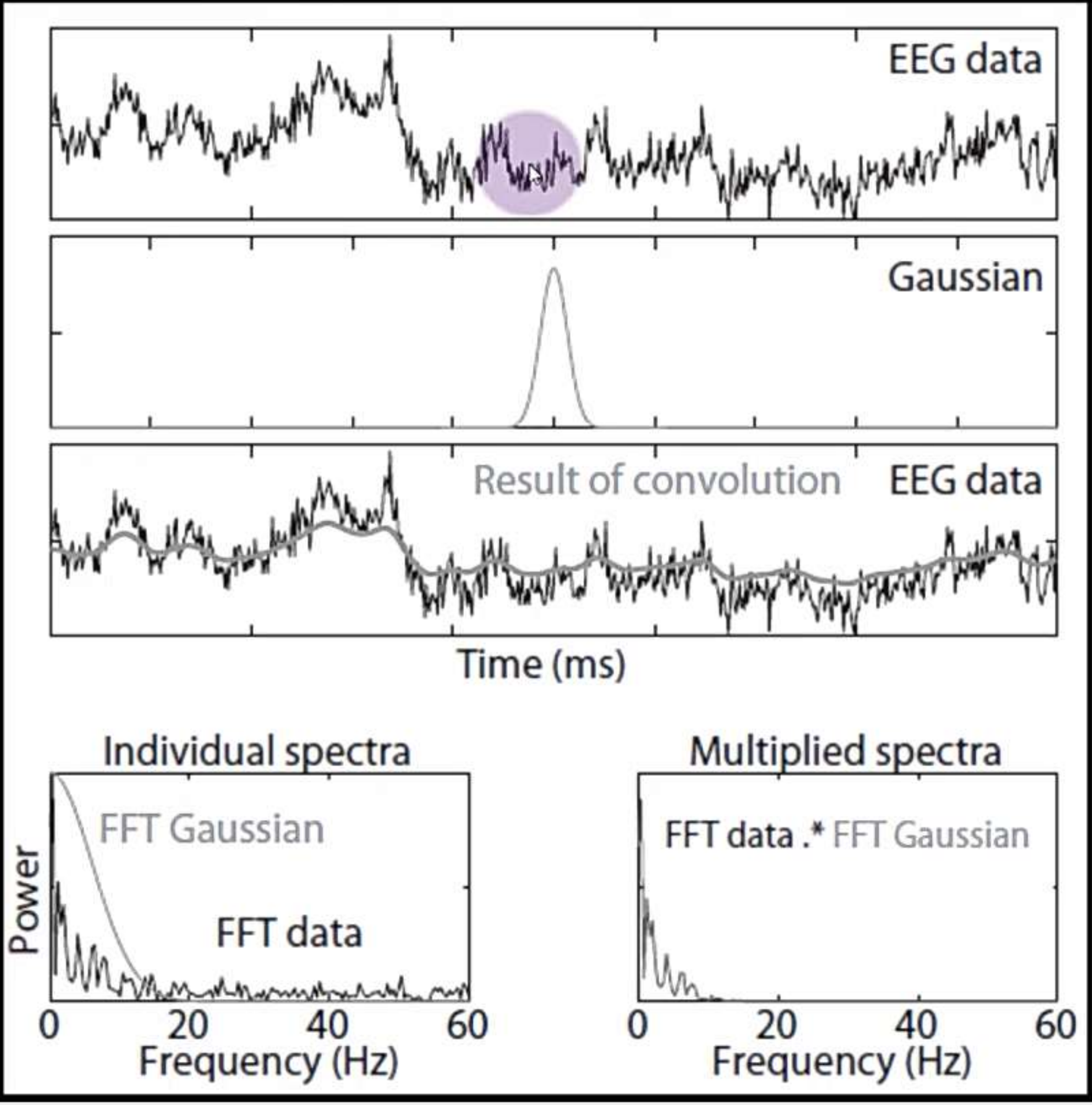
Convolution as a frequency filter





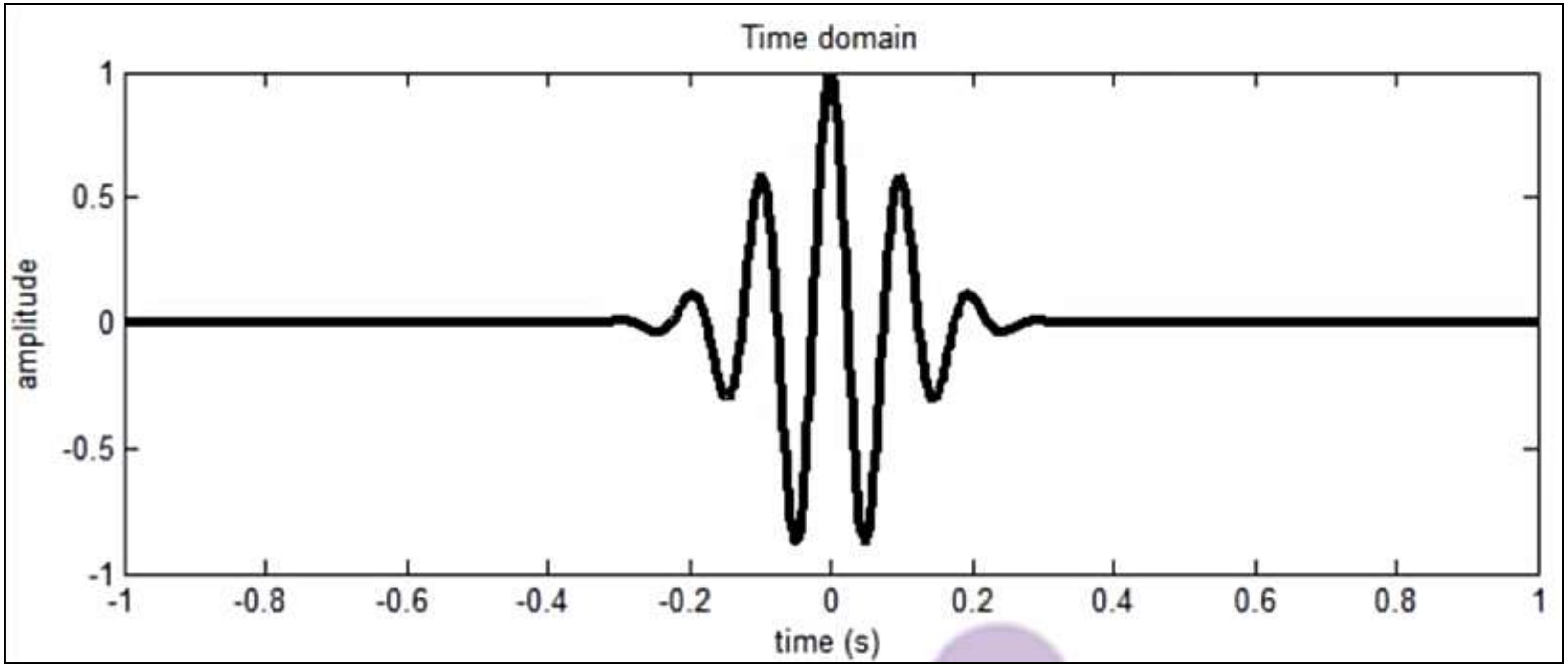
Think about convolution not in the time domain but in the frequency domain.

Convolution as a frequency filter



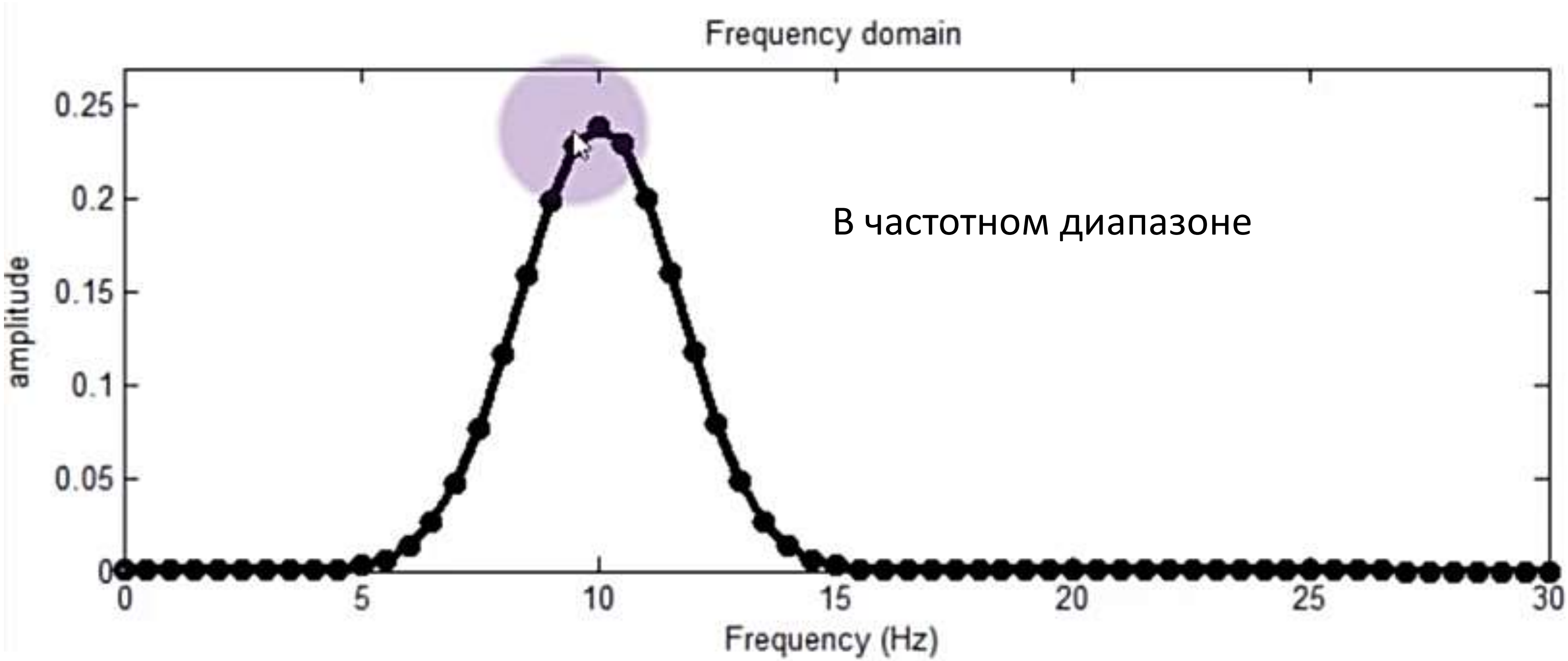
using a kernel that looks like this.

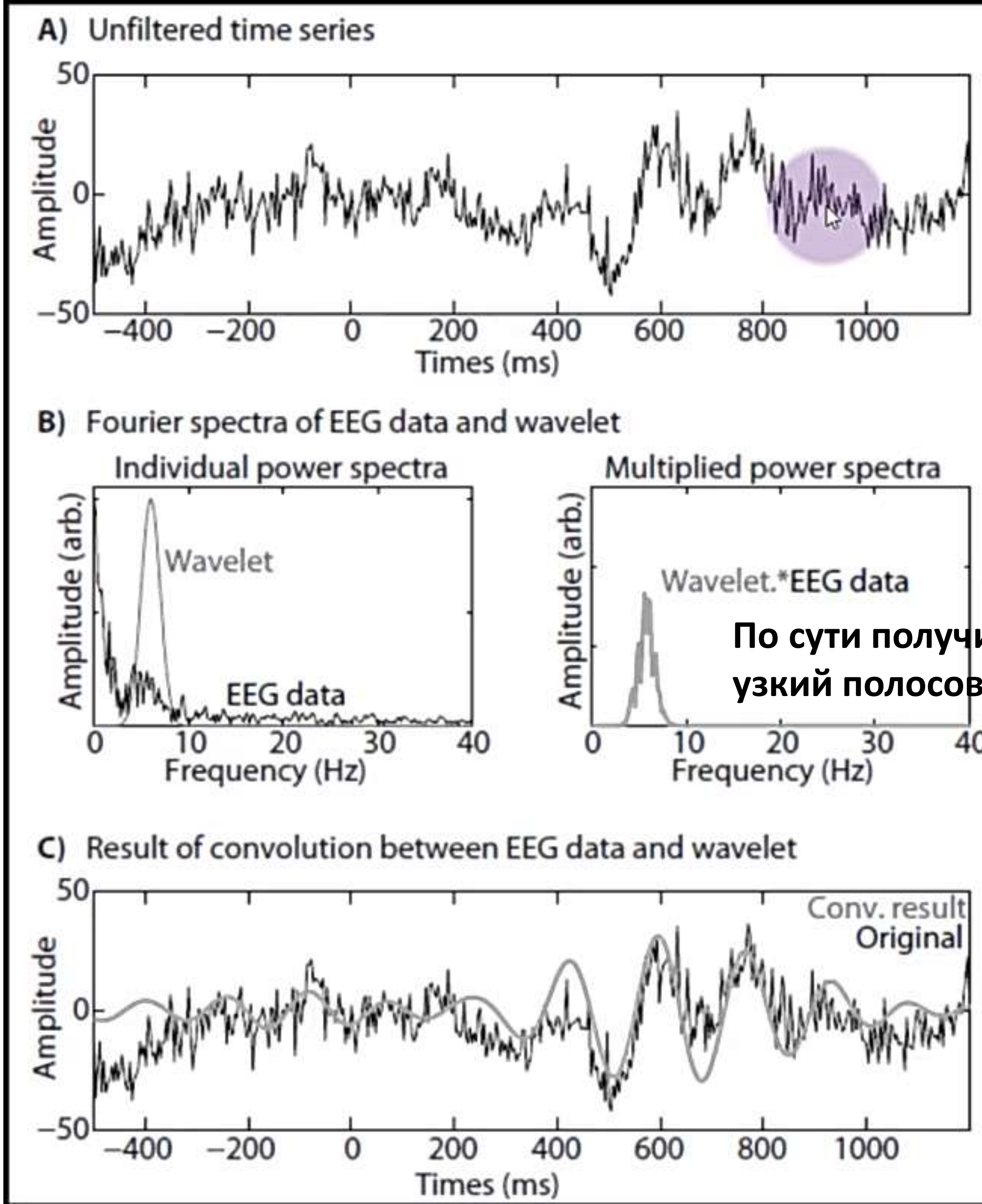
This is called a Morlet wavelet.



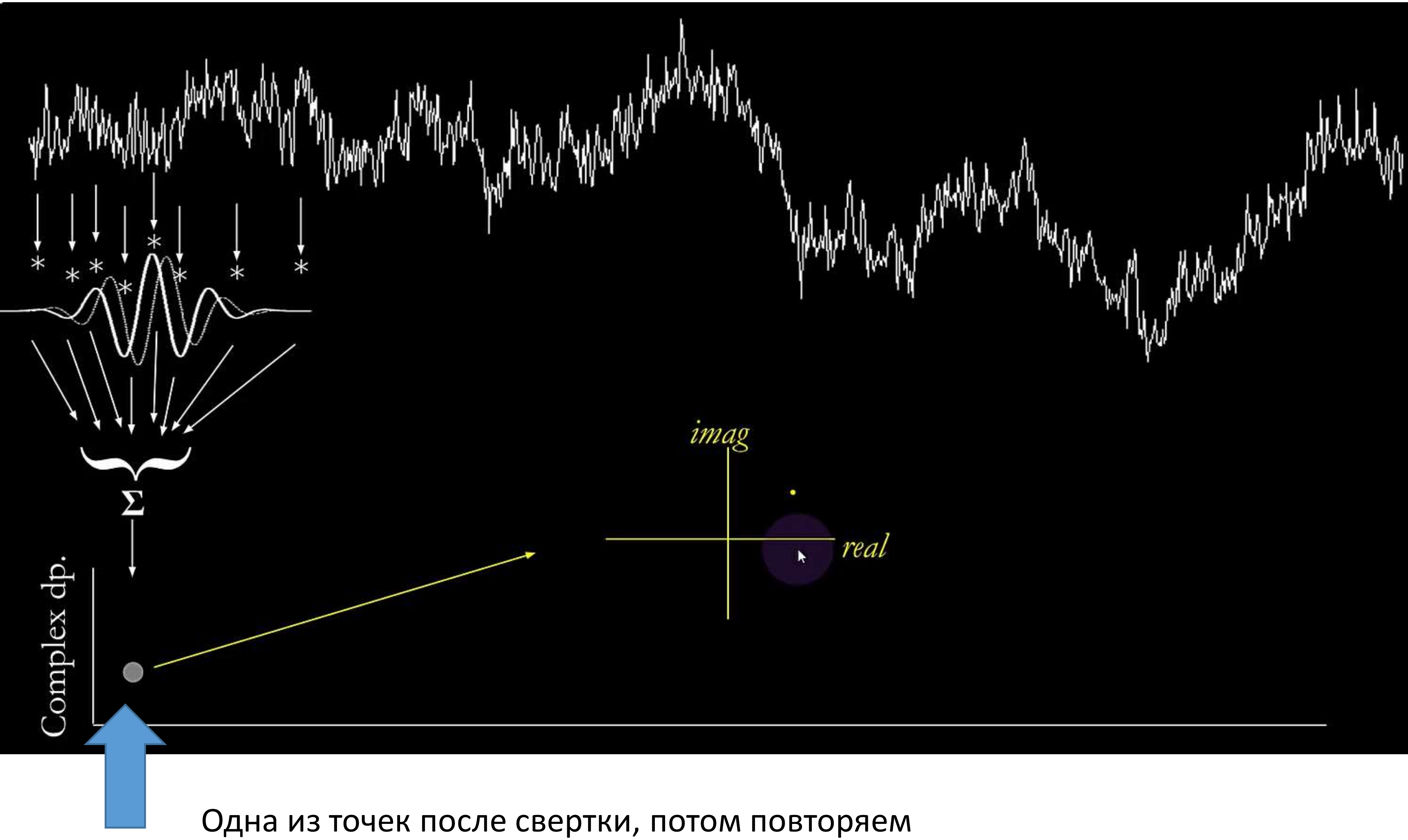
Morlet

wavelet is a sine wave that is multiplied by a Gaussian.

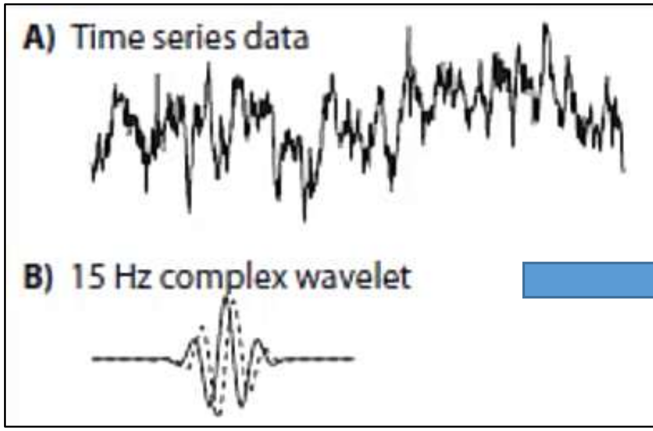


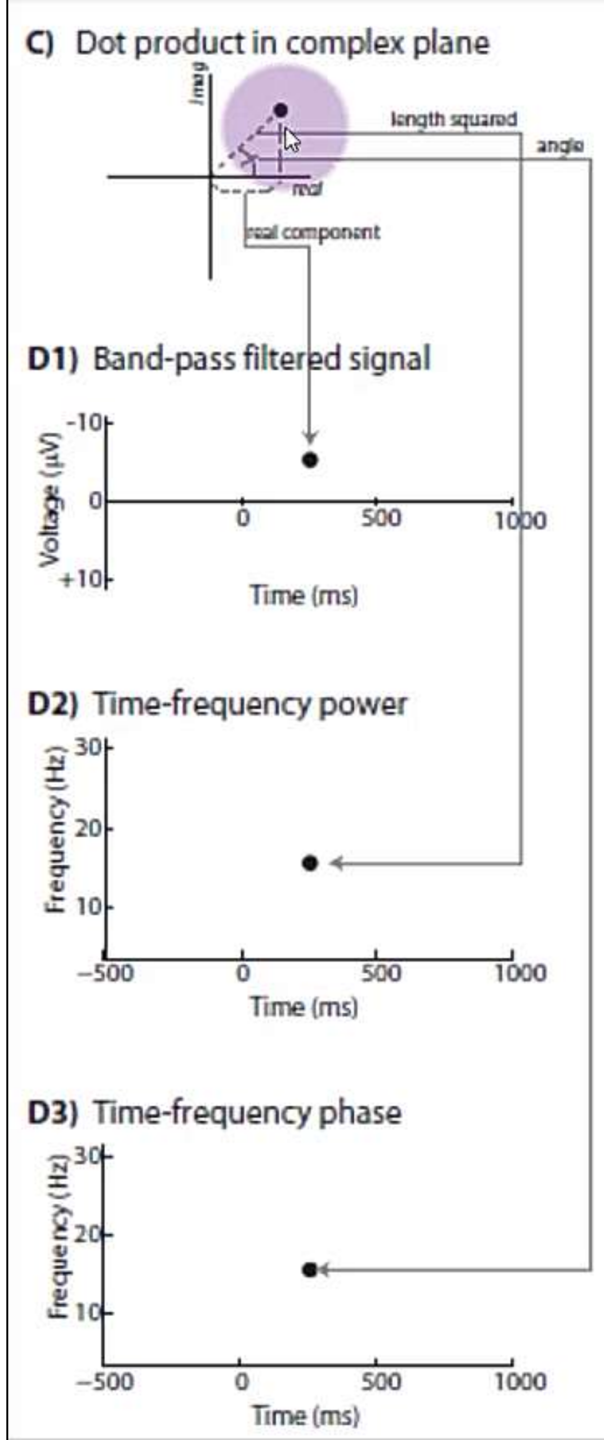


По сути получили узкий полосовой фильтр

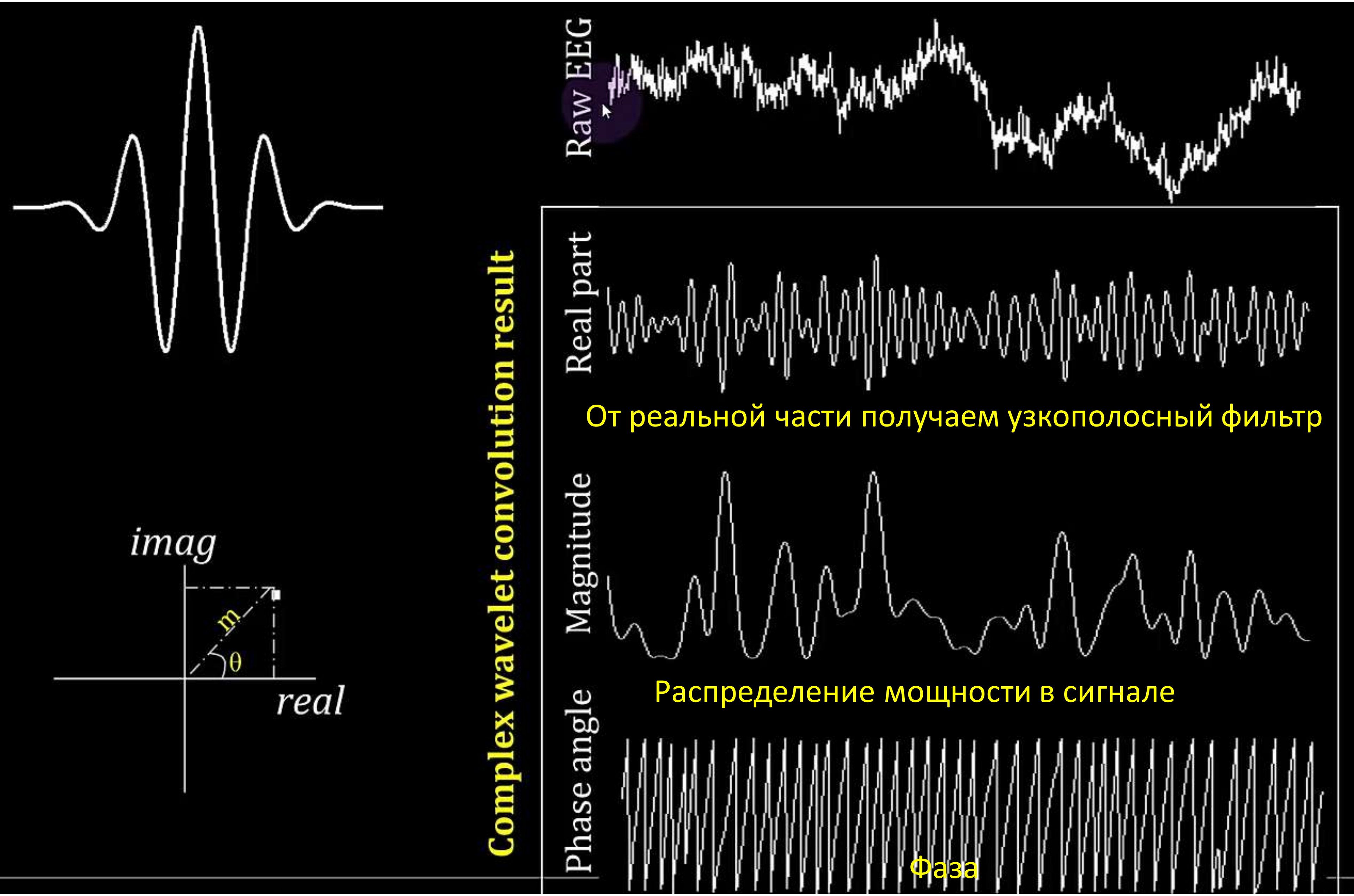


Одна из точек после свертки, потом повторяем



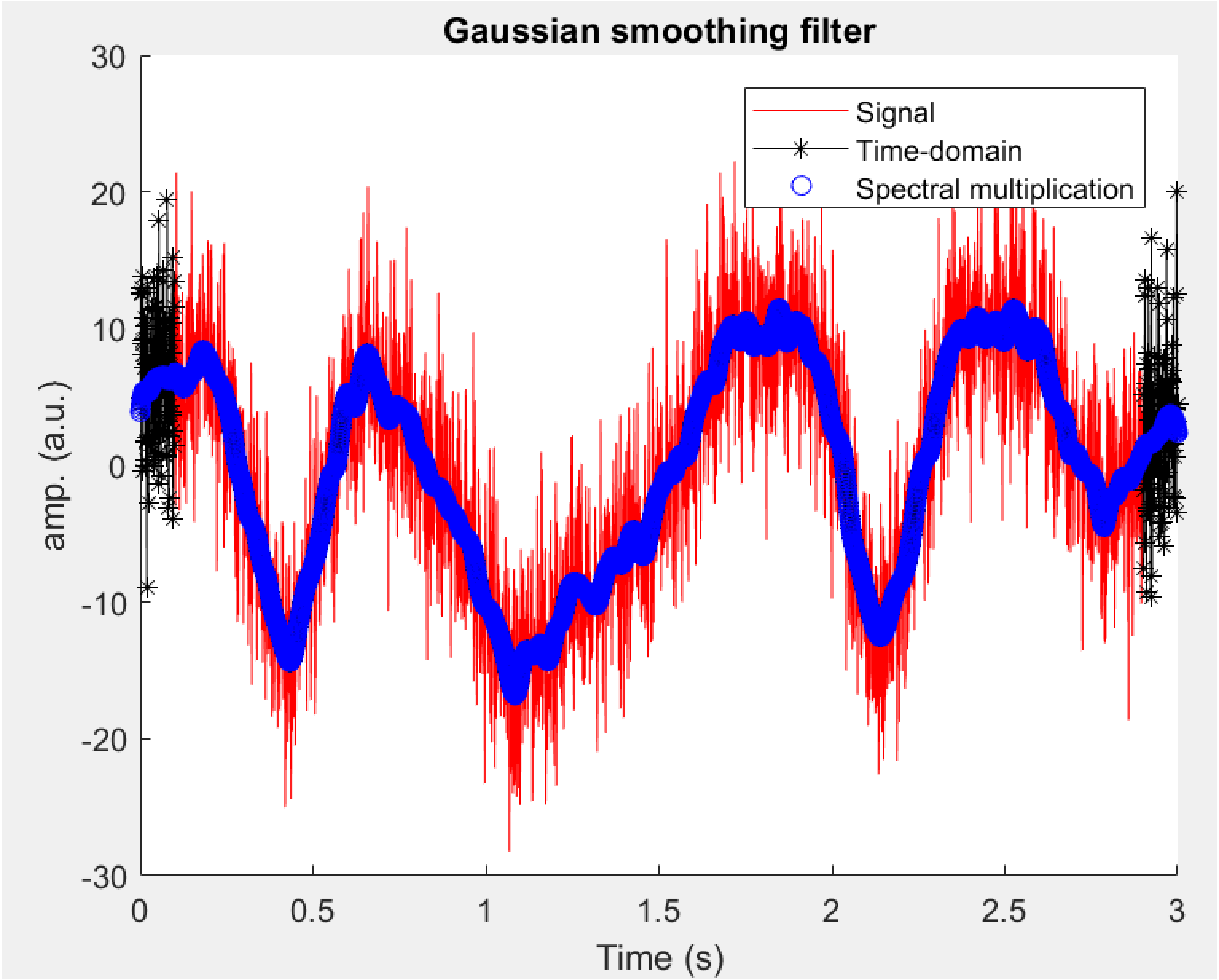


От комплексной плоскости
Получаем различные
Характеристики сигнала

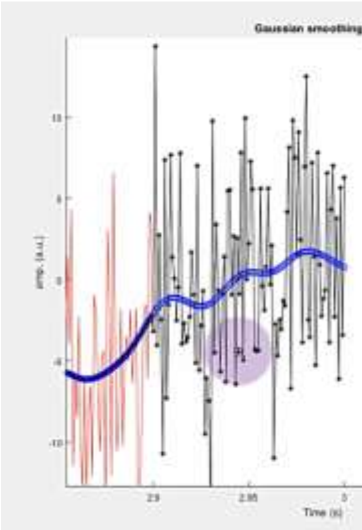


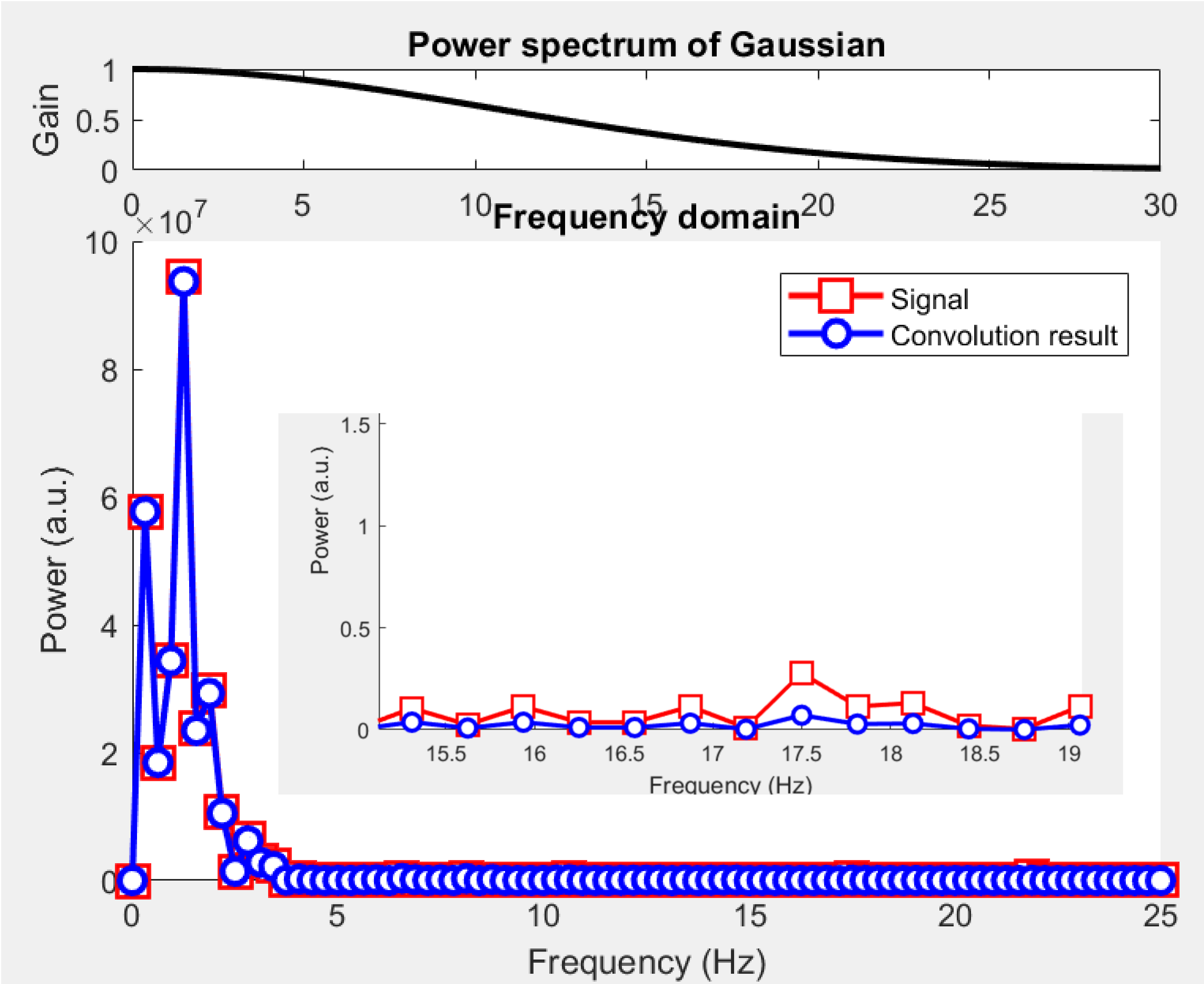
imag
m
θ
real
Complex wavelet convolution result
Raw EEG
Real part
От реальной части получаем узкополосный фильтр
Magnitude
Распределение мощности в сигнале
Phase angle
Фаза

Convolution with time-domain Gaussian (smoothing filter)



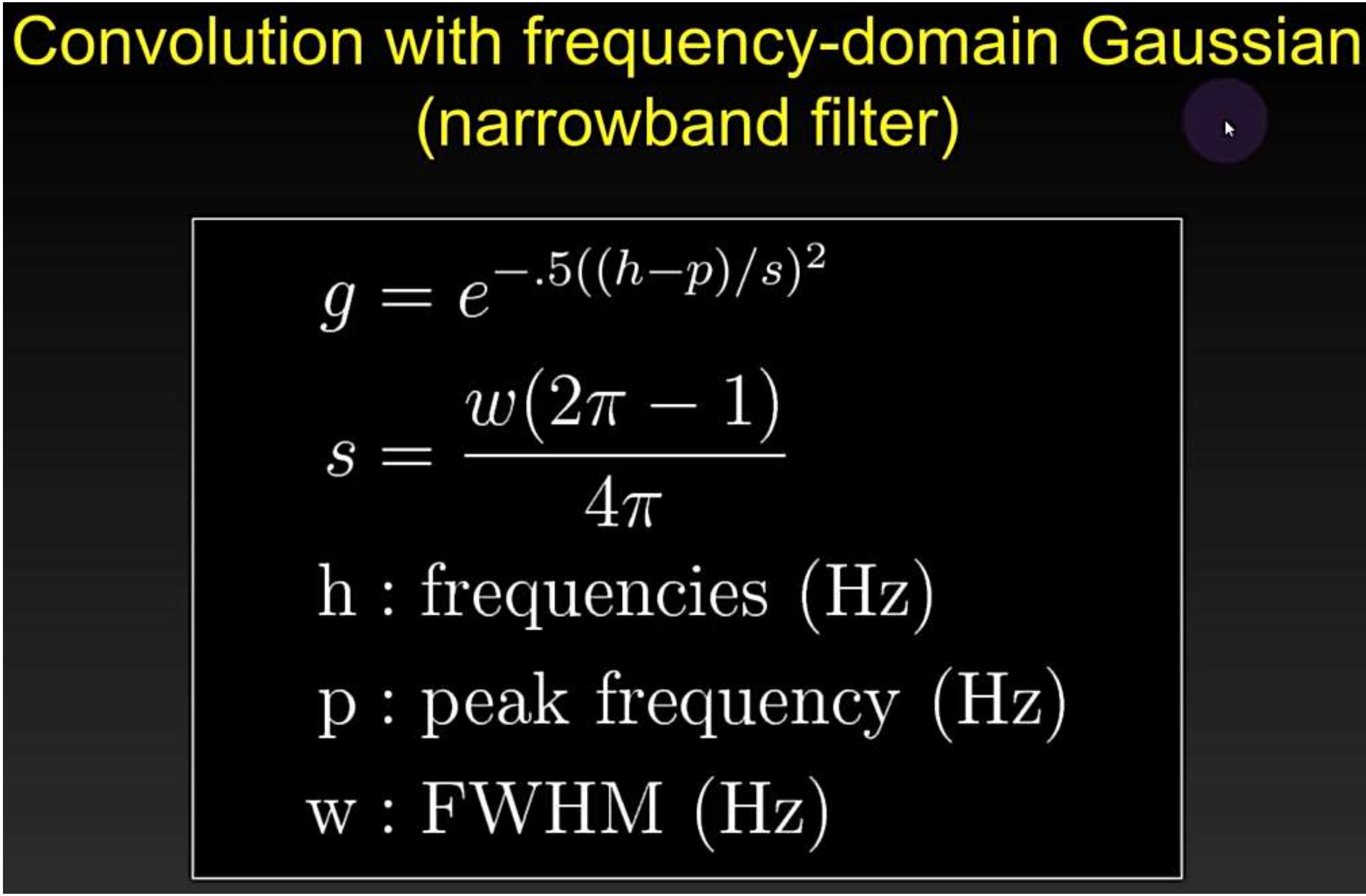
Краевой эффект



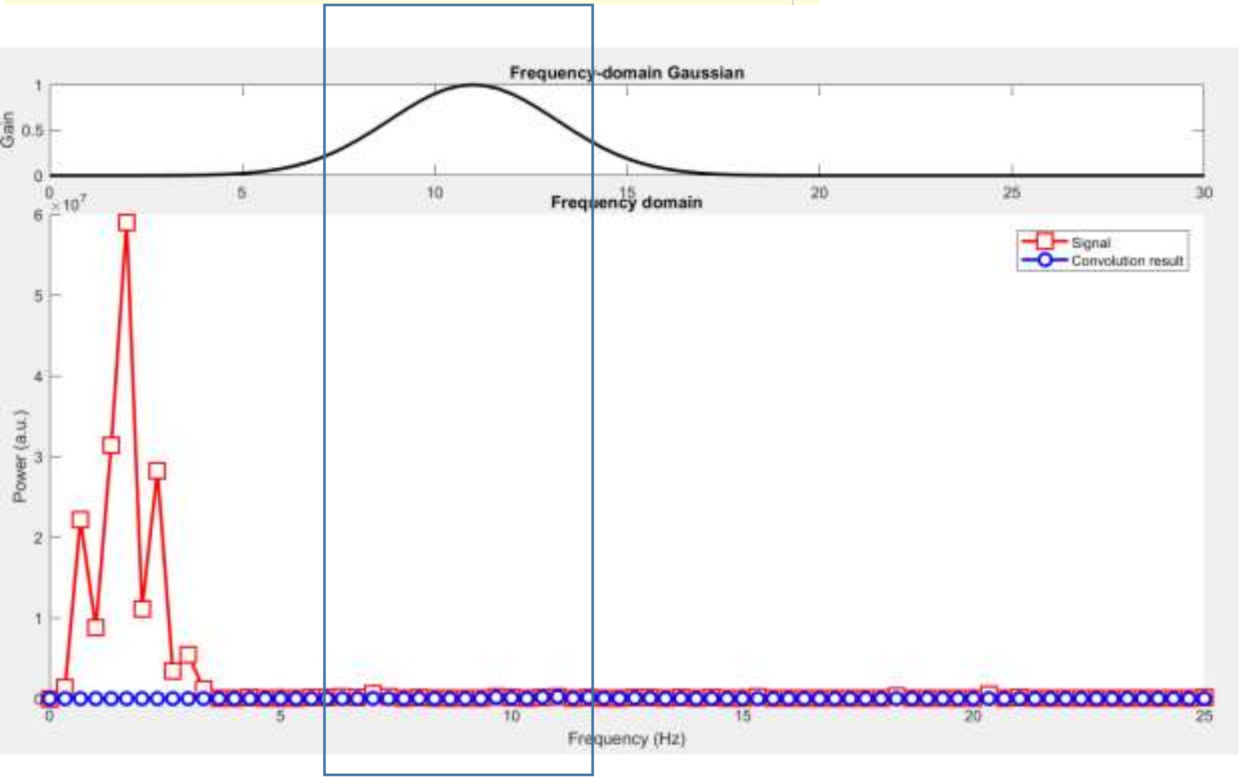


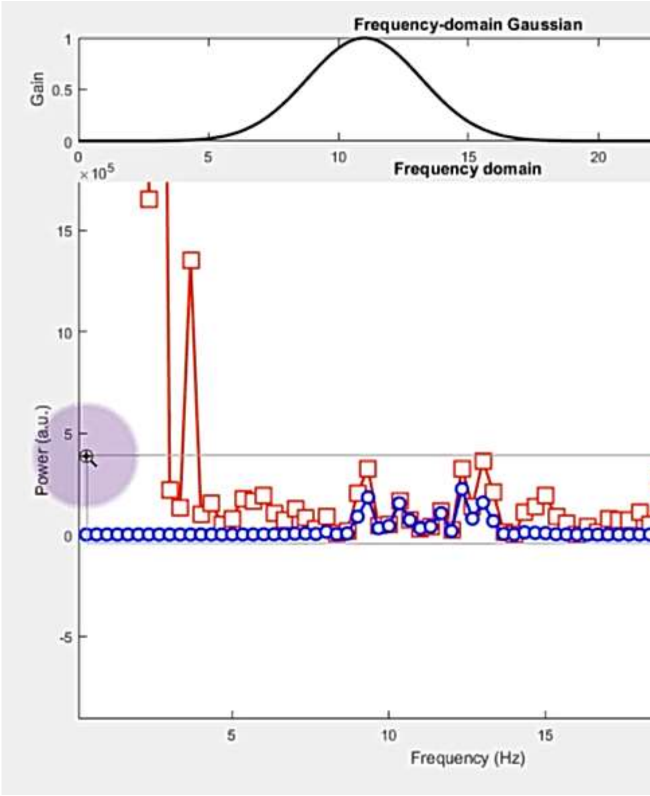
Power spectrum of Gaussian
Gain
1
0.5
0
0
5
10
15
20
25
30
Frequency domain
×10^7
10
8
6
4
2
0
Power (a.u.)
Signal
Convolution result
1.5
1
0.5
0
Power (a.u.)
15.5
16
16.5
17
17.5
18
18.5
19
Frequency (Hz)
0
5
10
15
20
25
Frequency (Hz)

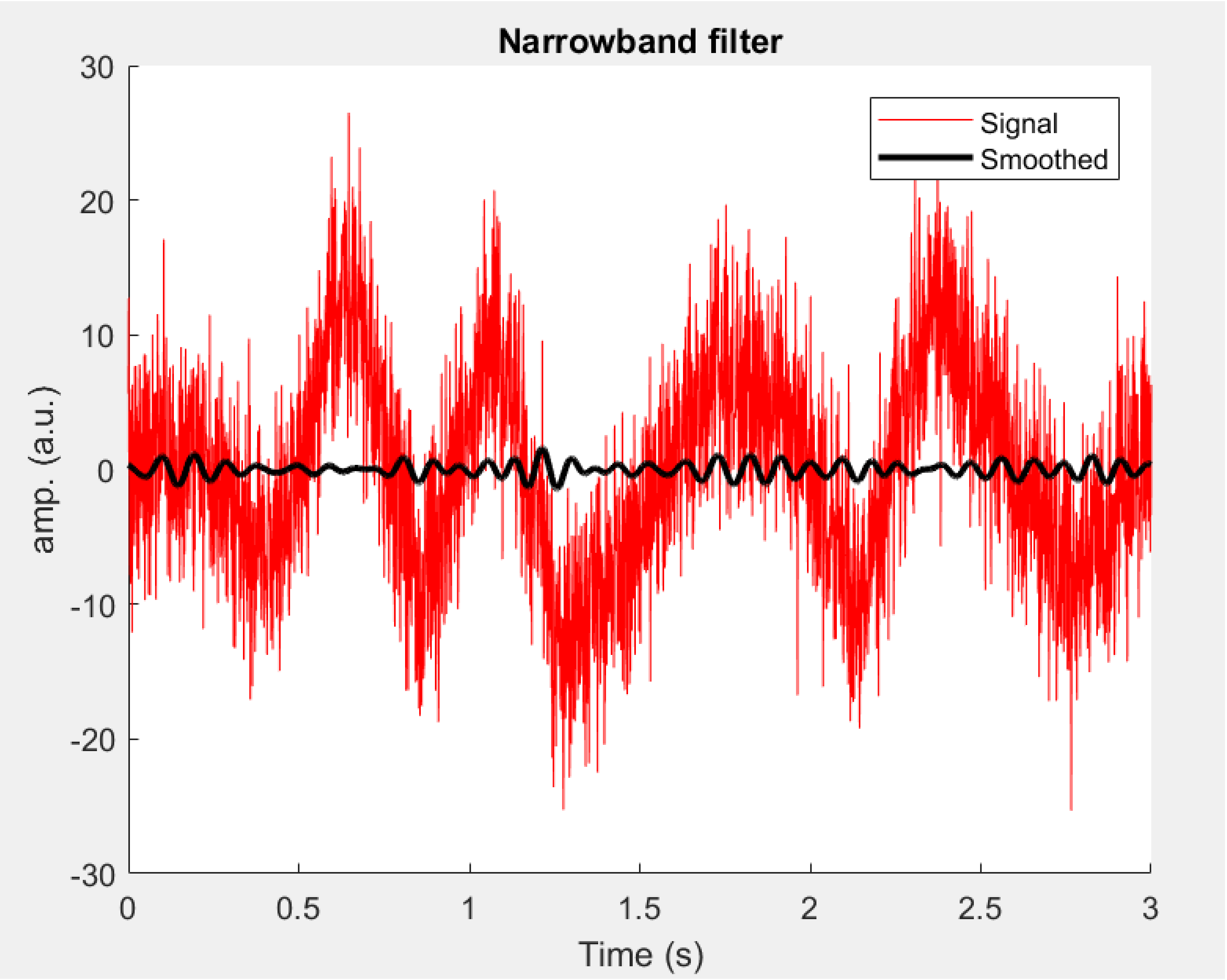
Convolution with frequency-domain Gaussian (narrowband filter)



```
%% create Gaussian spectral shape

% specify Gaussian parameters
peakf = 11;
fwhm  = 5.2;

% vector of frequencies
hz = linspace(0,srate,n);

% frequency-domain Gaussian
s  = fwhm*(2*pi-1)/(4*pi); % normalized width
x  = hz-peakf;             % shifted frequencies
fx = exp(-.5*(x/s).^2);    % gaussian
```

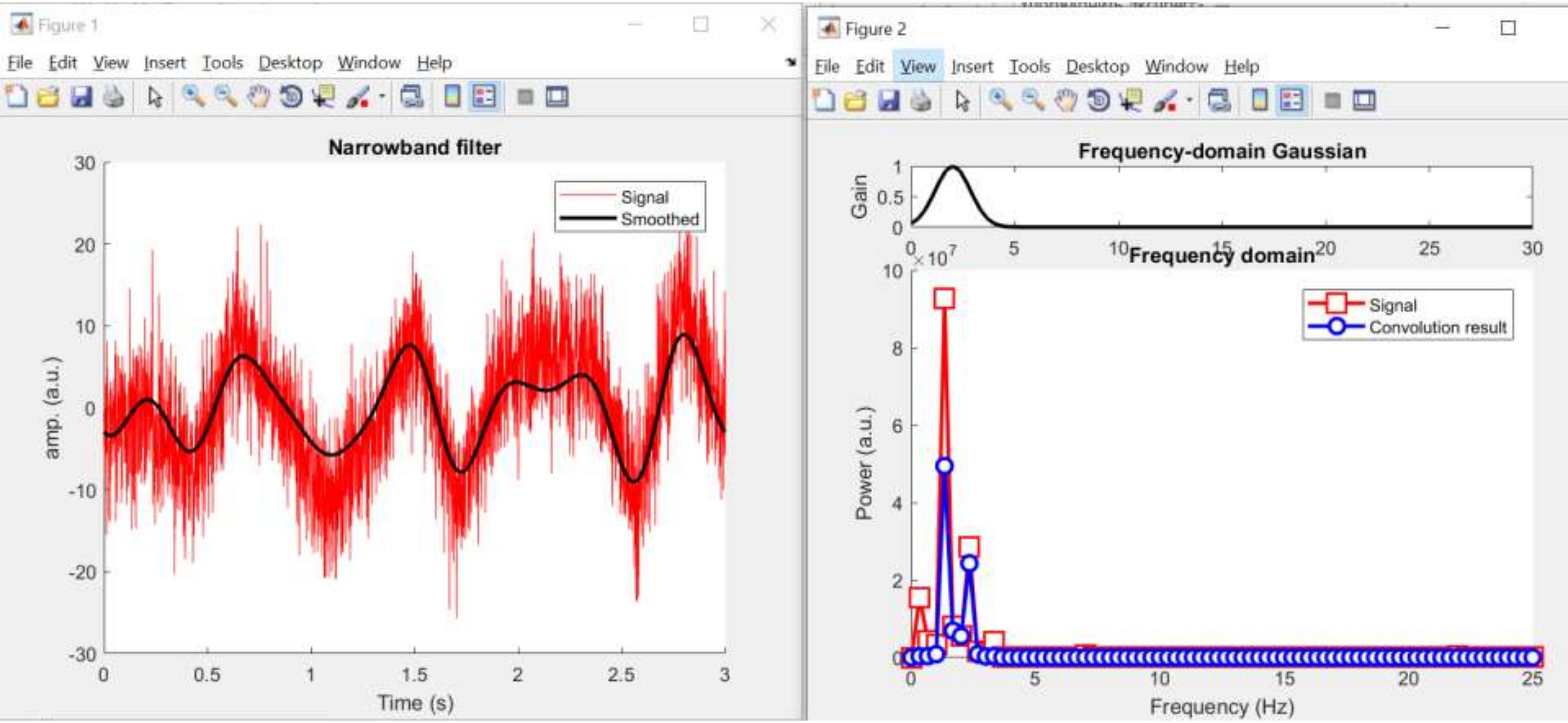






Narrowband filter
Signal
Smoothed
30
20
10
0
-10
-20
-30
amp. (a.u.)
0
0.5
1
1.5
2
2.5
3
Time (s)

```
% specify Gaussian parameters
% peakf = 11;
peakf = 2;
% fwhm  = 5.2;
fwhm  = 2.0;
```

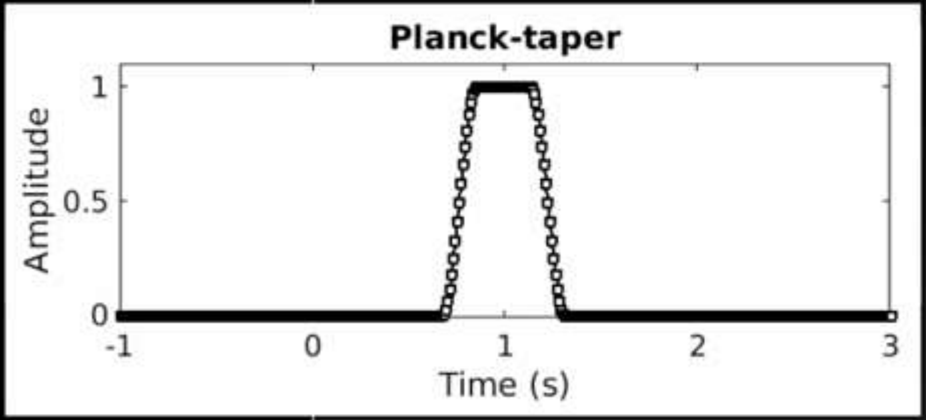


Convolution with Planck taper (bandpass filter)

Гладкие боковые поверхности и вершина плоская

Convolution with frequency-domain Planck taper (bandpass filter)

$$
\begin{cases}
\dfrac{1}{e^{z_l}+1}, & 0 < t < N-1 \\
1, & \epsilon(N-1) \le t \le (1-\epsilon)(N-1) \\
\dfrac{1}{e^{z_r}+1}, & (1-\epsilon)(N-1) < t < N-1
\end{cases}
$$



$$z_l = \epsilon(N-1)\left(\frac{1}{t} + \frac{1}{t-\epsilon(N-1)}\right)$$

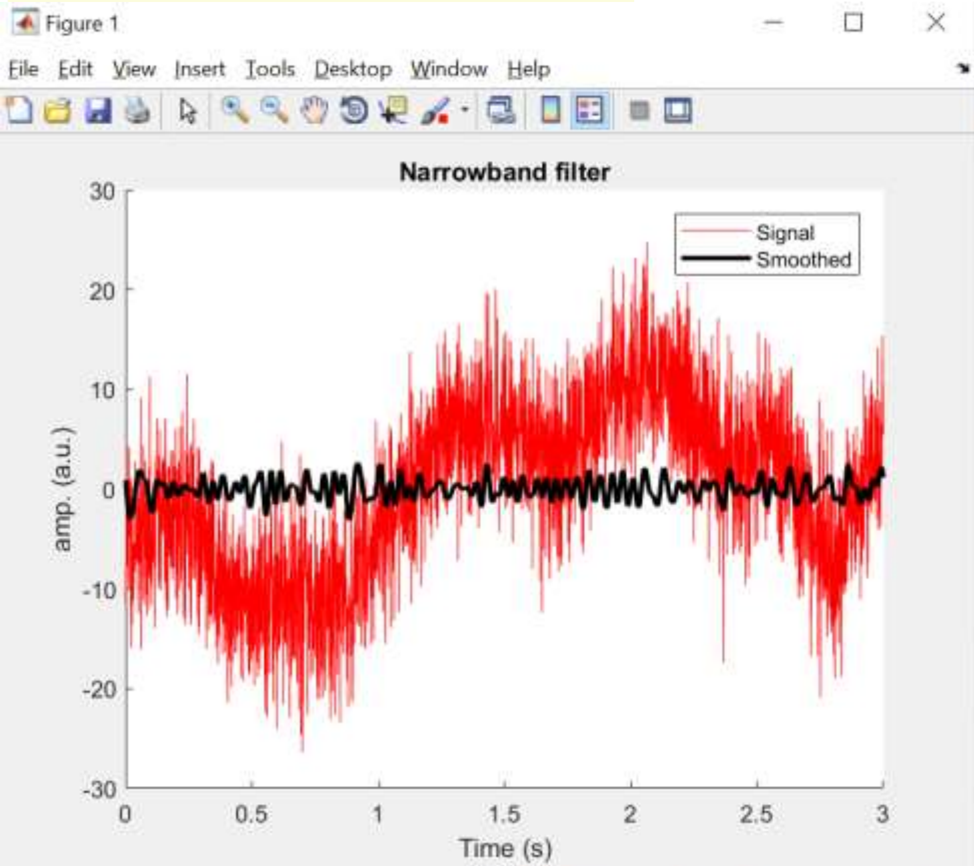
$$z_r = \epsilon(N-1)\left(\frac{1}{N-1-t} + \frac{1}{(1-\epsilon)(N-1)-t}\right)$$

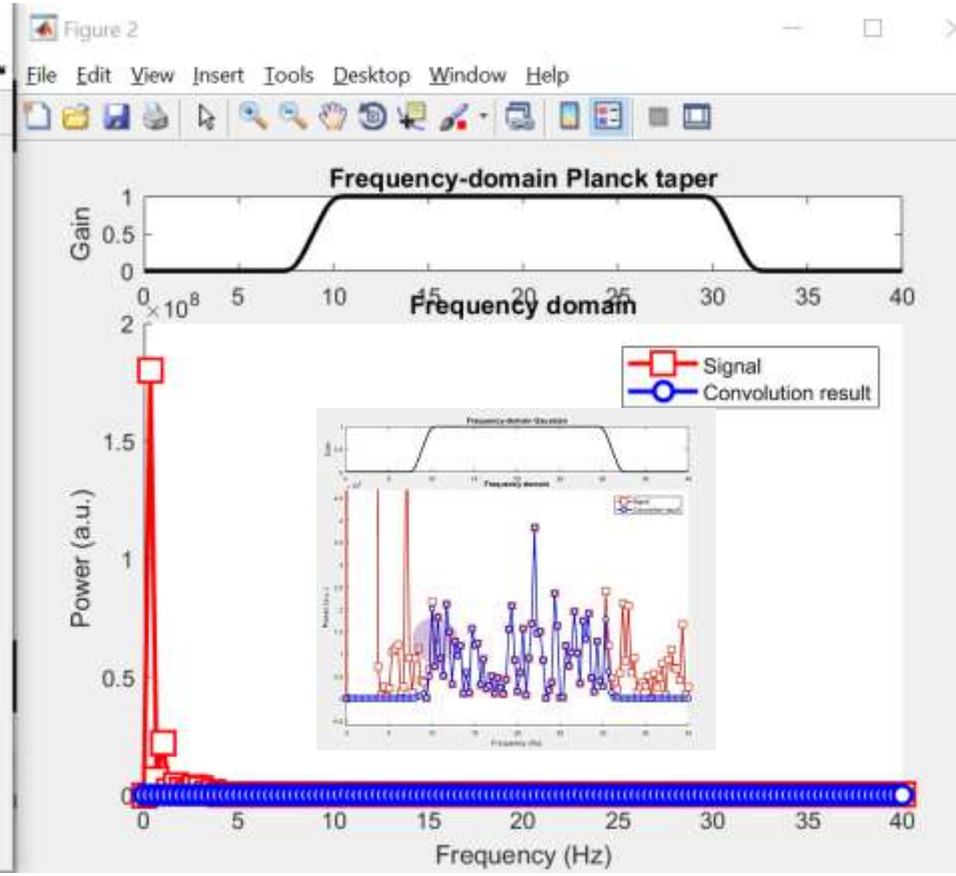
$0 < \epsilon \le .5$ Характеризует степень уменьшения и увеличения границ фильтра

Три части нашего фильтра

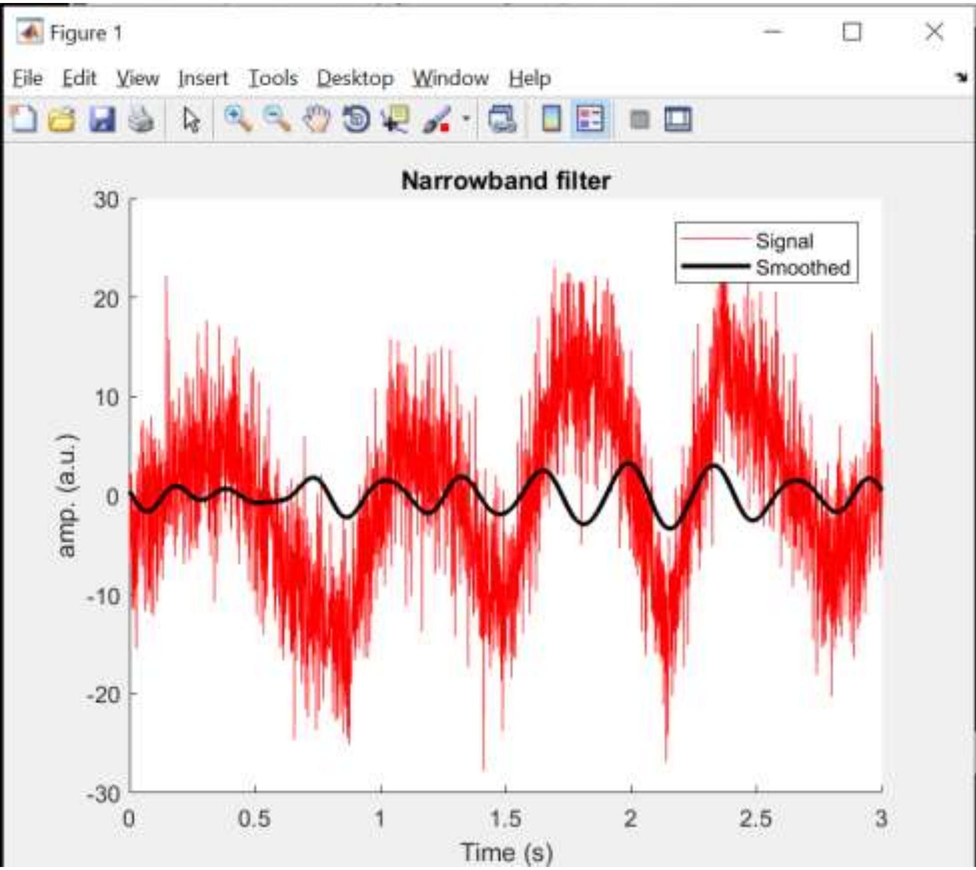
1) Левая функция – сигмоид
2) Центр
3) Правая функция сигмоид

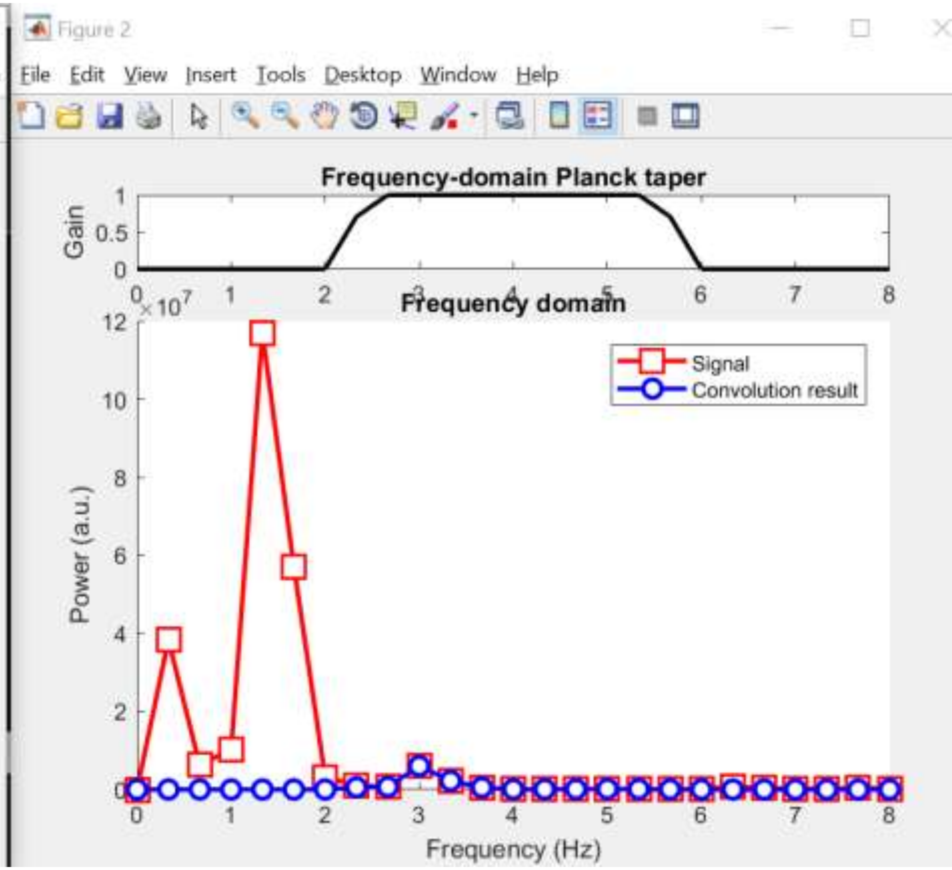
```
% frequencies
hz = linspace(0,srate,n);
% edge decay, must be between 0 and .5
eta = .15;
% spectral parameters
fwhm  = 13;
peakf = 20;
% convert fwhm to indices
np = round( 2*fwhm*n/srate );
pt = 1:np;
% find center point index
fidx = dsearchn(hz',peakf);
```

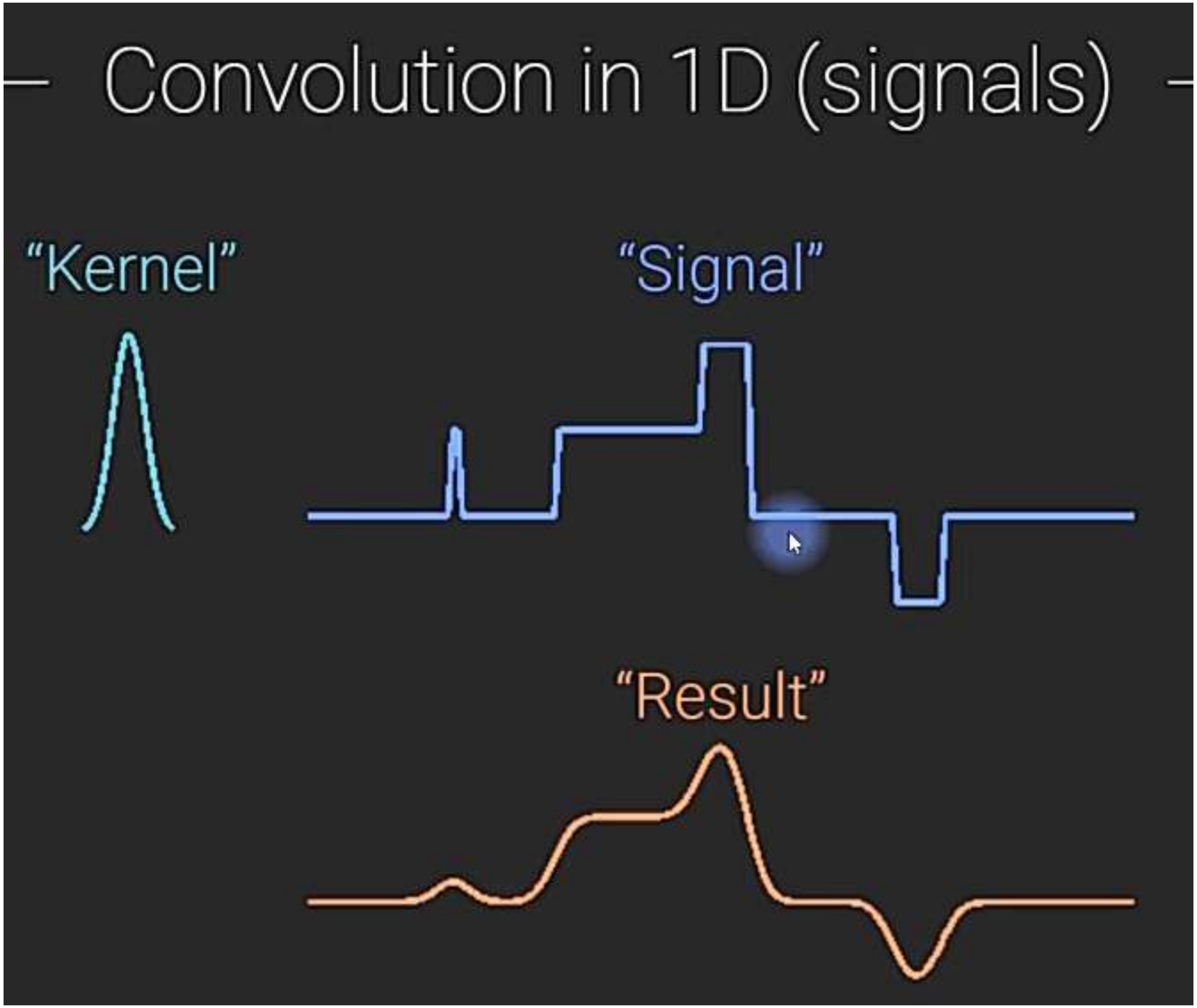


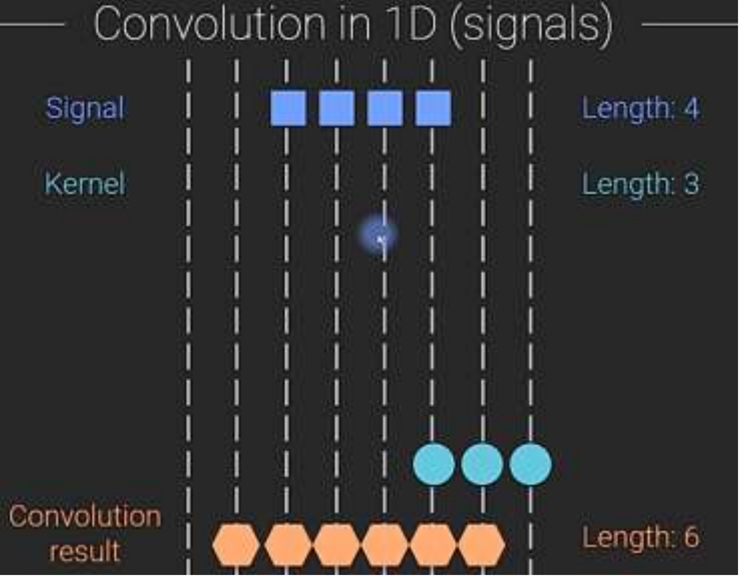


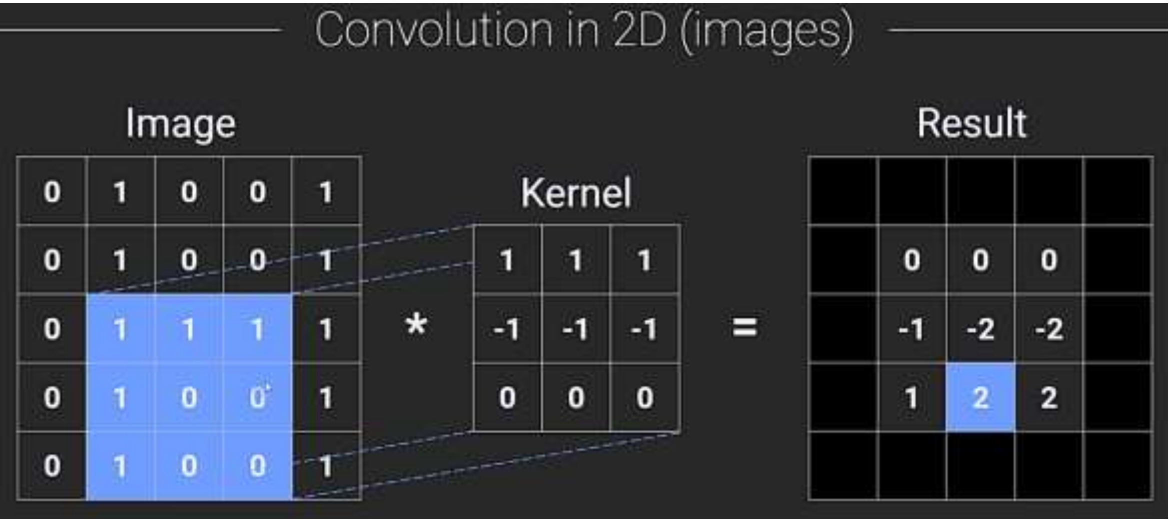
```
hz = linspace(0,srate,n);
% edge decay, must be between 0 and .5
eta = .15;
% spectral parameters
fwhm  = 2;
peakf = 4;
```





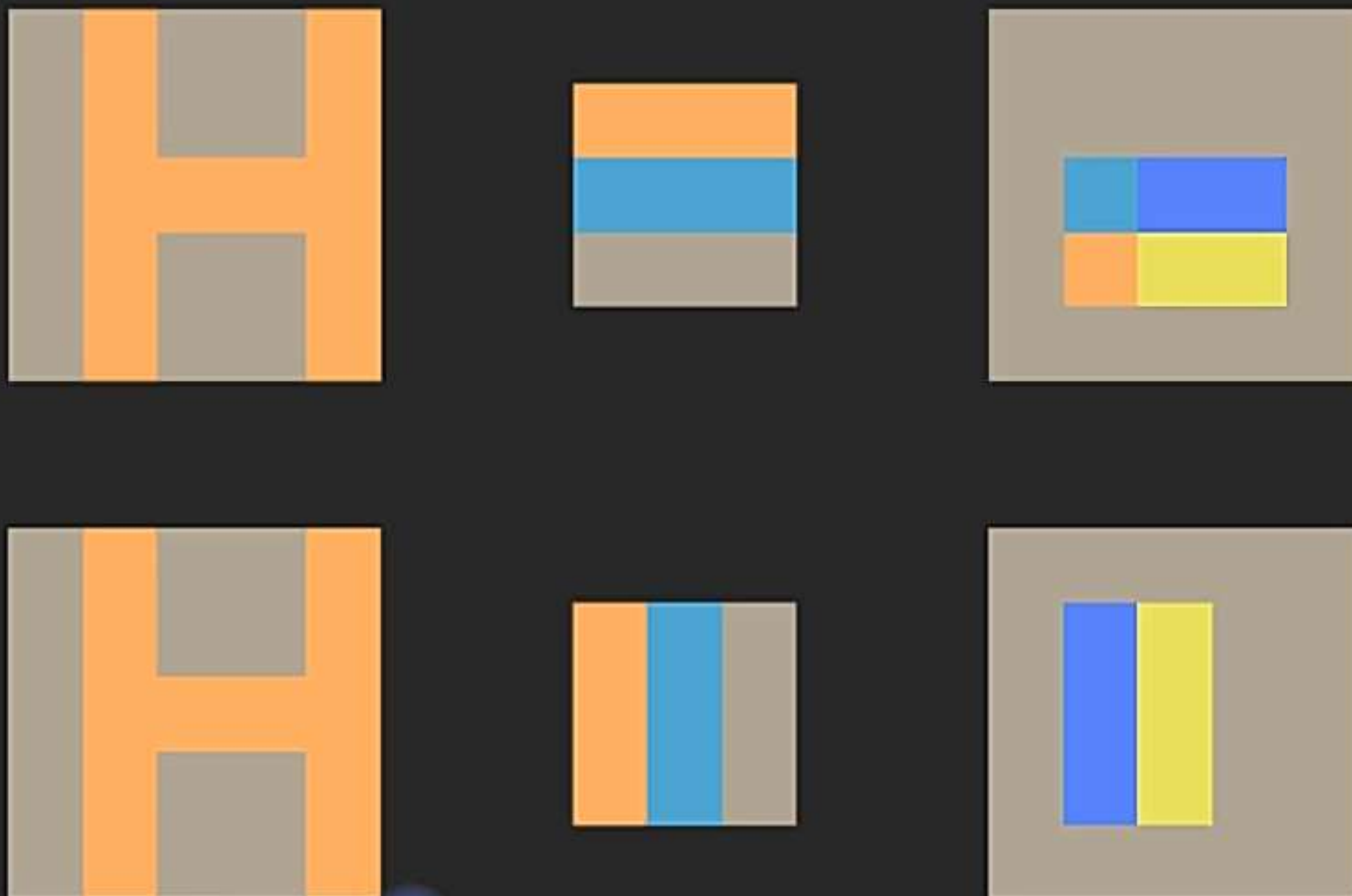

Convolution in 1D (signals)
"Kernel"
"Signal"
"Result"

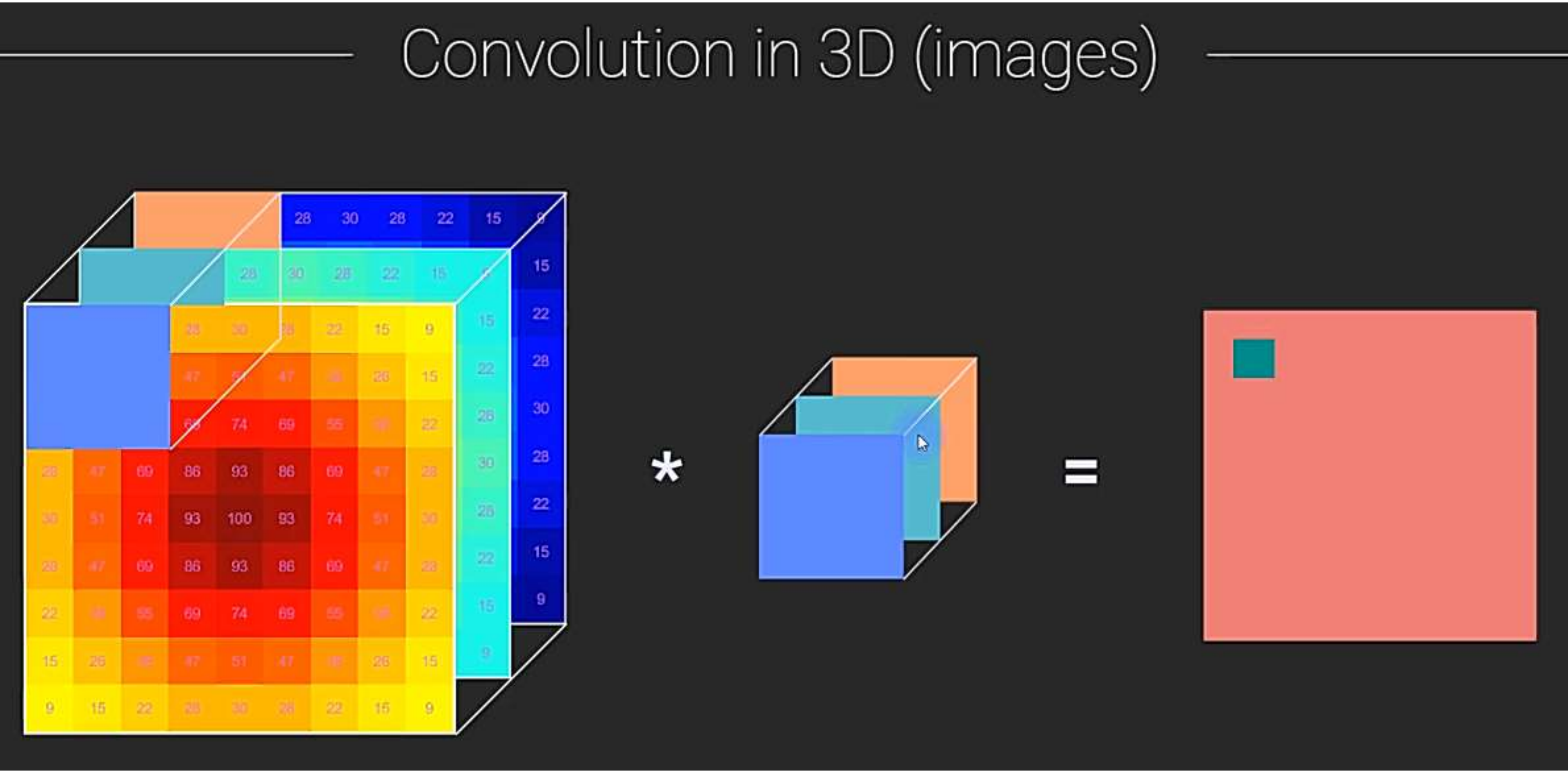




So we got the result that was actually longer than the original signal here.

Convolution in 2D (images)




Convolution in 3D (images)

Лабораторная работа