Convolution Analysis of Signals

Indian Institute of Technology
Hyderabad

Group Quiz : 02

EE1060: Differential Equations and Transform Techniques

Ankit Jainar	**EE24BTECH11004**
Arnav Yadnopavit	**EE24BTECH11007**
Dulla Karthik	**EE24BTECH11017**
Janjanam Kedarananda	**EE24BTECH11030**
Harsha Vardhan Reddy	**EE24BTECH11063**
Harshil Rathan Y	**EE24BTECH11064**

Contents

1 Introduction : Problem Statement

Compute the convolution of a given continuous-time signal $f(t)$ with a rectangular kernel $h(t)$, defined by:

$$h(t) = \begin{cases} 1, & -T \le t \le T, \\ 0, & \text{otherwise.} \end{cases} \tag{0.1}$$

Derive the analytical expression for

$$y(t) = (f * h)(t) = \int_{-\infty}^{\infty} f(\tau)\, h(t-\tau)\, d\tau, \tag{0.2}$$

in terms of known elementary functions. Analyze the resulting output for different kernel durations T and for a chosen input signal $f(t)$ (e.g., a step or sinusoid).

a) Extended Scenarios

1) Modify the kernel to only include its causal part:

$$h_{\text{causal}}(t) = \begin{cases} 1, & 0 \le t \le T, \\ 0, & \text{otherwise,} \end{cases}$$

and determine how this change affects $y(t)$.

2) Introduce a time shift τ_0 in the kernel,

$$h_{\tau_0}(t) = h(t - \tau_0),$$

and analyze the impact of τ_0 on the convolution result, discussing its interpretation in time-delayed systems.

You are free to choose a well-defined input signal $f(t)$ suitable for convolution, such as a unit step or a sine wave, and illustrate your results using both analytical expressions and plots (if desired).

Introduction

Convolution is a fundamental mathematical operation that represents how a system characterized by an impulse response modifies an input signal. In this report, we focus on the convolution of an arbitrary signal $f(t)$ with a rectangular kernel $h(t)$, a common smoothing and averaging filter. The rectangular kernel serves as a simple finite-duration window, and its convolution with various input signals reveals key behaviors such as transient responses, pulse spreading, and filtering effects.

The objective of this analysis is threefold:

- To derive closed-form expressions for the convolution $y(t) = (f * h)(t)$ for kernel durations $2T$ and different input functions.
- To examine how causality (restricting the kernel to $t \ge 0$) alters the output in practical, realizable systems.
- To explore the effect of a time delay τ_0 on the system response, providing insights into time-shifted or delayed systems in engineering applications.

Understanding these aspects helps build intuition for more complex filtering operations and lays the groundwork for applications in control systems, communications, and other areas where time-domain analysis is essential.

2 Problem Statement Solutions

2.1 Part(a) - Causal Kernel

In this scenario, the kernel is changed from a symmetric rectangular pulse to one that is non-zero only for $t > 0$, i.e., an asymmetric, causal kernel:

$$h(t) = \begin{cases} 1, & 0 \leq t \leq T \\ 0, & \text{otherwise} \end{cases}$$

2.1.1 Effects on Convolution Outputs

- The convolution integral is effectively limited to a one-sided window, i.e., it accumulates contributions only from past input values ($t - \tau \geq 0 \Rightarrow \tau \leq t$).
- This results in a causal system output, meaning the system begins to respond only after the input is applied - aligning with real-world system behavior.
- Compared to the symmetric kernel (which averages across past and future values of $f(t)$), the asymmetric kernel behaves like a moving average with a delay, emphasizing accumulated past effects and not anticipating future changes.

2.1.2 Significance

- It makes the system predictable and stable, as it responds only to known inputs.
- In real-time systems (e.g., filters, control loops, audio processors), causality is mandatory, so using a kernel defined for $t > 0$ ensures physical realizability.

2.2 Part(b) - Time Shifted Kernel

The kernel is now shifted in time by a constant delay τ_0. The new kernel becomes:

$$h(t) = \begin{cases} 1, & -T + \tau_0 \leq t \leq T + \tau_0 \\ 0, & \text{otherwise} \end{cases}$$

This is equivalent to replacing $h(t)$ with $h(t - \tau_0)$, i.e., delaying the kernel in time.

2.2.1 Effects on Convolution Output

Using the time-shift property of convolution, we have:

$$f(t) * h(t - \tau_0) = y(t - \tau_0)$$

This means the entire convolution output is delayed by τ_0. The response of the system begins later, and all features of the output (such as peaks, transitions) are shifted in time.

The plot confirms this shift: the convolution result with the delayed kernel appears as a right-shifted version of the original result (obtained with the symmetric kernel).

2.2.2 Significance

- The shifted kernel models such systems, where a delay in system response must be accounted for.
- Time-delays are common in communication, control, and physical systems where signals experience latency.

2.3 Graphical Interpretation

The following plots aid in understanding

- **Input Signal Plot:** Visual reference for $f(t)$.
- **Kernel Comparison Plot:** Shows differences in support (symmetric, asymmetric, shifted).
- **Convolution Output Plot:**
  - **Symmetric vs Asymmetric:** Highlights the effect of causal restriction.
  - **Original vs Shifted Kernel:** Visualizes time delay in response.

Special Case : Log

If the input signal is logarithmic - $\log(|t| + 1)$

- The logarithmic function grows slowly and diverges near $t = 0$, which can cause numerical instability during convolution.
- Its gradual slope leads to smoother output curves, especially when convolved with finite-width rectangular kernels.
- Care must be taken to handle the singularity at $t = 0$, often by using $\log(|t| + \epsilon)$ with small ϵ.

2.4 Conclusion

This analysis demonstrates how modifying a convolution kernel affects the system output

- A causal ($t > 0$) kernel ensures physical realizability and modifies the convolution to depend only on past inputs â acting like a one-sided moving average.
- A time-shifted kernel introduces a delay in system response, modeling latency in real-world systems.

2.5 Outputs

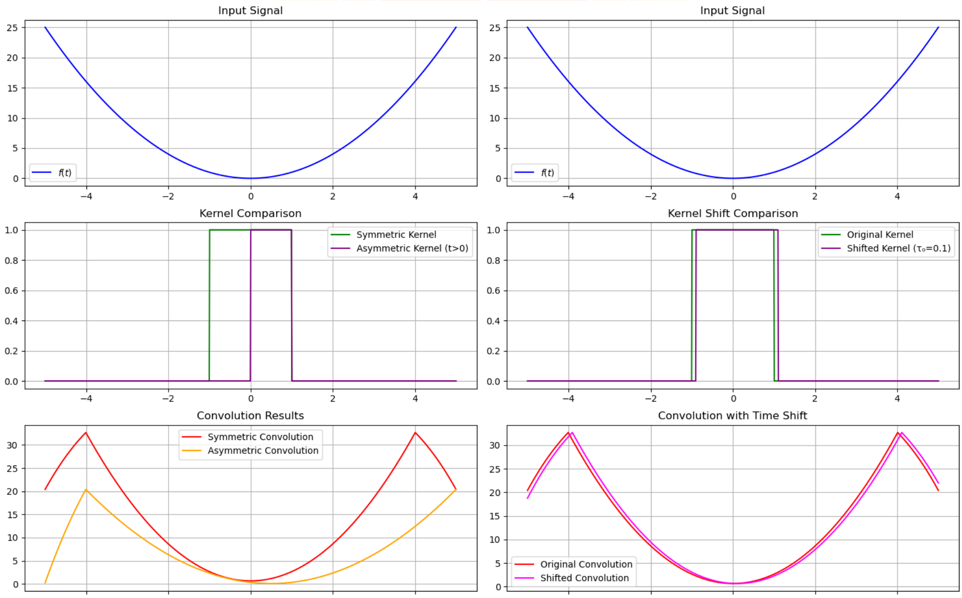


Fig. 2.1: Algebraic

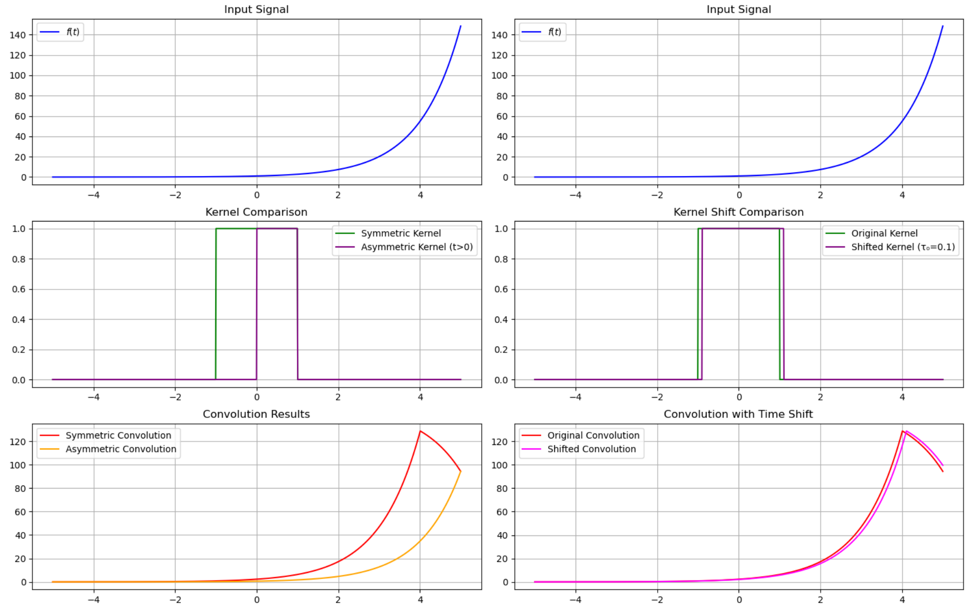


Fig. 2.2: Exponential

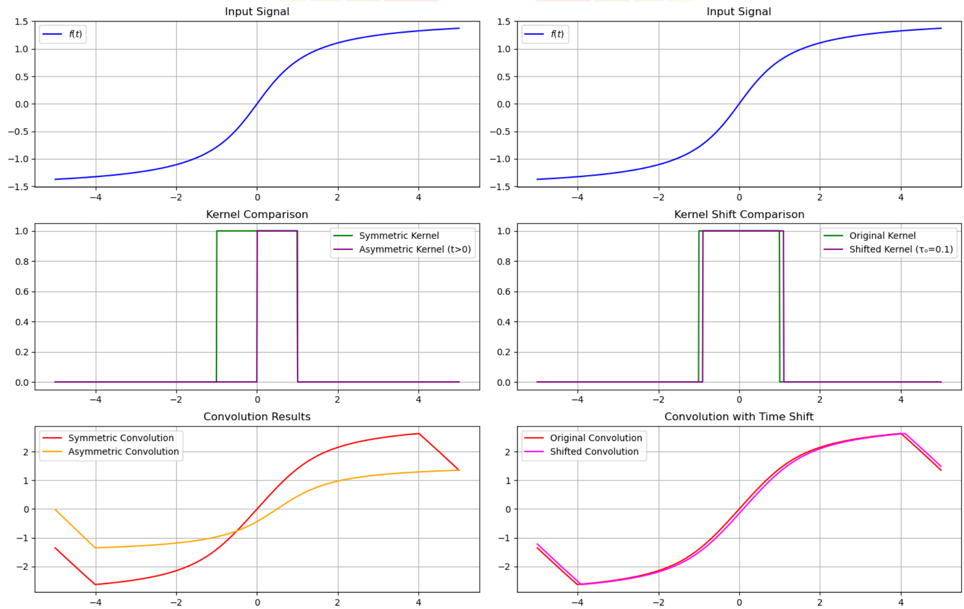


Fig. 2.3: Inverse-Trigonometric

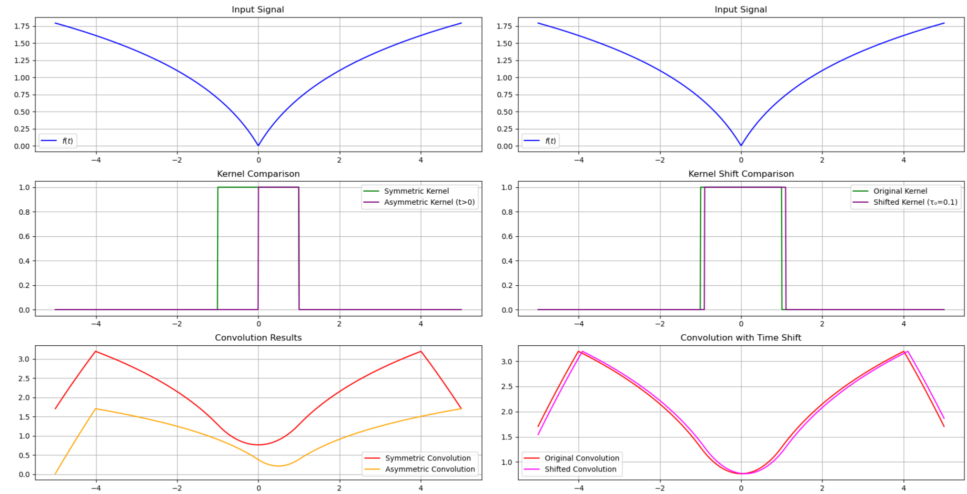


Fig. 2.4: Logarthmic

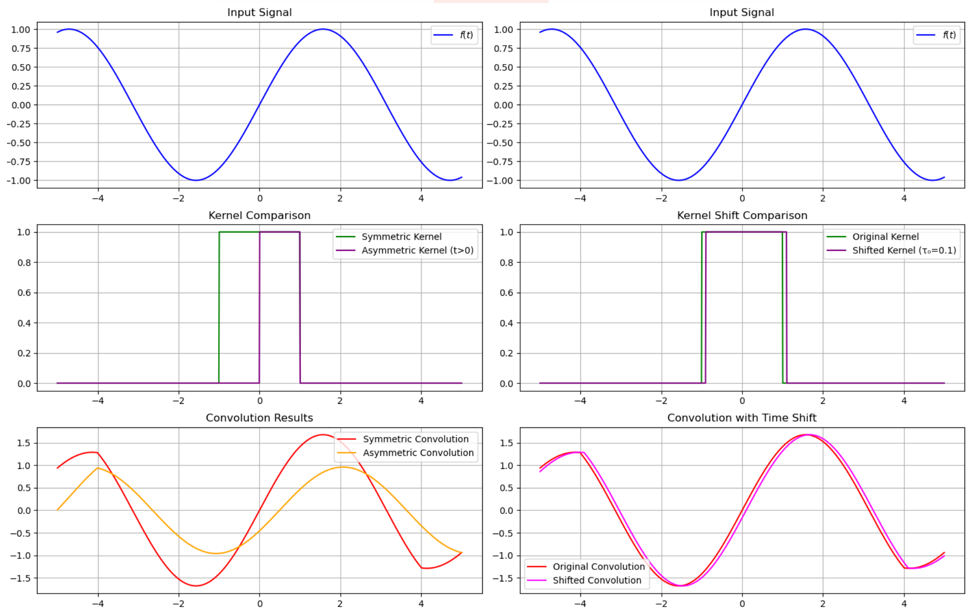


Fig. 2.5: Trigonometric

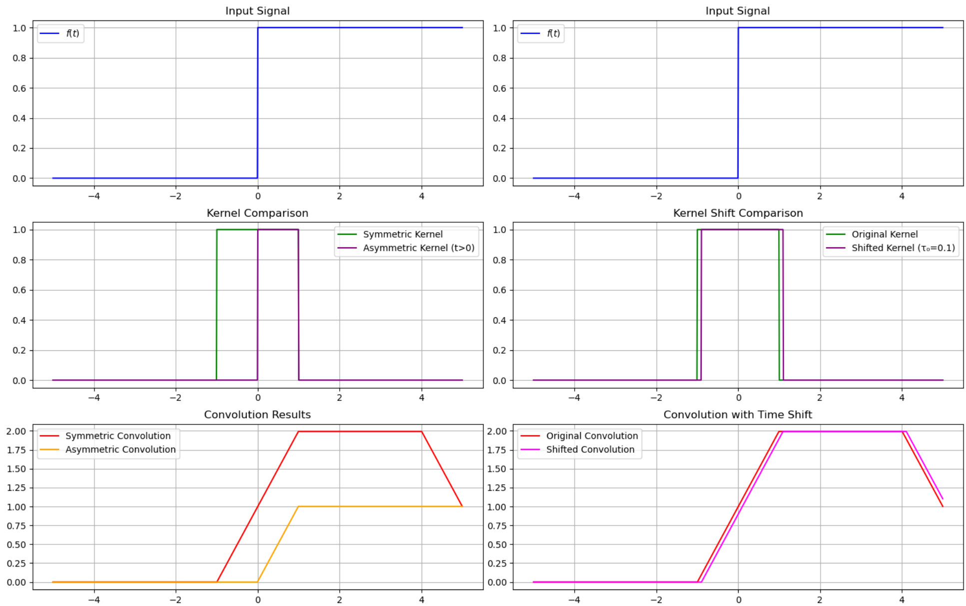


Fig. 2.6: Unit-Step

3 Convolution and Indefinite Integration

The convolution of two functions $f(t)$ and $g(t)$ is defined as:

$$(f * g)(t) = \int_0^t f(\tau)\, g(t-\tau)\, d\tau$$

This is a definite integral with limits from 0 to t. However, in some contexts (such as signal theory or systems starting from negative infinity), convolution may be expressed as an indefinite integral :

$$(f * g)(t) = \int_{-\infty}^t f(\tau)\, g(t-\tau)\, d\tau$$

Here, the integral accumulates the entire past input $f(\tau)$ weighted by the time-reversed response $g(t-\tau)$. This form is particularly useful in systems that do not start at $t = 0$ or when dealing with generalized functions like the unit step $u(t)$.

4 Discrete Convolution

4.1 Definition of Discrete Convolution

Given two discrete functions (or sequences), $f[n]$ (the input signal) and $g[n]$ (the kernel or filter), the **discrete convolution** $(f * g)[n]$ is defined as:

$$(f * g)[n] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[n-k] \tag{2.1}$$

In practice, since signals and kernels are finite, the limits of summation are adapted to the length of the sequences.

4.2 Step-by-Step Approach

1) **Flipping the Kernel:**
   - Discrete convolution involves flipping the kernel g around its center.
   - Mathematically, this corresponds to replacing $g[k]$ with $g[-k]$.
2) **Shifting the Kernel:**
   - The flipped kernel is shifted across the input signal.
   - For each position n, the kernel is aligned with the signal.
3) **Element-wise Multiplication:**
   - For each shift n, multiply overlapping elements of the signal and the flipped kernel.
4) **Summation:**
   - Sum the results of the element-wise multiplications to obtain the output at position n.

4.3 Example

Suppose we have a 1D input signal:

$$f = [1, 2, 3, 4]$$

and a kernel:

$$g = [1, 0, -1]$$

The steps are:

- Flip g: $[-1, 0, 1]$
- Slide and compute for each position (with appropriate padding).

Calculation:

$$(f * g)[0] = (0 \times -1) + (0 \times 0) + (1 \times 1) = 1$$

$$(f * g)[1] = (0 \times -1) + (1 \times 0) + (2 \times 1) = 2$$

$$(f * g)[2] = (1 \times -1) + (2 \times 0) + (3 \times 1) = 2$$

$$(f * g)[3] = (2 \times -1) + (3 \times 0) + (4 \times 1) = 2$$

$$(f * g)[4] = (3 \times -1) + (4 \times 0) + (0 \times 1) = -3$$

Thus, the result is:

$$[1, 2, 2, 2, -3]$$

4.4 Different Functions

The analysis examines five distinct input functions:

1) Algebraic function: $f[n] = n^2$
2) Exponential function: $f[n] = e^{n/20}$
3) Inverse trigonometric function: $f[n] = \arctan(n/10)$
4) Trigonometric function: $f[n] = \sin(n/10)$
5) Logarithmic function: $f[n] = \log(|n| + 1)$

Each function is convolved with a rectangular kernel defined as:

$$h[n] = \begin{cases} 1, & \text{if } -N \le n \le N \\ 0, & \text{otherwise} \end{cases}$$

where N is the half-width parameter, typically set to $N = 5$.

For each function, three convolution scenarios are examined:

1) Standard two-sided rectangular kernel
2) Half-sided (causal) kernel: $h[n] = \begin{cases} 1, & \text{if } 0 \le n \le N \\ 0, & \text{otherwise} \end{cases}$
3) Shifted kernel: $h[n - \tau_0]$ where $\tau_0 = 3$

The convolution is calculated manually using a direct implementation of the discrete convolution formula.

4.5 1. Algebraic Function Analysis

For the algebraic function $f[n] = n^2$, the convolution with a rectangular kernel has an analytical expression:

$$(f * h)[n] = (2N + 1)n^2 + \frac{N(N + 1)(2N + 1)}{3}$$

This shows that the convolution with a rectangular kernel:

- Scales the original quadratic function by a factor of $(2N + 1)$
- Adds a constant offset term $\frac{N(N+1)(2N+1)}{3}$

When using a half-sided kernel, the symmetry of the output is broken, resulting in an asymmetric averaging effect. When shifted by τ_0, the output is simply shifted along the time axis.

4.6 2. Exponential Function Analysis

For the exponential function $f[n] = e^{an}$ where $a = 1/20$, the convolution with a rectangular kernel has the analytical expression:

$$(f * h)[n] = e^{a(n-N)} \frac{1 - e^{a(2N+1)}}{1 - e^a}$$

This demonstrates that:

- The output remains an exponential function with the same rate parameter a
- The amplitude is scaled by a factor determined by the kernel width N
- There is a phase shift of N samples

The half-sided kernel makes the system causal but introduces distortion compared to the two-sided kernel. The shifted kernel introduces an additional delay of τ_0 samples.

4.7 3. Inverse Trigonometric Function Analysis

The inverse trigonometric function $f[n] = \arctan(n/10)$ represents a saturating non-linear function. The convolution with a rectangular kernel performs a moving average operation that smooths the transition regions.

Key observations:

- The convolution maintains the general shape of the arctangent function
- The smoothing effect is more pronounced in regions with higher curvature
- The half-sided kernel introduces asymmetry in the output
- The shifted kernel shifts the output function along the time axis

4.8 4. Trigonometric Function Analysis

For the trigonometric function $f[n] = \sin(\omega n)$ where $\omega = 1/10$, the convolution with a rectangular kernel has the analytical expression:

$$(f * h)[n] = \sin(\omega n)\frac{\sin(\omega(N + 0.5))}{\sin(\omega/2)}$$

This demonstrates that:

- The convolution acts as a low-pass filter
- The frequency response is determined by the ratio $\frac{\sin(\omega(N+0.5))}{\sin(\omega/2)}$
- Larger N values result in narrower main lobes in the frequency response
- The half-sided kernel introduces phase distortion
- The shifted kernel introduces a phase delay of $\omega\tau_0$

The frequency domain analysis shows that the rectangular kernel has a sinc-like frequency response, with nulls at frequencies $\omega = \frac{2\pi k}{2N+1}$ for non-zero integers k.

4.9 5. Logarithmic Function Analysis

The logarithmic function $f[n] = \log(|n| + 1)$ represents a slowly growing function with a singularity in its derivative at $n = 0$. The convolution with a rectangular kernel:

- Smooths the sharp transition at $n = 0$
- Preserves the general logarithmic growth pattern
- Demonstrates the kernel's averaging effect on non-linear functions

The half-sided and shifted kernel variations show similar effects as observed with the other functions.

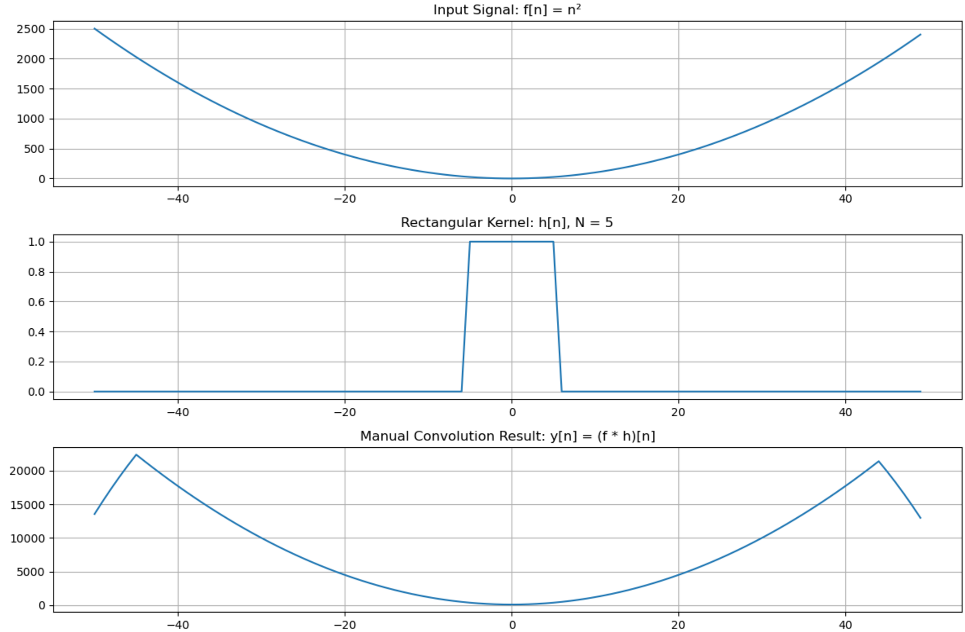
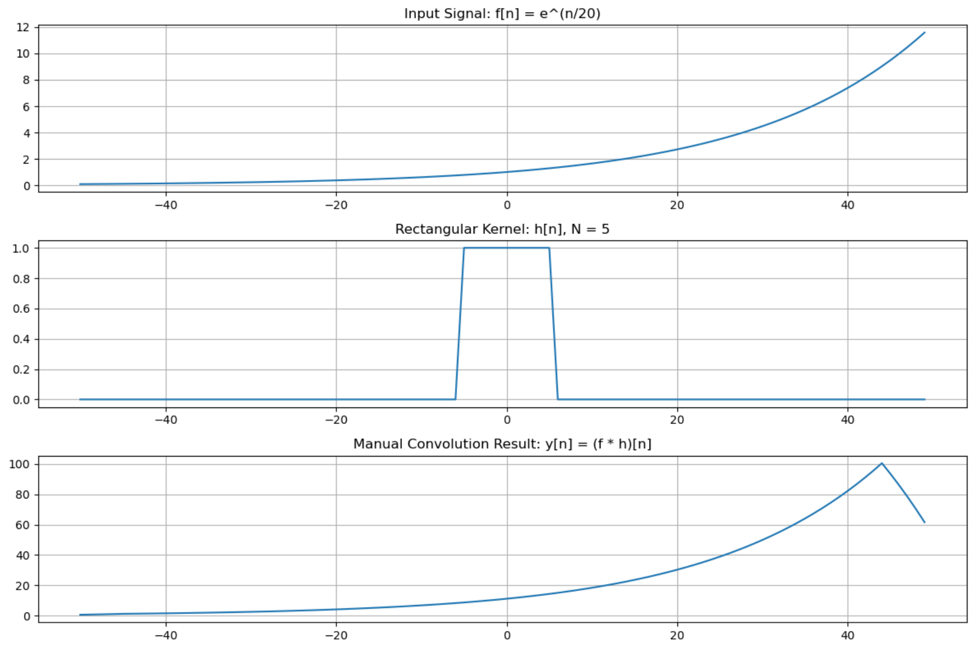


Fig. 3.1: Convolution of n^2

4.10 Outputs

4.10.1 Algebraic function : $f(t) : t^2$

4.10.2 Exponential function : $f(t) : e^t$



4.10.3 Inverse trigonometric function : $f(t)$: arctan($n/10$)

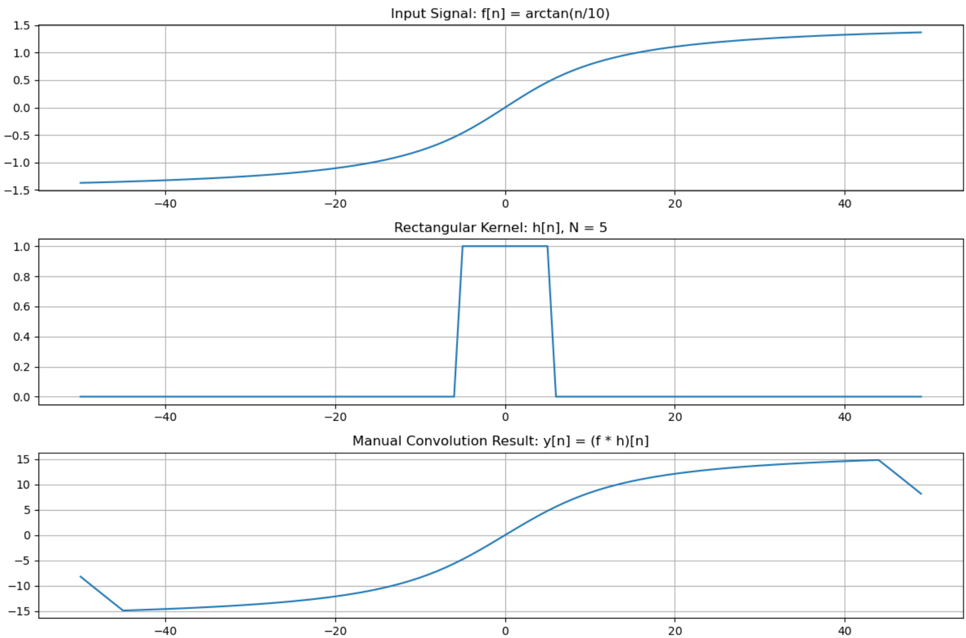


Fig. 3.3: Convolution of n^2

4.10.4 Trigonometric function : $f(t)$: $sin(n/10)$

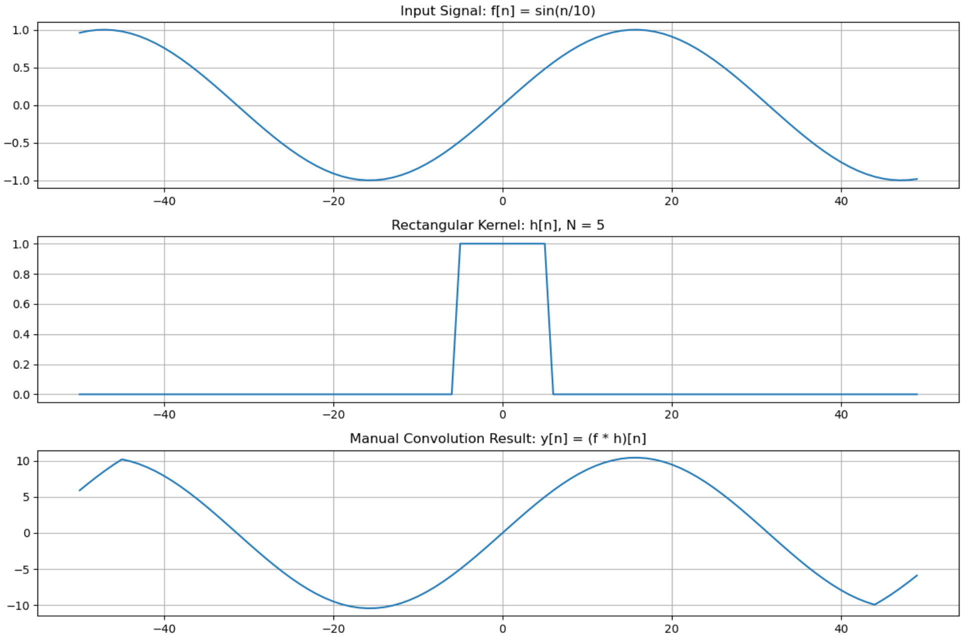


Fig. 3.4: Convolution of n^2

4.10.5 Logarthmic function : $f(t) : ln(1 + t)$

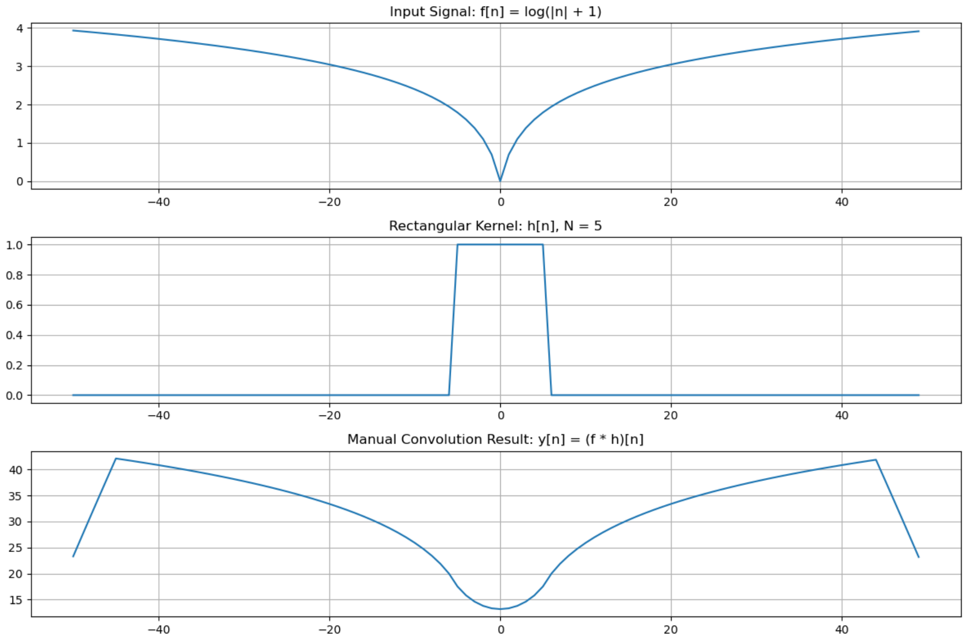


Fig. 3.5: Convolution of n^2

5 Circular Convolution

5.1 Introduction

- Circular convolution is a special case of periodic convolution, which is the convolution of two periodic functions that have the same period.
- Circular convolution plays an important role in maximizing the efficiency of a certain kind of common filtering operation.

5.2 Linear Convolution

We know that

$$y[n] = \sum_{-\infty}^{\infty} x_1[k] \quad x_2[n-k]$$

For $x_2[k]$

- On flipping the signal k changes to -k = $x_2[-k]$
- Shifting the signal to the right by n samples k changes to k-n resulting in

$$x_2[-k] = x_2[-(k-n)] = x_2[n-k]$$

5.3 Circular Convolution

If both the signals are periodic

$$x_1[N+n] = x_1[n]$$

$$x_2[N+n] = x_2[n]$$

then

$$x_1[k]x_2[n_0-k]$$

y[n] will also be a periodic signal with period N
The periodic convolution of two T-periodic functions $h_T(t)$ and $x_T(t)$ can be defined as

$$\int_{t_0}^{t_0+T} h_T(t)\cdot x_T(t-\tau)d\tau$$

then

$$\int_{t_0}^{t_0+T} h_T(\tau)\cdot x_T(t-\tau)\,d\tau = \int_{-\infty}^{\infty} h(\tau)\cdot x_T(t-\tau)\,d\tau \triangleq (h*x_T)(t) = (x*h_T)(t).$$

for circular convolution y[n] changes to

$$y[n] = \sum_{k=0}^{N-1} x_1[k] \quad x_2[n-k]$$

where

- y[n] is periodic with period N
- n-k can be replaced by $< n-k >_N$ (n-k mod N)
- "Circular" Convolution:

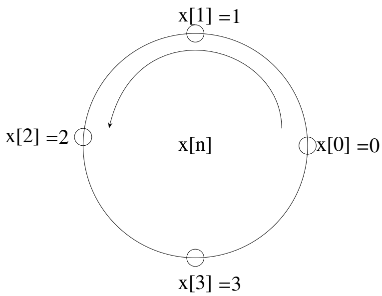
$$\tilde{y}[n] = \tilde{x}_1[n] \circledast \tilde{x}_2[n]$$

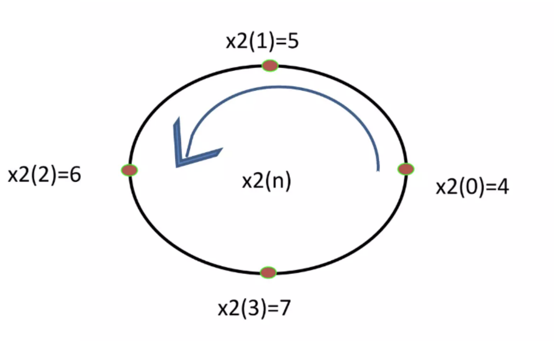
$$\tilde{y}[n] \stackrel{\text{def}}{=} \sum_{k=0}^{N-1} \tilde{x}_1[k]\,\tilde{x}_2[(n-k)_N] \quad \text{for } n = 0, 1, \ldots, N-1$$

5.3.1 Circular Representation

for a signal x[n]

- $x_1[n] = [0, 1, 2, 3]$
- $x_2[n] = 4, 5, 6, 7$





5.4 Linear vs Circular Convolution

- Linear Convolution
  - Computes how much two signals overlap as one slides over the other.
  - No periodicity, each sample is treated distinctly, with zero values outside signal range.
  - Used in LTI system analysis, filtering, and modeling physical systems.
- Circular Convolution
  - Assumes signals are periodic with period N, so indices wrap around using modulo.
  - Arises in DFT-based processing, where time-domain multiplication corresponds to frequency-domain convolution.
  - Requires zero-padding to avoid aliasing or overlap-add issues.

Aspect	Linear Convolution	Circular Convolution
Domain	Continuous or discrete	Only discrete (digital signals)
Definition	$(y*h)[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k]$	$y[n] = \sum_{k=0}^{N-1} x[k] \cdot h[(n-k) \bmod N]$
Signal Extension	Zero-padding (non-periodic)	Periodic extension (wraps around)
Length of Output	$L + M - 1$, for input lengths L and M	Same as the length N, usually $\max(L, M)$ or predefined
Frequency Domain	Multiplication in continuous/discrete Fourier Transform (FT/DTFT)	Multiplication in Discrete Fourier Transform (DFT)
Applications	Physical systems, filtering, general signal processing	Fast convolution via FFT, circular buffers, DFT-based systems
Artifacts	Natural result of signal overlap	Wrap-around distortion if zero-padding is not used

5.4.1 Examples

Input Signals :

- Signal A : Rectangular pulse of 6 ones

$$x[n] = [1, 1, 1, 1, 1, 1]$$

- Signal B : Rectangular pulse of 4 ones

$$h[n] = [1, 1, 1, 1]$$

Computing the linear convolution

$$y[n] = x[n] * h[n]$$

The result slides $h[n]$ over $x[n]$, summing overlaps at each shift.
Output :

- Length = 6+4-1 = 9
- Shape : ramp up and ramp down
- Values :
  - y[n] = [1,2,3,4,4,4,3,2,1]

Computing the Circular Convolution

- Both the signals are defined with length N=7

$$y[n] = x[n] * h[n]$$

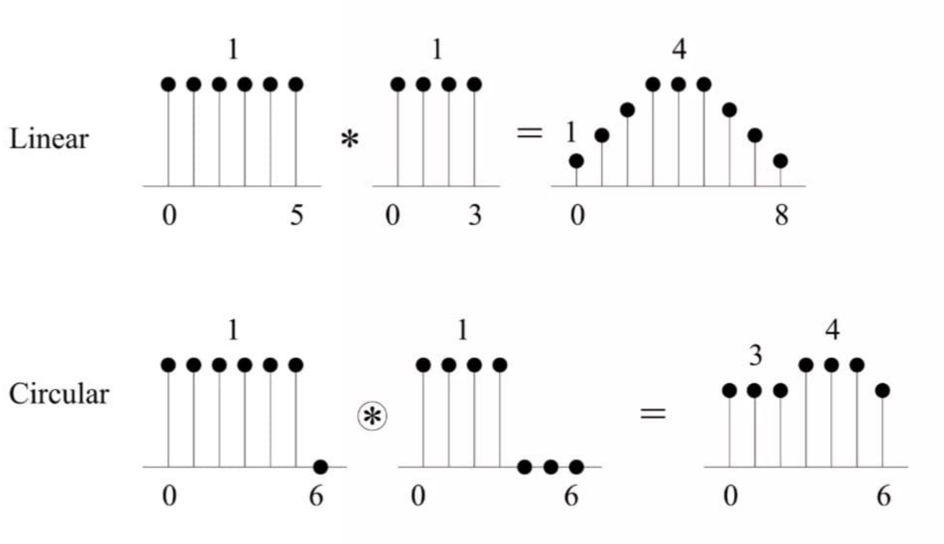
The signal modifies accordingly to

- x[n] = [1,1,1,1,1,1,0]
- h[n] = [1,1,1,1,0,0,0]

When the convolution exceeds length 7, values wrap to the beginning.
Output :

- y[n] = [3,4,4,4,4,3,2]

This matches circular convolution behavior



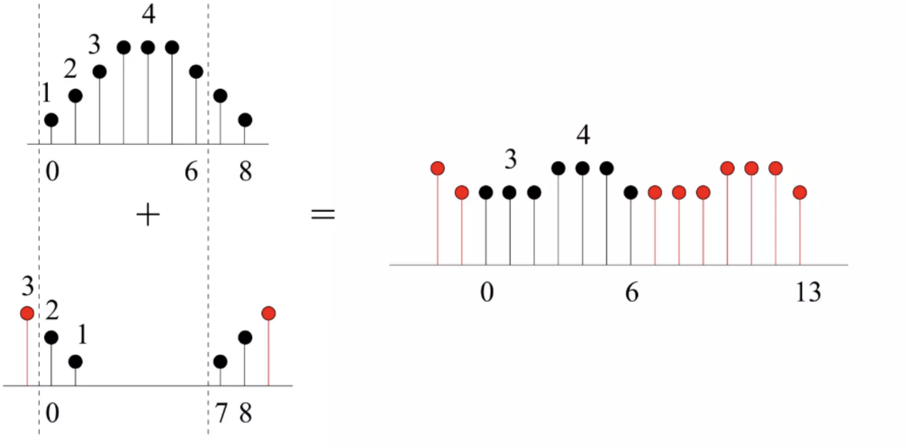
5.5 Relationship between L&C Convolution

- If x[n] has length P and h[n] has length Q then y[n] has length P+Q-1
- $N > max(P, Q)$, in general

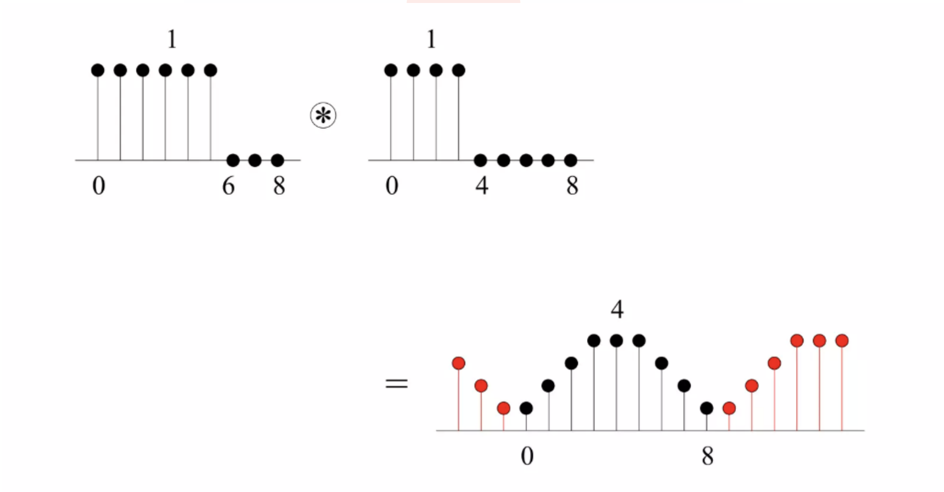
$$\overline{x_1}[n] * \overline{x_2}[n] \neq x_1[n] * x_2[n]$$

- If $N \geq P + Q - 1$

$$\overline{x_1}[n] * \overline{x_2}[n] = x_1[n] * x_2[n]$$



- If $N \geq 9$ one period of circular convolution will be equal to linear convolution



An algorithm, called the **Fast Fourier Transform (FFT)** is more efficient to compute the circular convolution

$$x_1[n] * x_2[n] \leftrightarrow X_1[k]X_2[k]$$

$$y[n] = FFT^{-1}(X_1[k]X_2[k])$$

5.6 Rectangular Kernel

Circular convolution with a rectangular kernel offers a computationally efficient alternative to linear convolution, especially when working with periodic signals or implementing filtering in frequency domain.

$$h[n] = \begin{cases} 1, & 0 \leq n < T \\ 0, & \text{otherwise} \end{cases}$$

Applying circular convolution with a signal x[n]

$$y[n] = \sum_{k=0}^{N-1} x[k] \cdot h[(n-k) \bmod N]$$

This assumes both the signals as periodic with period N, so the rectangular kernel is treated as a periodic square wave.

- Using a rectangular kernel as the impulse response $h[n]$ simulates a finite integration window.
- Computing circular convolution using FFT reduces the complexity from $O(N^2)$ to $O(NlogN)$

5.7 Input Signal analysis on CC

For a Rectangular kernel

$$h[n] = \begin{cases} 1, & \text{if } 0 \leq n < T \\ 0, & \text{otherwise} \end{cases}$$

Circular convolution for period N

$$y[n] = (x \circledast h)[n] = \sum_{k=0}^{N-1} x[k] \cdot h[(n-k) \bmod N]$$

5.7.1 Inverse Trigonometric

$$x[n] = \tan^{-1} x$$

Properties

- $lim_{n\to\infty} \tan^{-1} n = \frac{\pi}{2}$
- $\frac{d}{dn} \tan^{-1} n = \frac{1}{1+n^2}$

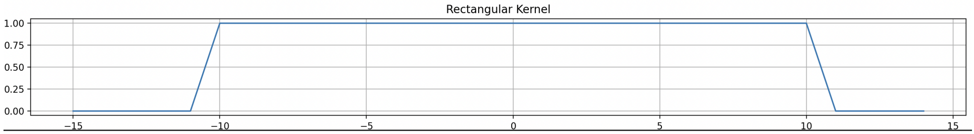
Applying CC

$$y[n] = \sum_{k=0}^{T-1} \tan^{-1}((n-k) \bmod N)$$

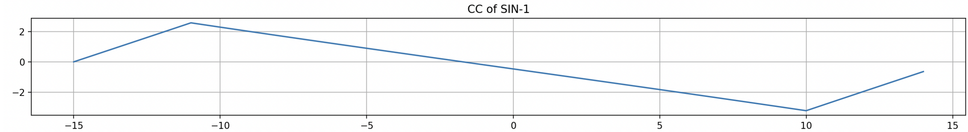
- The input is smooth, so the rectangular kernel further smooths and slightly lags the signal.
- Circular convolution preserves the shape and wrap-around distortion is small unless N is small or T is large.

Outputs

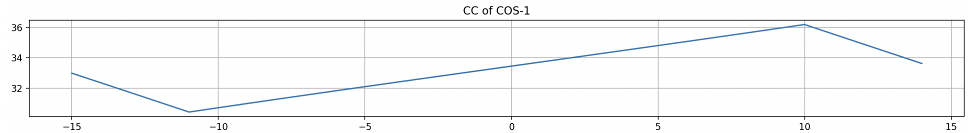
Rectangular Kernel



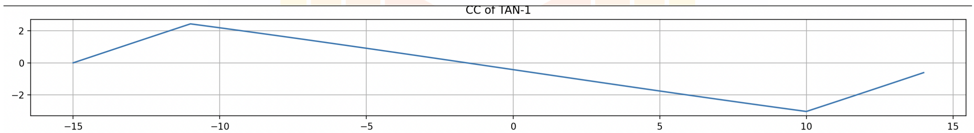
Inverse Sin



Inverse Cos



Inverse Tan



5.7.2 Algebraic

$$x[n] = n^2$$

Properties

- Grows quadratically, second order polynomial
- $x[n+1] - x[n] = 2n + 1$, difference increases linearly

Applying CC

$$y[n] = \sum_{k=0}^{T-1} ((n-k) \bmod N)^2$$

Example
Given: $x[n] = [0, 1, 4, 9]$, $h[n] = [1, 1]$

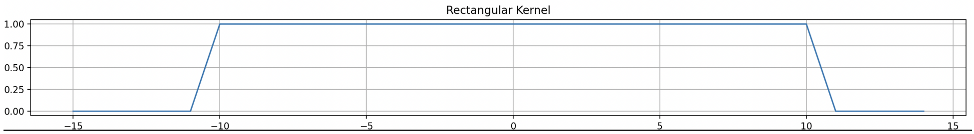
$$y[0] = x[0] \cdot h[0] + x[3] \cdot h[1] = 0 \cdot 1 + 9 \cdot 1 = 0 + 9 = 9$$

In linear case y[0] = 0

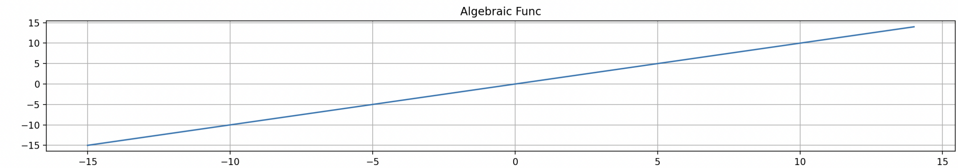
- Circular convolution causes false high values unless zero-padded.
- When values from the end (large n) wrap to the start (small n), distortion occurs.
- Use $N \geq L + T - 1$ to avoid aliasing.

Outputs

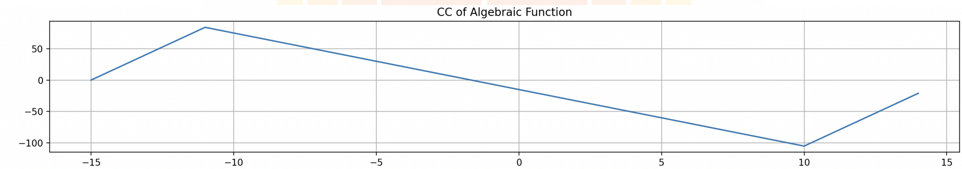
Rectangular Kernel



Algebraic Function



CC of Algebraic Function



5.7.3 Exponential

$$x[n] = e^{an}$$

Properties

- Grows faster than any polynomial
- Ratio between the successive terms

$$\frac{x[n+1]}{x[n]} = e^{a}$$

Applying CC

$$y[n] = \sum_{k=0}^{T-1} e^{a((n-k) \mod N)}$$

Example

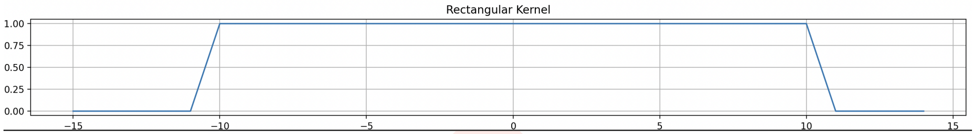
$$x[n] = [1, e, e^2, e^3], \quad h[n] = [1, 1], \quad N = 4$$

$$y[0] = x[0]h[0] + x[3]h[1] = 1 + e^3 \approx 1 + 20.1 \approx 21$$

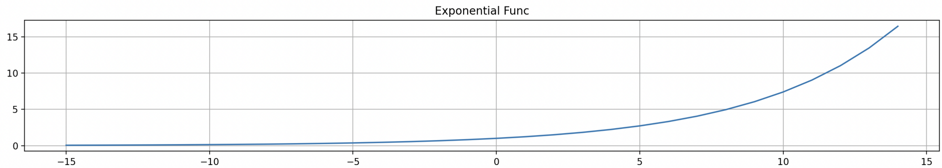
- Wrap-around introduces large high values at low indices due to exponential tail folding back
- Zero-padding is essential here
- Otherwise, circular convolution produces misleadingly large results

Outputs

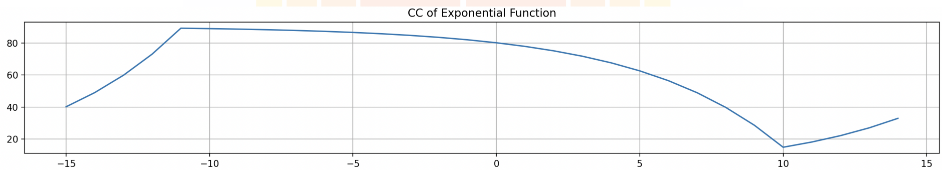
Rectangular Kernel



Exponential Function



CC of Exponential



5.7.4 Trigonometric

$$x[n] = \cos(\omega n)$$

Properties

- Periodic : $\cos(\omega n)$ repeats after every $\frac{2\pi}{\omega}$

Frequency Domain

$$Y[k] = X[k] \cdot H[k]$$

Where

- $$X[k] : \text{DFT of cosine} \Rightarrow \text{two impulses}$$

- $$H[k] : \text{DFT of rectangular kernel}$$

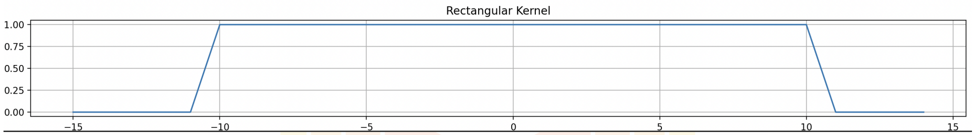
$$H[k] = \sum_{n=0}^{T-1} e^{-j\frac{2\pi}{N}kn} = e^{-j\pi\frac{(T-1)}{N}k} \cdot \frac{\sin\left(\pi\frac{T}{N}k\right)}{\sin\left(\pi\frac{1}{N}k\right)}$$

Acts as a LPF

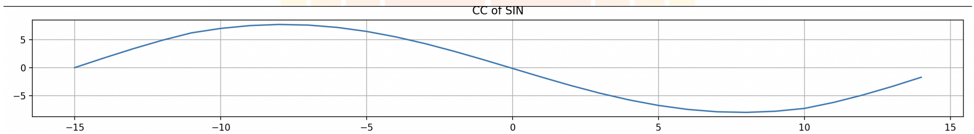
- High-frequency components are attenuated.
- Convolution with rectangular kernel ensures frequency smoothing
- Circular convolution is ideal here, especially when input is periodic

Outputs

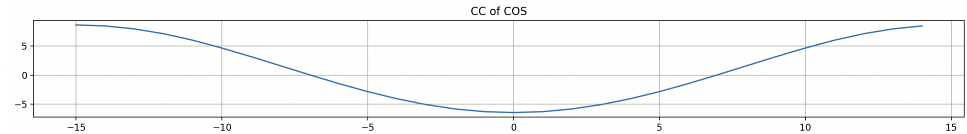
Rectangular Kernel



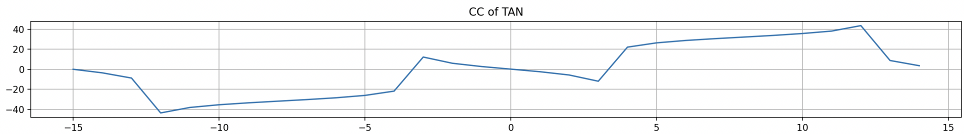
Sin



Cos



Tan



5.7.5 Logarithmic

$$x[n] = \log(n+1)$$

Properties

- Slow increase
- Difference between two successive terms

$$x[n+1] - x[n] = \log\frac{n+2}{n+1}$$

Applying CC

$$y[n] = \sum_{k=0}^{T-1} \log\left(((n-k) \bmod N) + 1\right)$$

Example

$$x[n] = [0, \log 2, \log 3, \log 4], \quad h[n] = [1, 1]$$

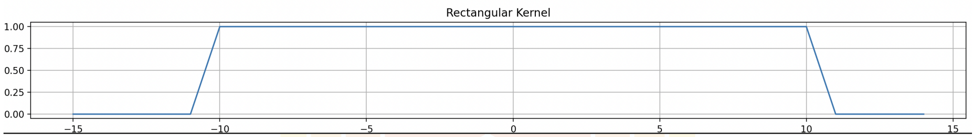
$$y[0] = \log(1) + \log(4) = 0 + \log(4) = \log(4)$$

This wrap-around should not happen in linear case

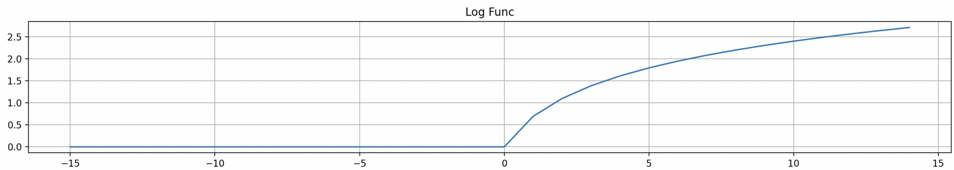
- Distortion is small in this case unless window is large.
- Wrap-around causes mild distortion, especially near $n = 0$

Outputs

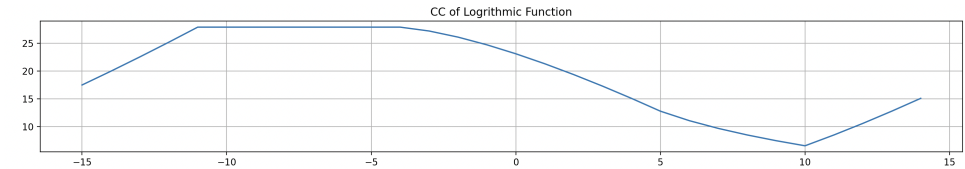
Rectangular Kernel



Log Function



CC of Log



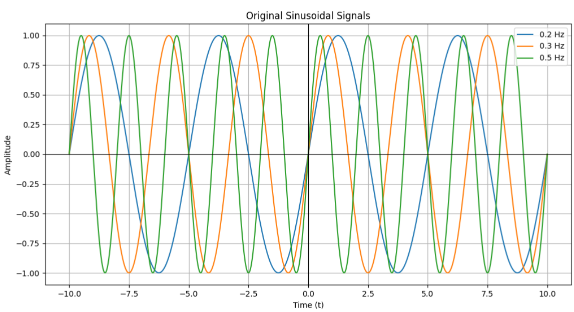
6 Spectral Analysis

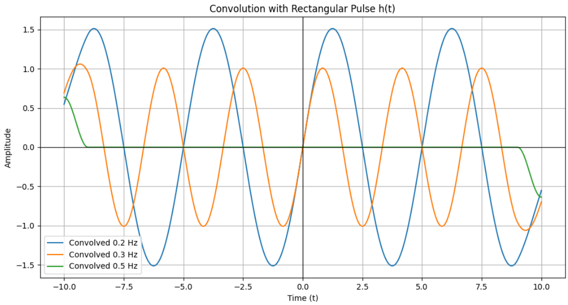
This section explores the spectral characteristics of the convolution between a fixed kernel $h(t)$ specifically a rectangular function and various types of input functions $f(t)$. Spectral analysis provides insight into how the frequency content of a signal is altered by convolution, particularly in terms of filtering effects. In this study, the convolution is analyzed across different categories of functions: trigonometric, algebraic, exponential, logarithmic, and inverse trigonometric.

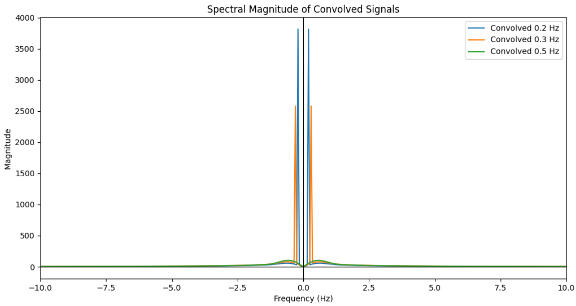
By examining the frequency domain representations, we observe that the rectangular kernel behaves as a low-pass filter attenuating high frequency components and allowing low frequencies to pass. Trigonometric inputs, which have inherent oscillatory components, produce distinct peaks at corresponding frequencies in the spectral plots. In contrast, functions like algebraic or exponential which lack significant oscillations primarily exhibit energy concentrated at zero frequency, resulting in a dominant peak at the origin. This comparative analysis helps in understanding how different functional forms interact with a filter in both time and frequency domains.

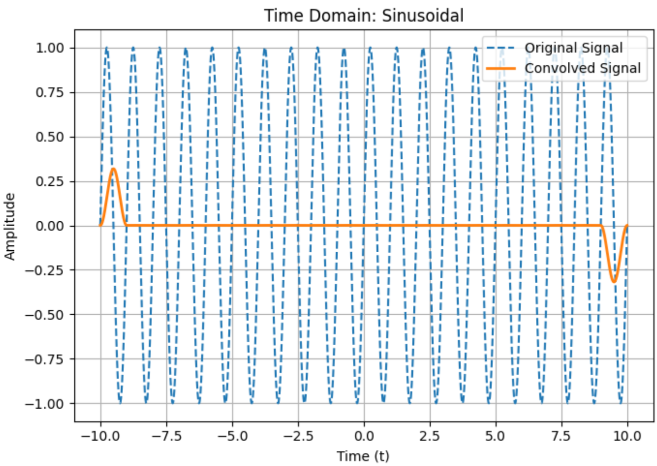
Spectral Analysis of convolution of $h(t)$ with various $f(t)$

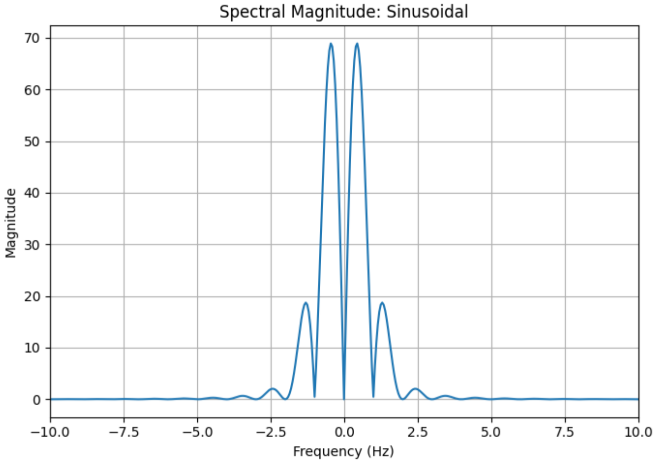
6.1 Trigonometric Functions




Convolution with Rectangular Pulse h(t)
Convolved 0.2 Hz
Convolved 0.3 Hz
Convolved 0.5 Hz
Amplitude
Time (t)

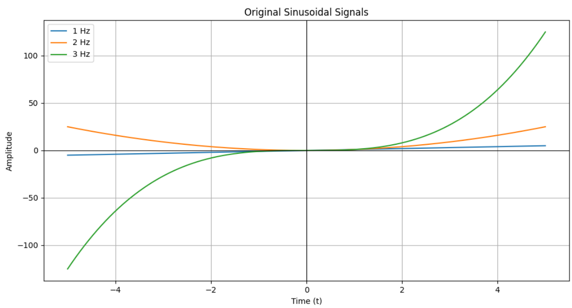

Spectral Magnitude of Convolved Signals
Convolved 0.2 Hz
Convolved 0.3 Hz
Convolved 0.5 Hz
Magnitude
Frequency (Hz)

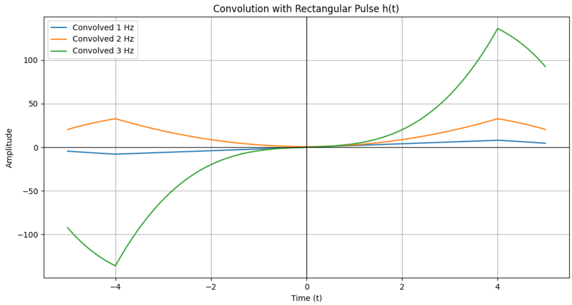

Time Domain: Sinusoidal
Original Signal
Convolved Signal
Amplitude
Time (t)

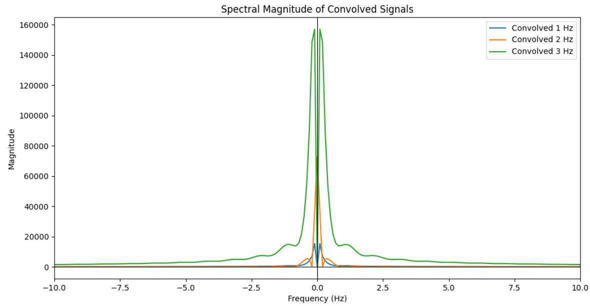


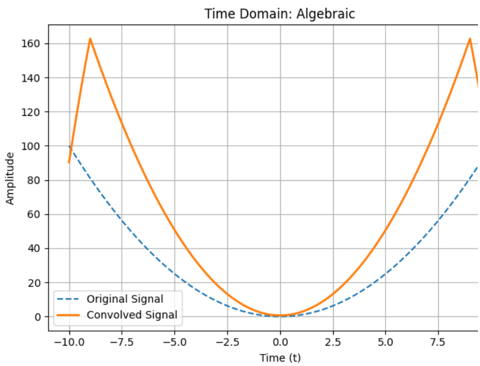
The rectangular kernel acts as a low-pass filter by scaling the amplitudes of higher frequencies in convolution. Peaks in the spectral analysis plot can be seen at frequencies of the trigonometric functions as they oscillate with that frequency.

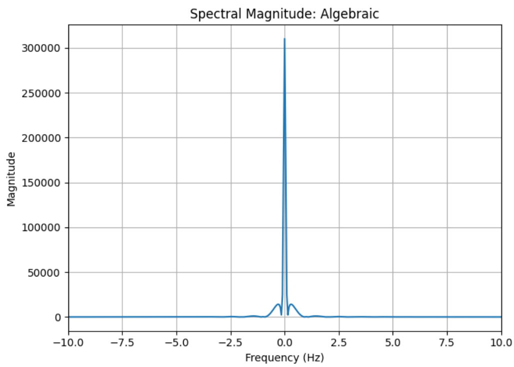
6.2 *Algebraic functions*





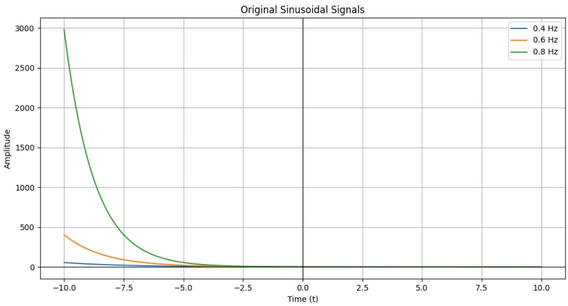


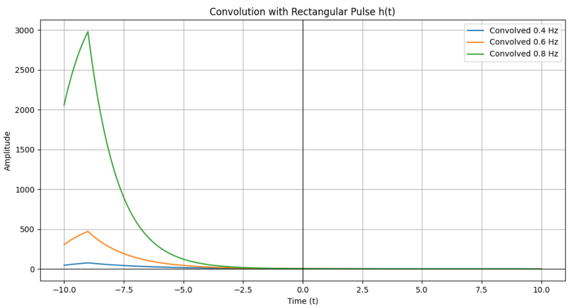


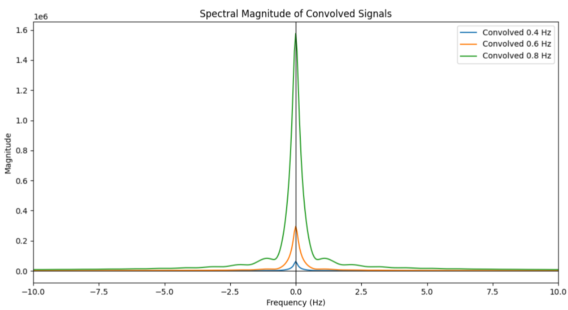


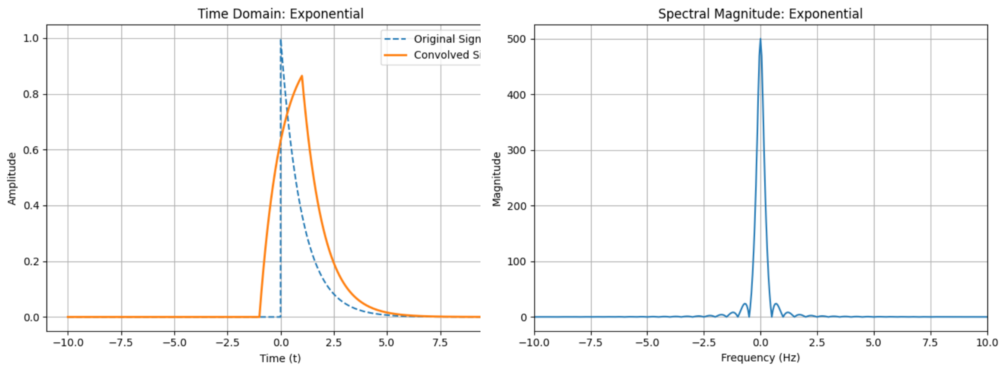
Peak in spectral Analysis can be seen at frequency=0 as no oscillation

6.3 Exponential Functions



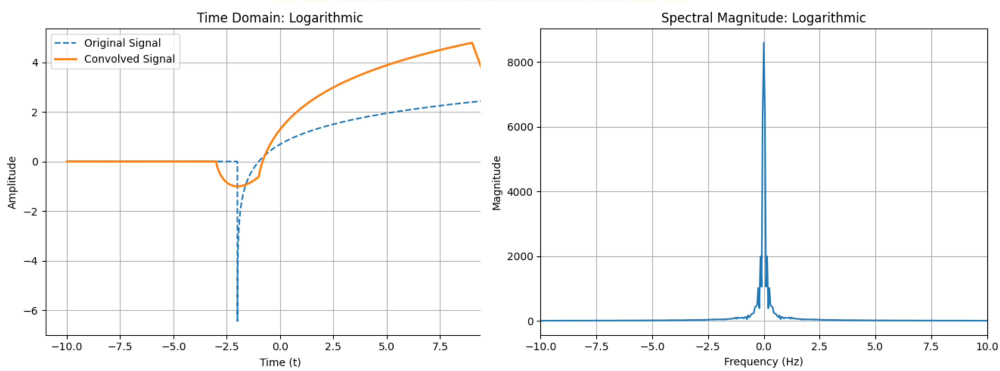






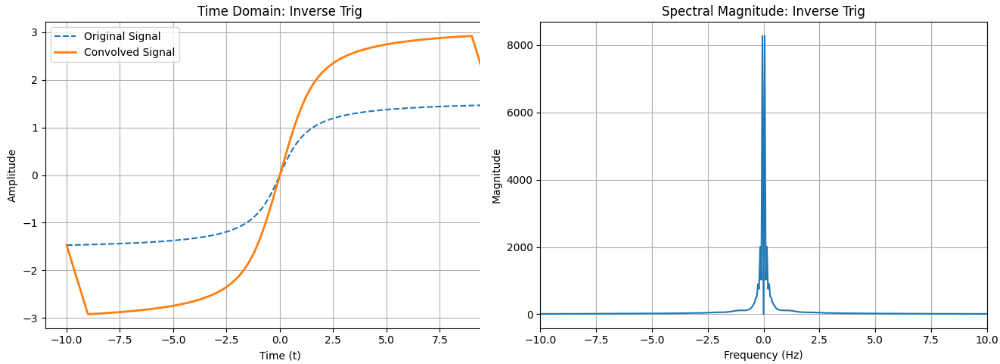
Peak in spectral Analysis can be seen at frequency=0 as no oscillation

6.4 Logarithmic Functions



Peak in spectral Analysis can be seen at frequency=0 as no oscillation

6.5 Inverse Tigonomentric Functions



Peak in spectral Analysis can be seen at frequency=0 as no oscillation

7 Analysis Using Modified and Shifted Kernel

The **original symmetric kernel** $h(t)$ defined as:

$$h(t) = \begin{cases} 1, & \text{for } -T \leq t \leq T \\ 0, & \text{otherwise} \end{cases}$$

7.1 Algebraic Functions

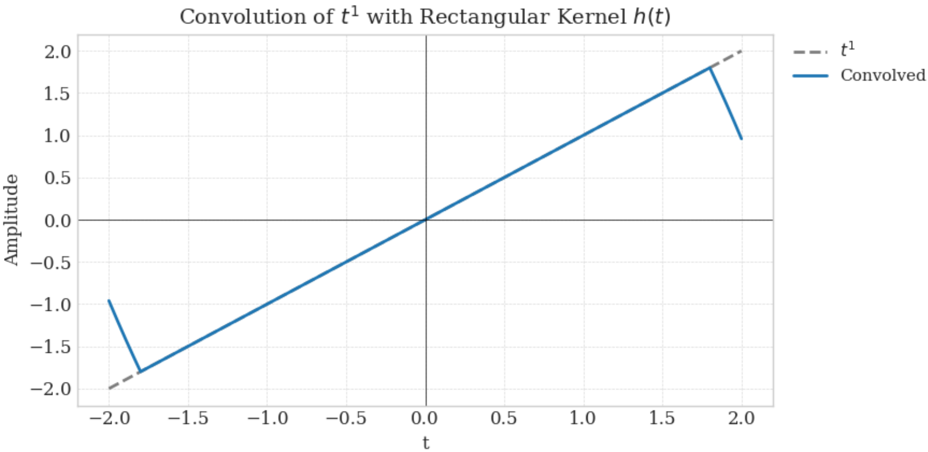
7.1.1 Definition of f(t)

$$f(t) = t^1, t^2, t^3, t^{-1}, t^{0.5}$$

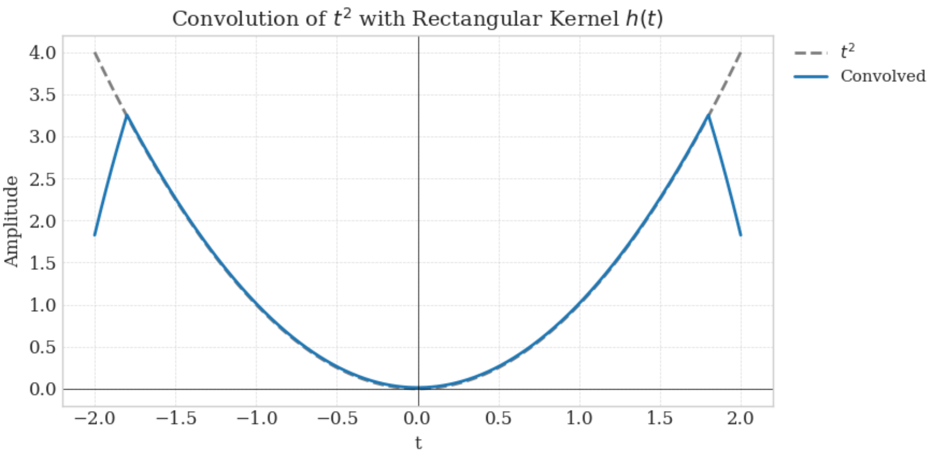
7.1.2 Original Kernel

When convolved with algebraic functions such as t, t^2, t^3, $\frac{1}{t}$, and t, the result is a smoothed version of the original function. The symmetric kernel includes both past and future values around time t, and this convolution generally retains the central trend of the original function.

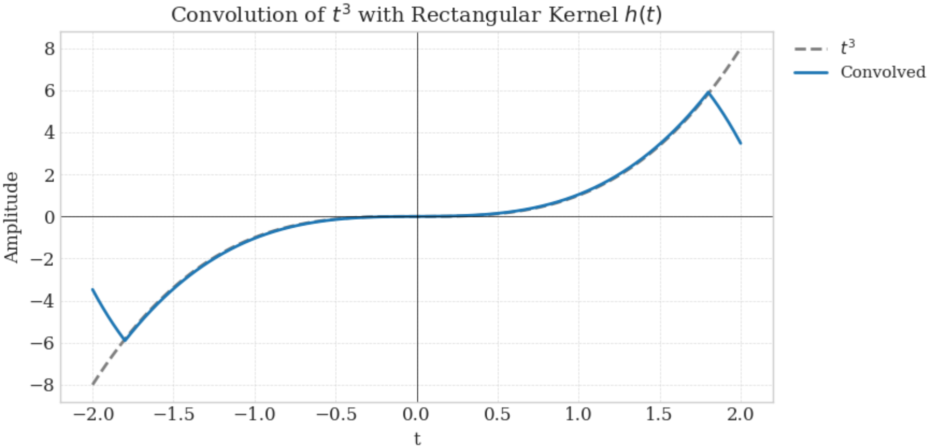
- The kernel acts as a moving average, smoothing the values of the algebraic functions.
- For $\frac{1}{t}$, the convolution avoids singularities at $t = 0$ by treating them appropriately (e.g., using zero-padding).
- The convolution result remains symmetric, preserving the overall characteristics of the algebraic functions.
- For t^2 and t^3, the smoothed result shows a gentle curve.
- For $\frac{1}{t}$, the singularity at $t = 0$ is handled gracefully by the convolution.

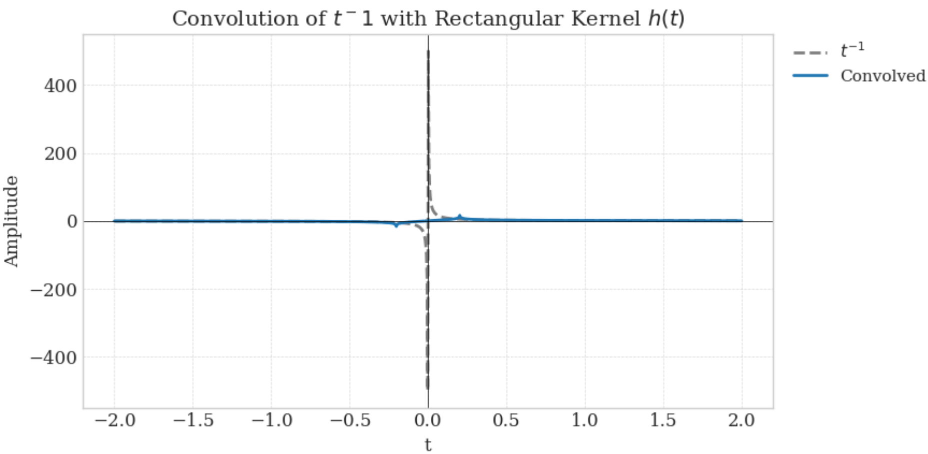


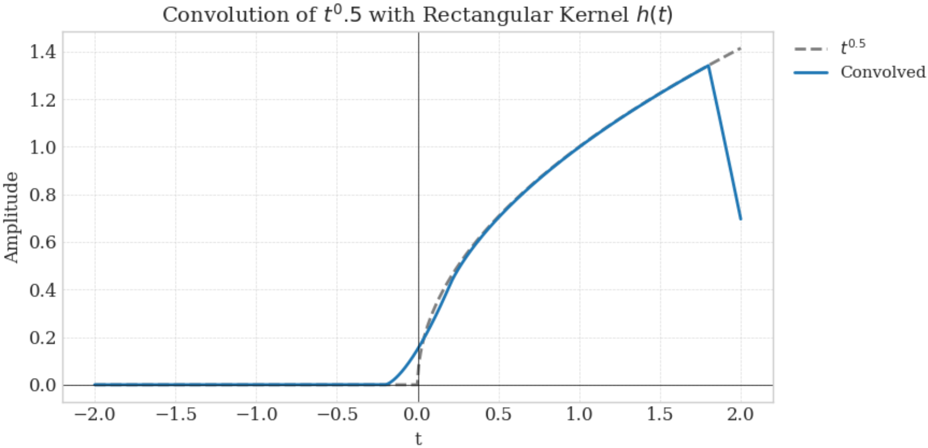
Convolution of t^1 with Rectangular Kernel $h(t)$
Amplitude
t
t^1
Convolved
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0



Convolution of t^2 with Rectangular Kernel $h(t)$
Amplitude
t
t^2
Convolved
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Convolution of t^3 with Rectangular Kernel $h(t)$
Amplitude
t
t^3
Convolved


Convolution of t^-1 with Rectangular Kernel $h(t)$
Amplitude
t
t^{-1}
Convolved



7.1.3 *Causal Kernel*

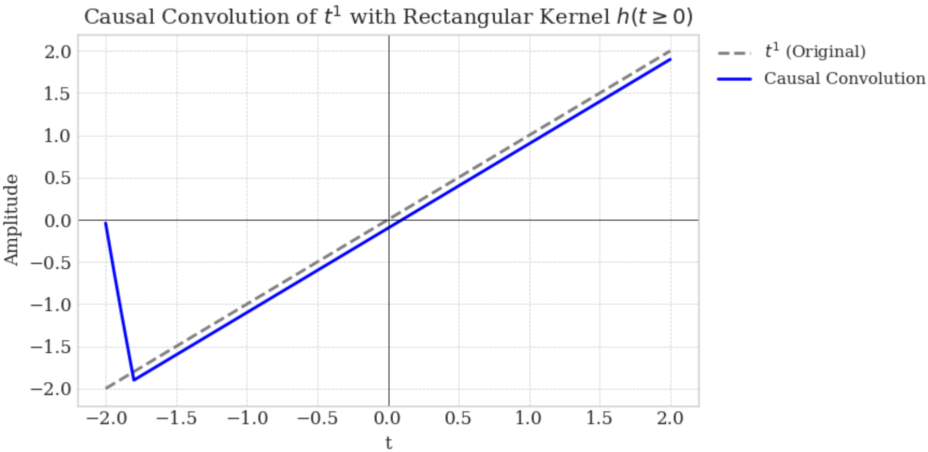
Causal convolution refers to convolution with a kernel $h(t)$ defined only for $t \geq 0$. This models real-world physical systems that cannot respond before an input is applied - i.e., the system is non-anticipatory or causal.

- In causal systems, the output at any time t depends only on the present and past values of the input - not the future.
- The kernel $h(t)$ is zero for $t < 0$, ensuring that the convolution integral only considers past and present inputs:

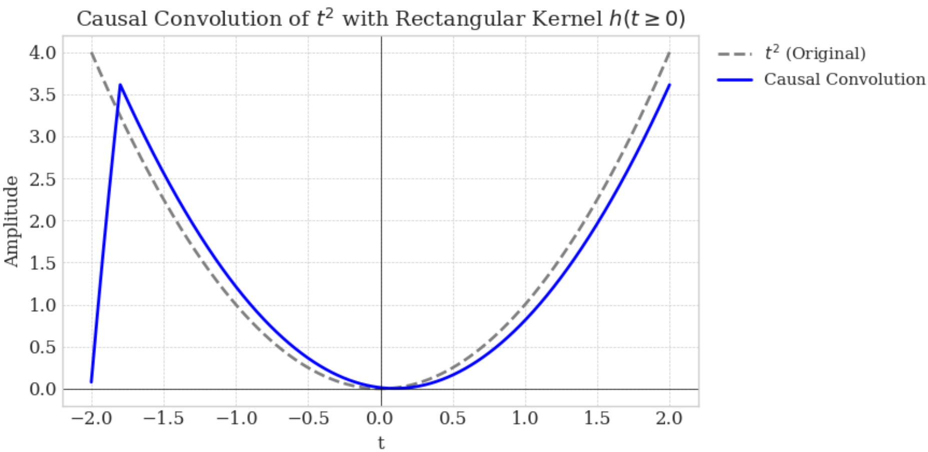
$$(f * h)(t) = \int_0^t f(\tau)\, h(t - \tau)\, d\tau$$

- This is especially relevant for real-time signal processing, control systems, and physical modeling where future inputs are unknown.

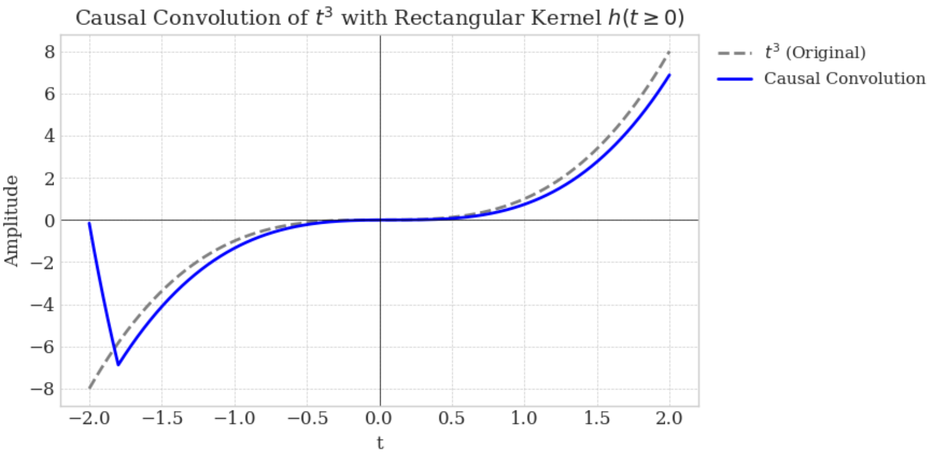
- **Asymmetry in Output:** Unlike the symmetric kernel (e.g., centered at $t = 0$), causal convolution distorts the original signal, especially near $t = 0$, because there is no contribution from $t < 0$.
- **Smoothing and Delay:** The algebraic input functions (like t, t^2) are smoothed and shifted slightly rightward, though not as cleanly as a purely shifted kernel.
- **Suppression of Singularities:** For functions like t^{-1}, the convolution smooths out the singular behavior at $t = 0$, creating a finite and regular output.

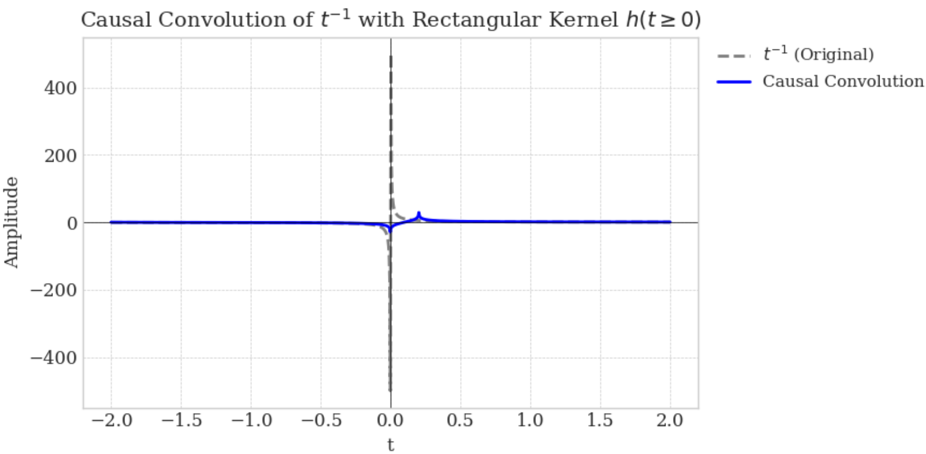


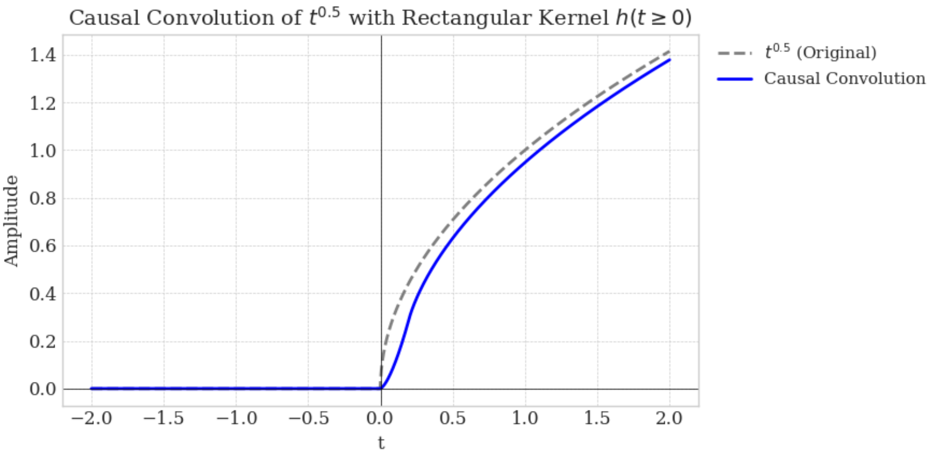
Causal Convolution of t^1 with Rectangular Kernel $h(t \geq 0)$
Amplitude
t
t^1 (Original)
Causal Convolution
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0



Causal Convolution of t^2 with Rectangular Kernel $h(t \geq 0)$
Amplitude
t
t^2 (Original)
Causal Convolution
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Causal Convolution of t^3 with Rectangular Kernel $h(t \geq 0)$
t^3 (Original)
Causal Convolution
Amplitude
t
8
6
4
2
0
−2
−4
−6
−8
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

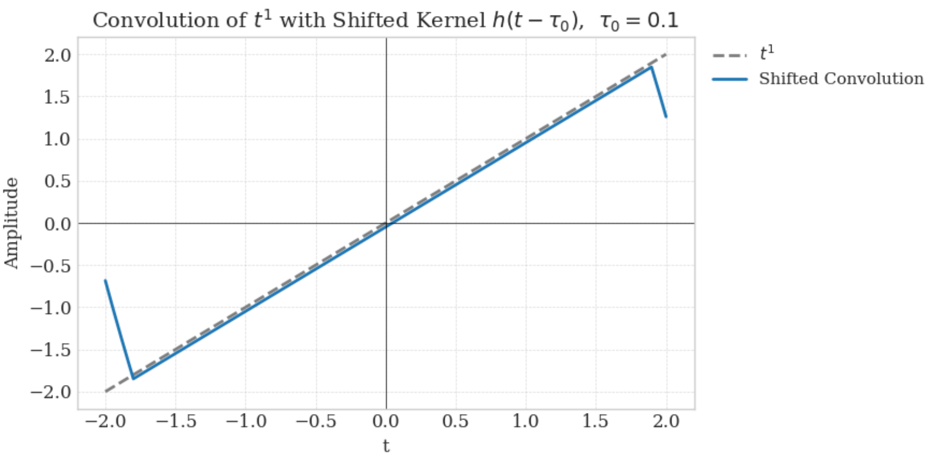

Causal Convolution of t^{-1} with Rectangular Kernel $h(t \geq 0)$
t^{-1} (Original)
Causal Convolution
Amplitude
t
400
200
0
−200
−400
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

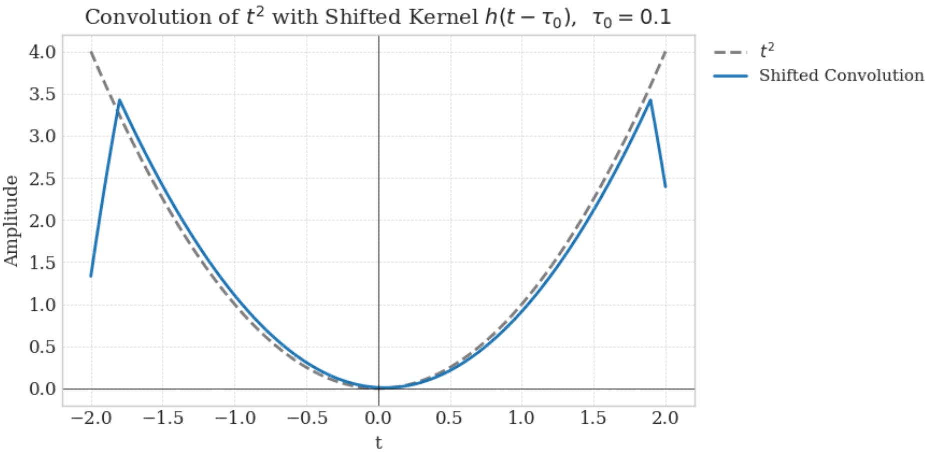


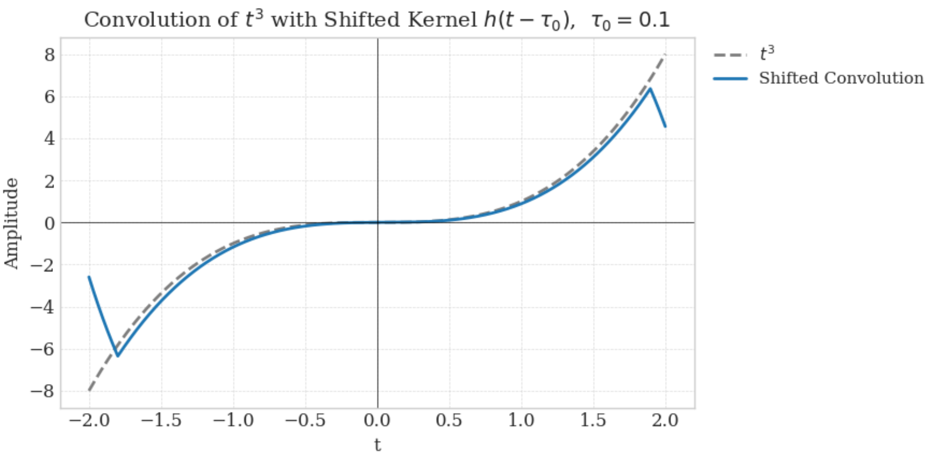
7.1.4 *Shifted Kernel*

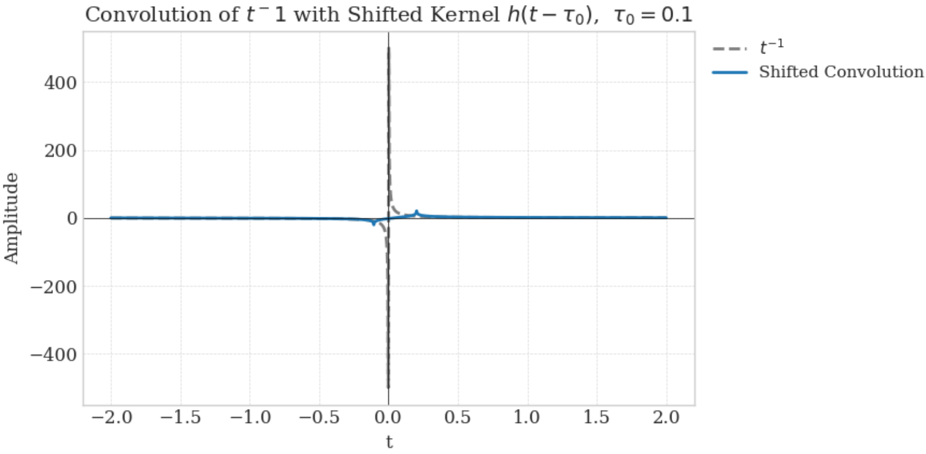
Convolving with a shifted kernel $h(t-\tau_0)$ introduces a time delay τ_0 in the convolution output. This model is useful for systems with a fixed delay or transmission delay, where the output is just a delayed version of the input.

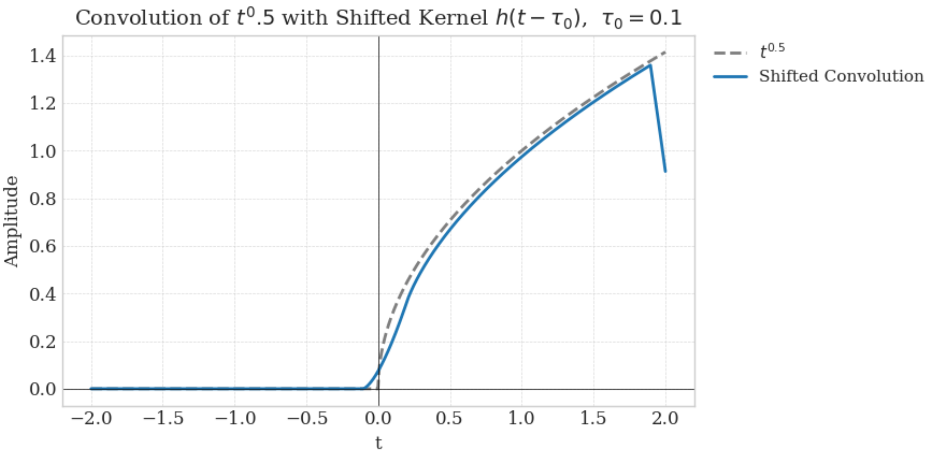
- The shape of the convolution result remains essentially unchanged compared to the original function, but the entire output is shifted by τ_0.
- This time-delay model preserves the smoothness and symmetry of the original algebraic functions, unlike the one-sided kernel that introduces asymmetry.
- For algebraic functions such as t, t^2, and t^3, the convolution simply shifts the function to the right by τ_0, keeping the characteristics intact.
- Such modeling is useful in time-delayed systems in control and signal transmission.




Convolution of t^2 with Shifted Kernel $h(t-\tau_0)$, $\tau_0 = 0.1$
Amplitude
t
t^2
Shifted Convolution
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Convolution of t^3 with Shifted Kernel $h(t-\tau_0)$, $\tau_0 = 0.1$
Amplitude
t
t^3
Shifted Convolution
8
6
4
2
0
−2
−4
−6
−8
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Convolution of t^-1 with Shifted Kernel $h(t-\tau_0)$, $\tau_0 = 0.1$
Amplitude
400
200
0
−200
−400
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
t
t^{-1}
Shifted Convolution


Convolution of $t^0.5$ with Shifted Kernel $h(t-\tau_0)$, $\tau_0 = 0.1$
Amplitude
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
t
$t^{0.5}$
Shifted Convolution

7.2 *Exponential Functions*

7.2.1 *Definition of f(t)*

$$f(t) = \exp(t).$$

We sample t uniformly in $[-2, 2]$ (1000 points) and compute $f(t)$ pointwise.

7.2.2 *Original Kernel*

By normalizing, we compute

$$y_{\text{sym}}(t) = (f * h_{\text{sym}})(t).$$

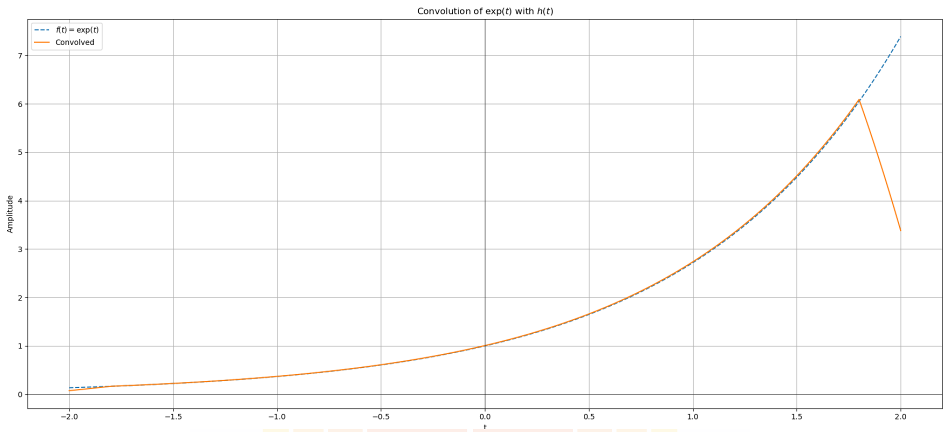


Fig. 3.10: Original $f(t)$ (dashed) and symmetric-kernel convolution $y_{\text{sym}}(t)$.

7.2.3 *Causal Kernel*

The causal (one-sided) kernel is

$$h_{\text{causal}}(t) = \begin{cases} 1, & 0 \le t \le T, \\ 0, & \text{else.} \end{cases}$$

After normalization,

$$y_{\text{causal}}(t) = (f * h_{\text{causal}})(t).$$

7.2.4 *Shifted Kernel*

We shift the symmetric kernel by $\tau_0 = 0.15$:

$$h_{\text{shift}}(t) = \begin{cases} 1, & -T + \tau_0 \le t \le T + \tau_0, \\ 0, & \text{else.} \end{cases}$$

Normalized, this yields

$$y_{\text{shift}}(t) = (f * h_{\text{shift}})(t).$$

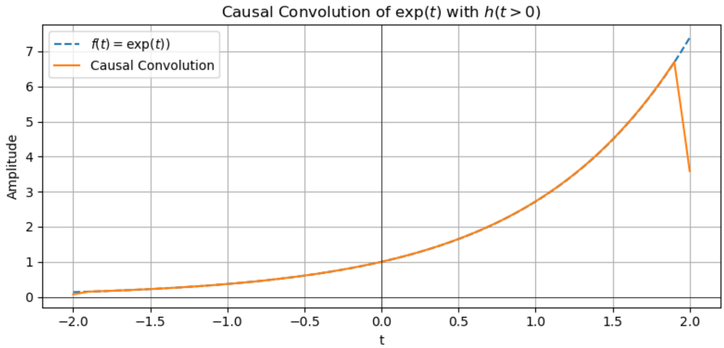


Fig. 3.11: Original $f(t)$ (dashed) and causal-kernel convolution $y_{\text{causal}}(t)$.

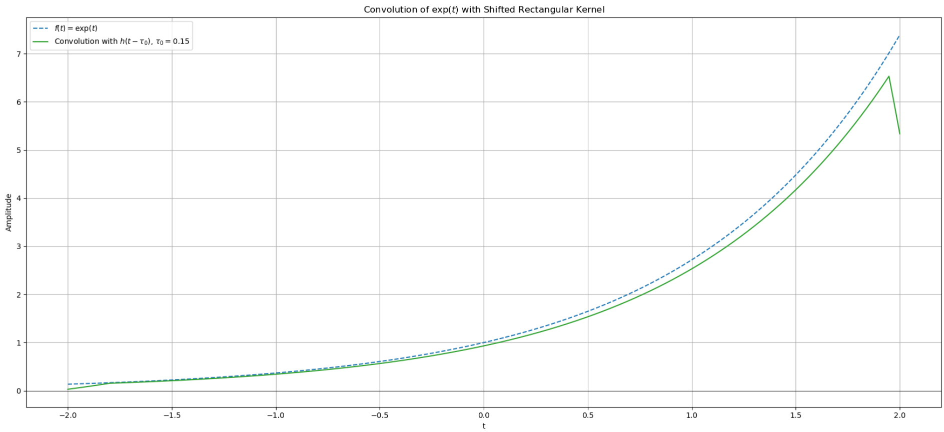


Fig. 3.12: Original $f(t)$ (dashed) and shifted-kernel convolution $y_{\text{shift}}(t)$.

7.3 Logarithmic Functions

7.3.1 Definition of f(t)

$$f(t) = \ln(1 + |t|).$$

We sample t uniformly in $[-2, 2]$ (1000 points) and compute $f(t)$ pointwise.

7.3.2 Original Kernel

The symmetric rectangular kernel is

$$h_{\text{sym}}(t) = \begin{cases} 1, & -T \le t \le T, \\ 0, & \text{else.} \end{cases}$$

After normalizing, we compute

$$y_{\text{sym}}(t) = (f * h_{\text{sym}})(t).$$

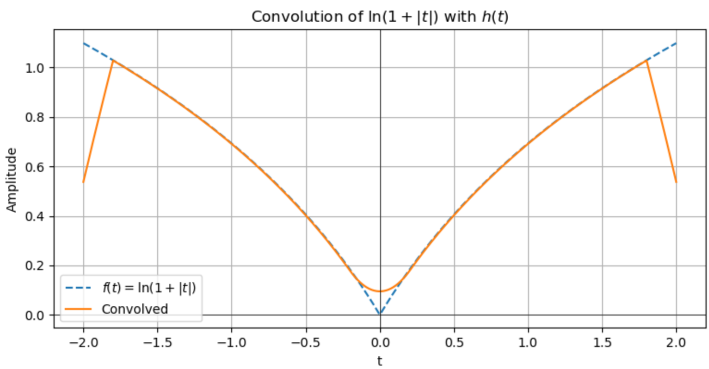


Fig. 3.13: Original $f(t)$ (dashed) and symmetric-kernel convolution $y_{\text{sym}}(t)$.

7.3.3 Causal Kernel

The causal (one-sided) kernel is

$$h_{\text{causal}}(t) = \begin{cases} 1, & 0 \le t \le T, \\ 0, & \text{else.} \end{cases}$$

After normalization,

$$y_{\text{causal}}(t) = (f * h_{\text{causal}})(t).$$

7.3.4 Shifted Kernel

We shift the symmetric kernel by $\tau_0 = 0.15$:

$$h_{\text{shift}}(t) = \begin{cases} 1, & -T + \tau_0 \le t \le T + \tau_0, \\ 0, & \text{else.} \end{cases}$$

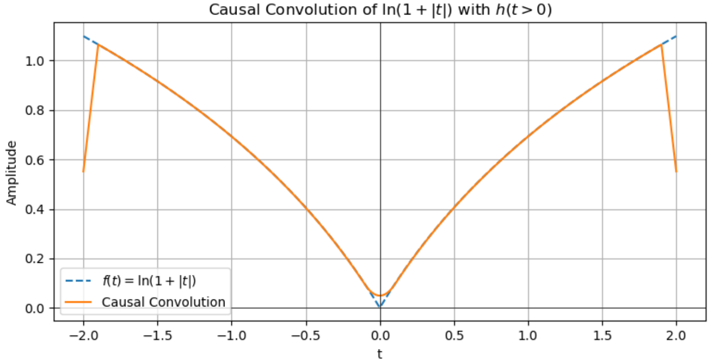


Fig. 3.14: Original $f(t)$ (dashed) and causal-kernel convolution $y_{\text{causal}}(t)$.

Normalized, this yields

$$y_{\text{shift}}(t) = (f * h_{\text{shift}})(t).$$

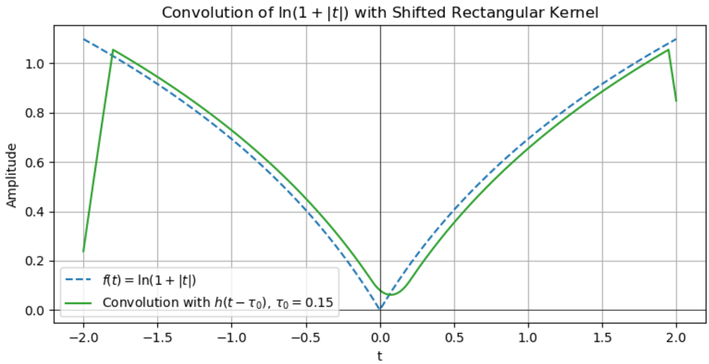


Fig. 3.15: Original $f(t)$ (dashed) and shifted-kernel convolution $y_{\text{shift}}(t)$.

7.4 Step & Delta Functions

7.4.1 Definition of f(t)

$$f(t) = \delta(t).$$

We sample t uniformly in $[-2, 2]$ (1000 points), and approximate $\delta(t)$ numerically as a narrow pulse centered at $t = 0$ with a small width ϵ, such that

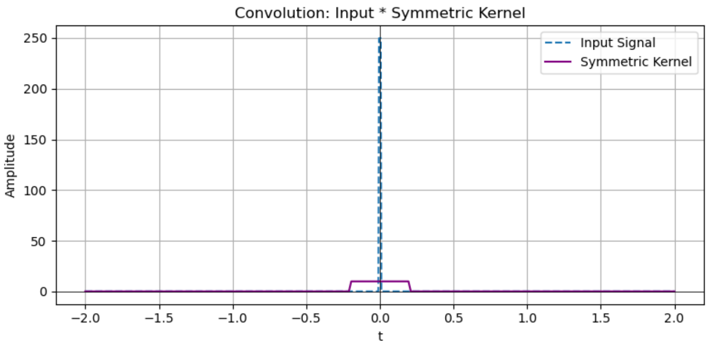
$$\delta_\epsilon(t) = \begin{cases} \frac{1}{\epsilon}, & |t| < \frac{\epsilon}{2}, \\ 0, & \text{otherwise}, \end{cases}$$

with $\epsilon = 0.01$. We then compute $f(t)$ pointwise using this approximation.

7.4.2 Original Kernel

After normalizing, we compute

$$y_{\text{sym}}(t) = (f * h_{\text{sym}})(t).$$



7.4.3 Causal Kernel

The causal (one-sided) kernel is

$$h_{\text{causal}}(t) = \begin{cases} 1, & 0 \le t \le T, \\ 0, & \text{else}. \end{cases}$$

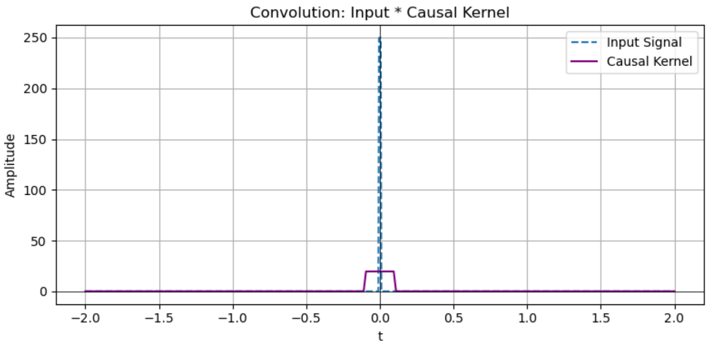
After normalization,

$$y_{\text{causal}}(t) = (f * h_{\text{causal}})(t).$$

7.4.4 Shifted Kernel

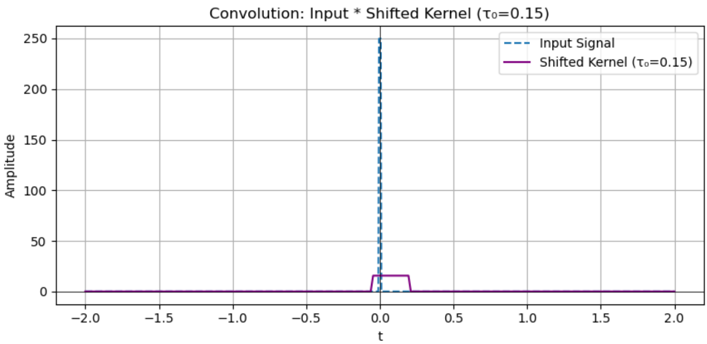
We shift the symmetric kernel by $\tau_0 = 0.15$:

$$h_{\text{shift}}(t) = \begin{cases} 1, & -T + \tau_0 \le t \le T + \tau_0, \\ 0, & \text{else}. \end{cases}$$



Normalized, this yields

$$y_{\text{shift}}(t) = (f * h_{\text{shift}})(t).$$



*7.4.5 **Definition of f(t)***

$$f(t) = u(t),$$

where

$$u(t) = \begin{cases} 0, & t < 0, \\ \\ 1, & t \geq 0. \end{cases}$$

We sample t uniformly in $[-2, 2]$ (1000 points) and compute $f(t)$ pointwise using the above definition.

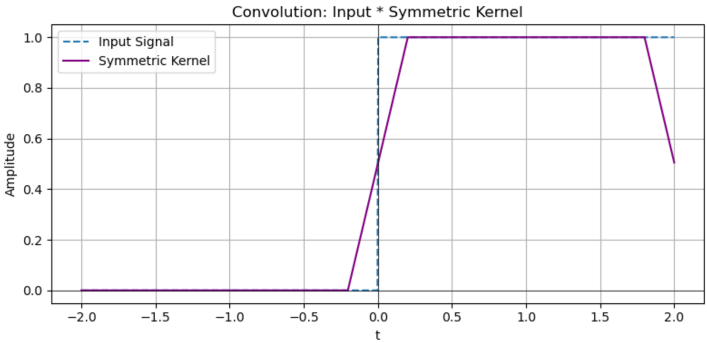
*7.4.6 **Original Kernel***

The symmetric rectangular kernel is

$$h_{\text{sym}}(t) = \begin{cases} 1, & -T \leq t \leq T, \\ 0, & \text{else.} \end{cases}$$

After normalizing, we compute

$$y_{\text{sym}}(t) = (f * h_{\text{sym}})(t).$$



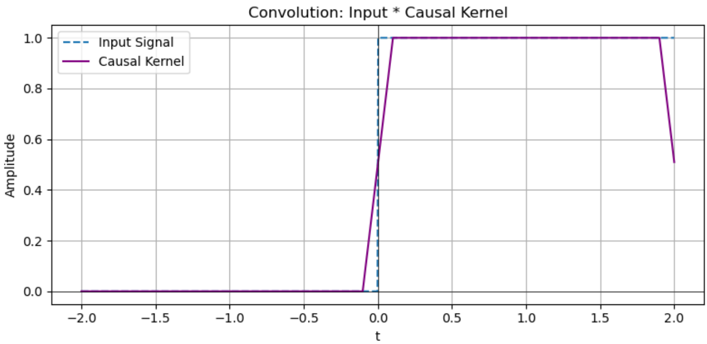
7.4.7 *Causal Kernel*

The causal (one-sided) kernel is

$$h_{\text{causal}}(t) = \begin{cases} 1, & 0 \le t \le T, \\ 0, & \text{else.} \end{cases}$$

After normalization,

$$y_{\text{causal}}(t) = (f * h_{\text{causal}})(t).$$



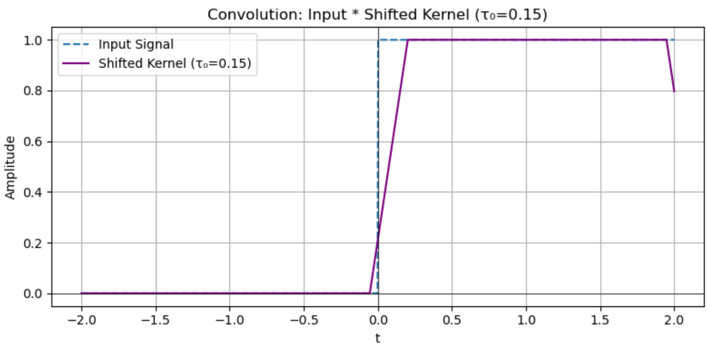
7.4.8 ***Shifted Kernel***

We shift the symmetric kernel by $\tau_0 = 0.15$:

$$h_{\text{shift}}(t) = \begin{cases} 1, & -T + \tau_0 \leq t \leq T + \tau_0, \\ 0, & \text{else.} \end{cases}$$

Normalized, this yields

$$y_{\text{shift}}(t) = (f * h_{\text{shift}})(t).$$



7.5 ***Trignometric Functions***

7.5.1 *Original Symmetric Kernel*

The **original symmetric kernel** $h(t)$ is defined as:

$$h(t) = \begin{cases} 1, & \text{for } -T \leq t \leq T \\ 0, & \text{otherwise} \end{cases}$$

When convolved with trigonometric functions:

- Acts as a **low-pass filter**, attenuating high frequencies

- For $\sin(t)$ and $\cos(t)$, scales amplitude by $\frac{\sin(T)}{T}$ (sinc function)
- Preserves **phase** due to kernel symmetry
- For $\tan(t)$, smooths sharp transitions while preserving overall trend
- Demonstrates **non-causal** behavior (uses future and past values)

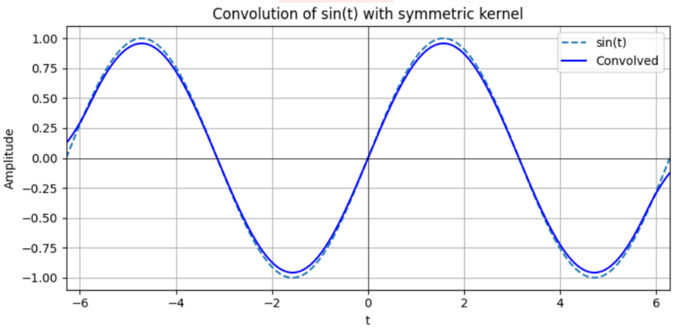


Fig. 3.16: Convolution of $\sin(t)$ with symmetric kernel

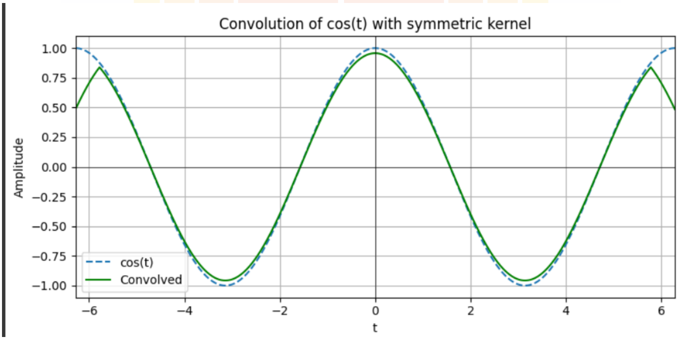


Fig. 3.17: Convolution of $\cos(t)$ with symmetric kernel

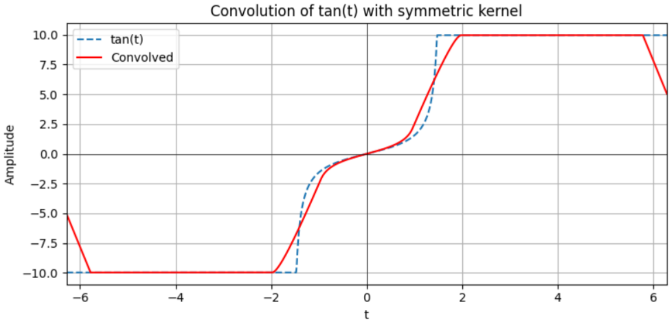


Fig. 3.18: Convolution of $\tan(t)$ with symmetric kernel

7.5.2 Modified One-Sided Kernel

The **modified causal kernel** ($t > 0$ only):

$$h(t) = \begin{cases} 1, & \text{for } 0 \leq t \leq T \\ 0, & \text{otherwise} \end{cases}$$

Key effects on trigonometric functions:

- Introduces **phase delay** of $T/2$ radians
- Reduces amplitude by additional factor of 2 compared to symmetric case
- Makes system **causal** (depends only on past/present)
- For $\sin(t)$, output becomes $2\sin(T/2)\sin(t - T/2)$
- Causes **asymmetric distortion** in $\tan(t)$ convolution

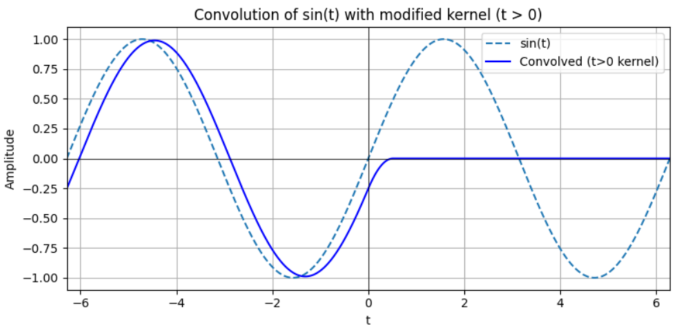


Fig. 3.19: Convolution of $\sin(t)$ with modified kernel

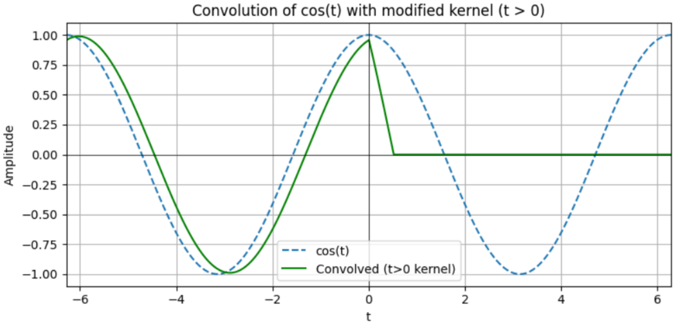


Fig. 3.20: Convolution of $\cos(t)$ with modified kernel

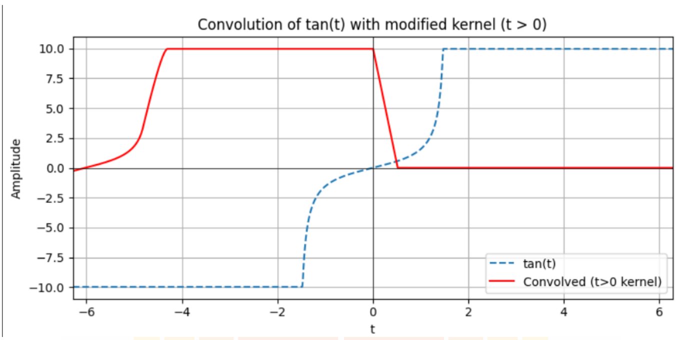


Fig. 3.21: Convolution of $\tan(t)$ with modified kernel

7.5.3 Shifted Kernel

The **time-shifted kernel** with delay τ:

$$h(t-\tau) = \begin{cases} 1, & \text{for } \tau \leq t \leq \tau + T \\ 0, & \text{otherwise} \end{cases}$$

Observations:

- Introduces **constant group delay** of $\tau + T/2$
- Preserves **waveform shape** unlike modified kernel
- For $\sin(t)$, results in $\frac{2\sin(T/2)}{T}\sin(t - \tau - T/2)$
- Maintains **relative phase relationships** between frequency components
- Models physical systems with **transmission delays**

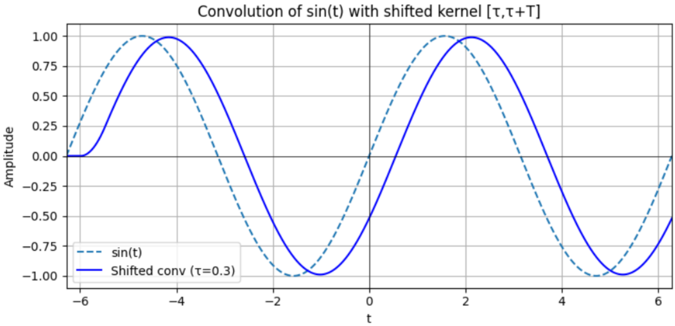


Fig. 3.22: Convolution of sin(t) with shifted kernel (τ = 0.3)

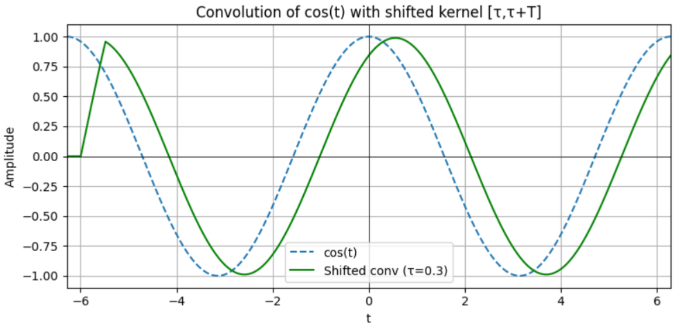


Fig. 3.23: Convolution of cos(t) with shifted kernel

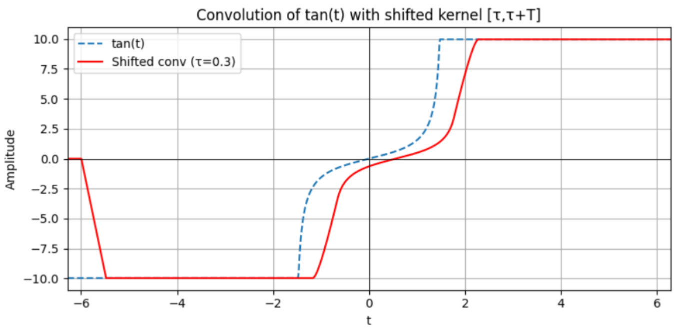


Fig. 3.24: Convolution of tan(t) with shifted kernel

7.6 *Inverse Trigonometric Functions*

7.6.1 *Definition of f(t)*

$$f_1(t) = \arccos(t), \quad f_2(t) = \arcsin(t), \quad f_3(t) = \arctan(t).$$

We sample t uniformly in $[-1, 1]$ for $\arccos(t)$ and $\arcsin(t)$ (since their domains are limited to $[-1, 1]$), and in $[-2, 2]$ for $\arctan(t)$, using 1000 points in each case. We compute each function pointwise over the sampled values of t.

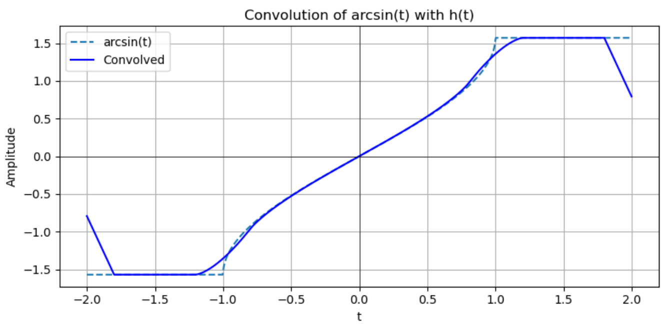
7.6.2 *Original Kernel*

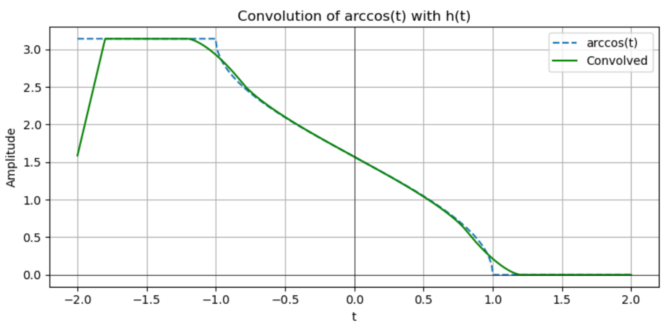
The **original symmetric kernel** $h(t)$ defined as:

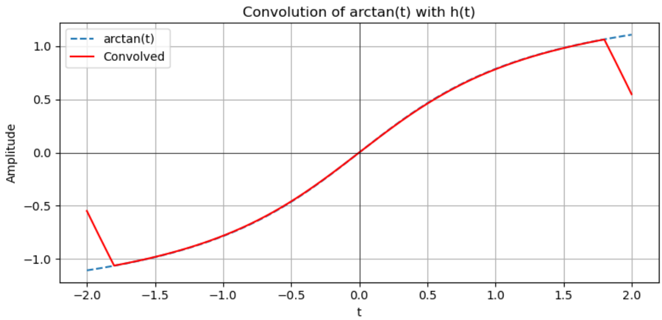
$$h(t) = \begin{cases} 1, & \text{for } -T \leq t \leq T \\ 0, & \text{otherwise} \end{cases}$$

The symmetric kernel includes both past and future values around time t. When this kernel is convolved with the inverse trigonometric functions:

- The result is a **smoothed** version of the original signal.
- The kernel acts like a moving average centered at each point.
- For $\arcsin(t)$ and $\arccos(t)$, the undefined regions outside $[-1, 1]$ are managed using `np.clip`, leading to saturation outside that interval.
- The convolution retains the central trend and symmetry of the original signal.







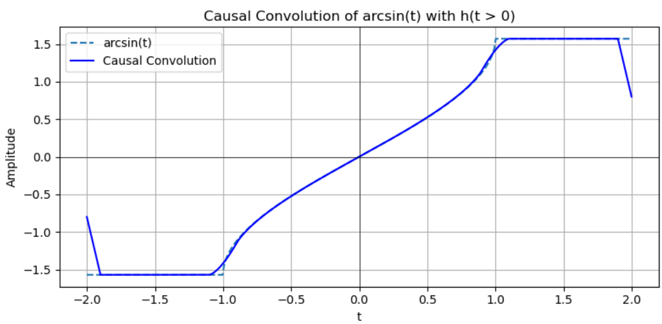
7.6.3 *Causal/Modified Kernel*

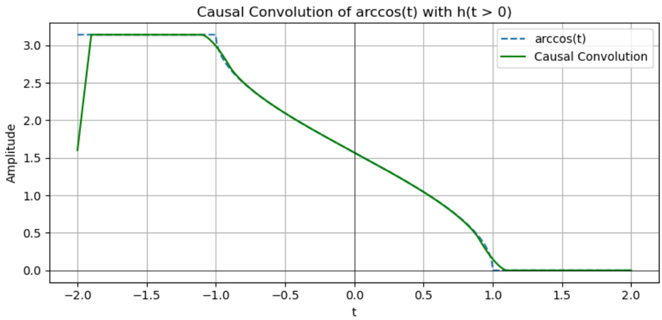
The **modified one-sided kernel** that only considers $t > 0$:

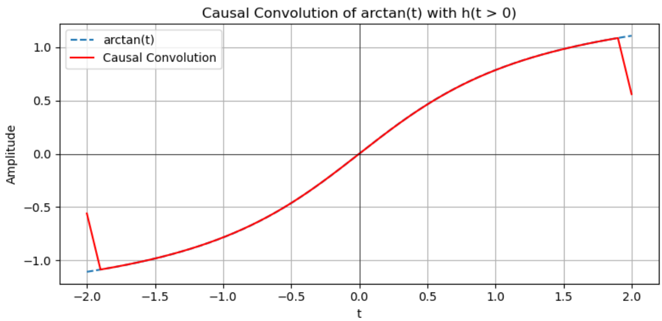
$$h(t) = \begin{cases} 1, & \text{for } 0 < t \leq T \\ 0, & \text{otherwise} \end{cases}$$

The one-sided kernel (nonzero only for $t > 0$) introduces an inherent **asymmetry** into the convolution:

- It models a **causal** system-one that responds only to the present and past.
- The output becomes more biased toward **past values**, creating a smoothing effect that **lags behind** the original function.
- For $\arcsin(t)$ and $\arccos(t)$, the convolution output shows a shift toward the left and becomes less symmetric.
- For $\arctan(t)$, the effect is more subtle but still causes a smoothing that trails the input.



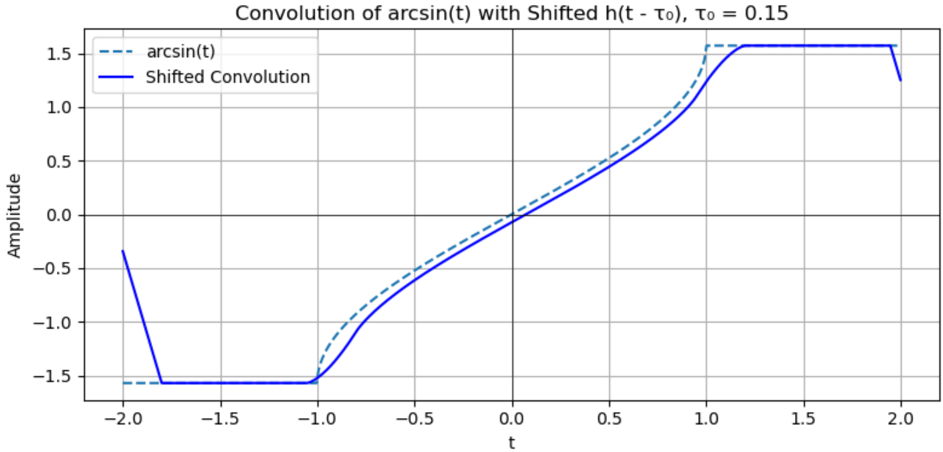




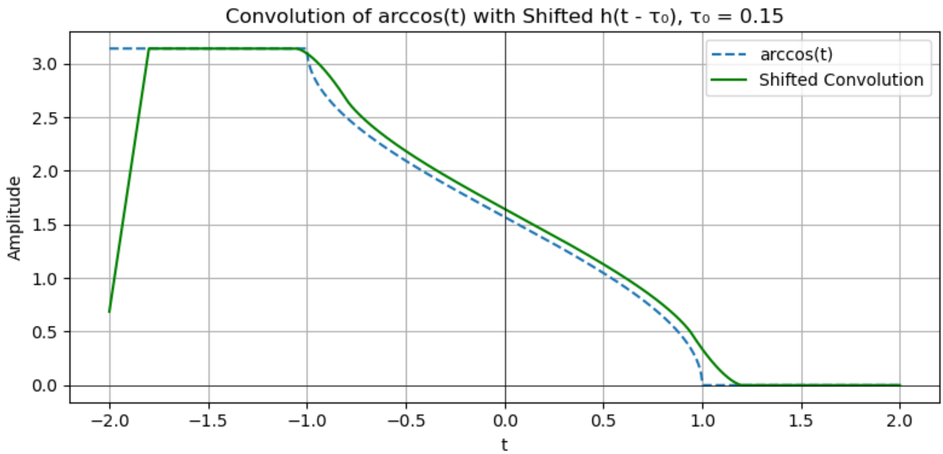
7.6.4 Shifted Kernel

Convolving with $h(t-\tau_0)$ introduces a time delay of τ_0 into the output:

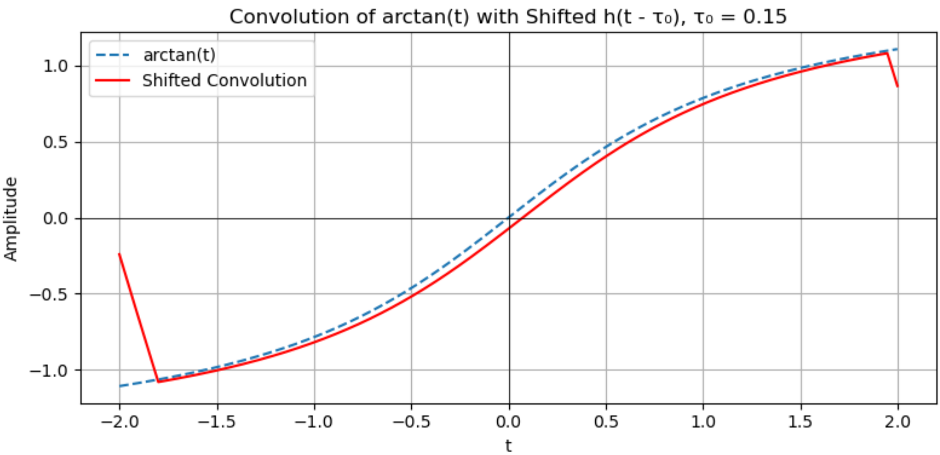
- The **shape** of the convolution result remains essentially unchanged.
- The entire output is **delayed** by τ_0, modeling systems with fixed time delay (e.g., transmission delays).
- This shift preserves symmetry and smoothness, unlike the one-sided kernel.
- Such modeling is useful in time-delayed systems in control and signal transmission.



Convolution of arcsin(t) with Shifted h(t - τ₀), τ₀ = 0.15
arcsin(t)
Shifted Convolution
Amplitude
t
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0



Convolution of arccos(t) with Shifted h(t - τ₀), τ₀ = 0.15
arccos(t)
Shifted Convolution
Amplitude
t
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0



Conclusion
Each type of kernel plays a different role:

- The **original kernel** performs symmetric smoothing.
- The **modified kernel** introduces causality and asymmetry.
- The **shifted kernel** adds a pure delay without distorting the function shape.

8 Applications of Convolution

8.1 Continuous Convolution (Signal Processing and Systems)

For two continuous functions $f(t)$ and $g(t)$, their convolution is defined as:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau$$

This is the mathematical definition of convolution for continuous functions. Here: - $f(t)$ is typically the input signal, - $g(t)$ is the system's impulse response, - The result, $(f*g)(t)$, is the output of the system at time t.

8.2 Discrete Convolution (Signal Processing and Systems)

For two discrete-time signals $f[n]$ and $g[n]$, their discrete convolution is defined as:

$$(f * g)[n] = \sum_{k=-\infty}^{\infty} f[k]g[n-k]$$

This is used in digital signal processing where sequences or discrete signals are convolved, and the system's response is calculated as a sum over all shifted versions of the input signal.

8.3 *Convolution of Probability Distributions (Probability Theory)*

For two independent random variables X and Y with probability density functions $f_X(x)$ and $f_Y(y)$, the probability density function $f_Z(z)$ of the sum $Z = X + Y$ is given by the convolution of $f_X(x)$ and $f_Y(y)$:

$$f_Z(z) = (f_X * f_Y)(z) = \int_{-\infty}^{\infty} f_X(x) f_Y(z - x)\, dx$$

This is a key result in probability theory, particularly in the study of the distribution of the sum of independent random variables.

8.4 *Convolution in Fourier Transforms*

Convolution is closely related to the Fourier transform. If $f(t)$ and $g(t)$ have Fourier transforms $\mathcal{F}[f(t)] = F(\omega)$ and $\mathcal{F}[g(t)] = G(\omega)$, then the Fourier transform of their convolution is the product of their individual Fourier transforms:

$$\mathcal{F}[f * g](\omega) = F(\omega) \cdot G(\omega)$$

This property is very useful in both signal processing and image processing because it simplifies convolution operations in the time domain by converting them to multiplication in the frequency domain.

8.5 *Convolution for Solving Differential Equations*

In the context of linear systems described by differential equations, the solution to a system with a given input $x(t)$ and impulse response $h(t)$ can be found using convolution:

$$y(t) = (x * h)(t) = \int_0^t x(\tau) h(t - \tau)\, d\tau$$

This is a common technique for solving ordinary differential equations (ODEs) with a given initial condition.

8.6 *Convolution in Discrete Linear Systems (Control Theory)*

For discrete systems, the output $y[n]$ of a system with input $x[n]$ and system response $h[n]$ is given by:

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

This is the discrete equivalent of the continuous convolution and is used to describe how a system reacts to discrete inputs.

8.7 *Convolution in Functional Analysis*

In functional analysis, convolution is used to define operations between functions that are elements of various function spaces. For example, in the space of square-integrable functions $L^2(\mathbb{R})$, convolution is an operation that combines functions f and g in a way that results in another function $h \in L^2(\mathbb{R})$.

9 GITHUB LINK:

Convolution-DETT-GroupQuiz-2