
# EE1060 - DETT GROUP QUIZ 2

EE24BTECH11053 - S A Aravind Eswar
EE24BTECH11026 - Gunda Srihaas
EE24BTECH11009 - A Mokshith Kumar Reddy
EE24BTECH11022 - Eshan Sharma
EE24BTECH11039 - Mandala Ranjith
EE24BTECH11012 - Bhavanisankar G S

May 5, 2025

# Contents

# 1 Question

Compute the convolution of a given signal $f(t)$ with a rectangular kernel $h(t)$, analytically. The rectangular kernel is defined as:

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

Derive the convolution expression $y(t) = (f * h)(t)$ in terms of known functions, and analyze the system's behavior for various values of the kernel duration $T$ and the input signal $f(t)$. Additionally, investigate the following scenarios:
(a) Modify the kernel to only consider the part of the kernel for $t > 0$. How does this affect the convolution result?
(b) Shift the kernel by a time $t_0$.
Analyze how the shift impacts the convolution output and discuss the significance of this shift in the context of time-delayed systems.
You are free to choose the form of the input signal f(t), but make sure it is well-defined and appropriate for convolution. A step or sinusoidal signal would work well for this analysis.

# 2 What is convolution ?

Convolution is a mathematical operation on two functions, $f$ and $g$, as the integral of the product of the two functions after one is reflected about the y-axis and shifted, i.e., If we are convolving a signal with a kernel, at each time $t$, we are sliding the kernel over the signal and computing how much they overlap. This is useful in various fields of study like finding the system response given the impulse response, signal processing and in probability.
Given two functions $f(x)$ and $g(x)$, convolution of the two signals,

$$x(t) = f(t) * g(t) \tag{2}$$

$$= \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau \tag{3}$$

Some properties of convolution :

1. **Commutativity** :

$$f(t) * g(t) = g(t) * f(t)$$

2. **Distributivity** :

$$f(t) * [g(t) + h(t)] = [f(t) * g(t)] + [f(t) * h(t)]$$

3. **Associativity** :

$$f(t) * [g(t) * h(t)] = [f(t) * g(t)] * h(t)$$

4. **Time shifting property** : If we are given $y(t) = x_1(t) * x_2(t)$, then

$$x_1(t) * x_2(t - T) = y(t - T)$$
$$x_1(t - T) * x_2(t) = y(t - T)$$
$$x_1(t - T_1) * x_2(t - T_2) = y(t - T_1 - T_2)$$

5. **Width property** : If the duration of signals $f(t)$ and $g(t)$ are $T_1$ and $T_2$ respectively, then the duration of signal convolving $f(t)$ and $g(t)$ is equal to $T_1 + T_2$

# 3 Significance of time-shift

1. In time-delayed systems, a positive shift ($t_0$) means that the system responds later to changes in the input. This can cause the system to react too slowly, which may lead to oscillations or instabilities in feedback systems, especially if the system relies on feedback loops (like in control systems or biological systems).

2. A negative shift ($t < 0$) might result in the system anticipating the input and responding before it occurs. This could cause overreaction or instability if the system is not designed to handle anticipatory behavior.

3. In devices like thermostat, the system may respond to temperature changes with a delay because of the time it takes to heat/cool the room. By shifting of kernel, if the delay becomes too large, then the system might not be able to track the input accurately, leading to instability (or) oscillations.

4. In the context of signal processing, shifting the kernel corresponds to a phase shift in the signal, e.g., in delay filters, the impulse response can be delayed.

5. In biological systems, time delays are common in processes like hormonal responses, neural signals, or metabolic pathways. Shifting the kernel models how these systems react over time. For example, the delayed release of insulin after glucose intake could be modeled using a time-shifted kernel.

# 4 Convolution of step function

For the kernel in (1), we shall find the convolution with $f(t) = u(t)$

## 4.1 Analytical Convolution

Since $u(\tau) = 0$ for $\tau < 0$, the integral becomes:

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

We determine the overlap of $h(t-\tau)$ within its support $[-T, T]$, resulting in:

$$t - T \leq \tau \leq t + T$$

Intersecting this with $\tau \geq 0$, the integration limits become:

$$\max(0, t-T) \leq \tau \leq t + T$$

Thus:

$$y(t) = \int_{\max(0,t-T)}^{t+T} 1\, d\tau = t + T - \max(0, t-T)$$

Breaking into cases:

$$y(t) = \begin{cases} 0, & t < -T \\ t+T, & -T \leq t < T \\ 2T, & t \geq T \end{cases}$$

## 4.2 Scenario Analysis

### 4.2.1 Causal Kernel

Modified kernel:

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

Now:

$$y(t) = \int_0^\infty h(t-\tau)\, d\tau = \int_{\max(0,t-T)}^{t} 1\, d\tau = \min(t, T)$$

Thus:

$$y(t) = \begin{cases} 0, & t < 0 \\ t, & 0 \leq t < T \\ T, & t \geq T \end{cases}$$

### 4.2.2 Shifted Kernel

Let $h(t) \to h(t - \tau_0)$. Then:

$$y(t) = \int_0^\infty u(\tau) h(t - \tau - \tau_0)\, d\tau = y_{\text{original}}(t - \tau_0)$$

The output is simply delayed by $\tau_0$.
The corresponding graphs are shown below -

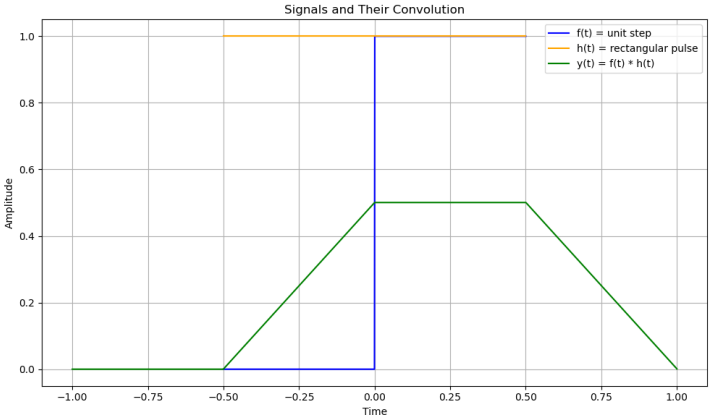



Figure 1: T = 0.5

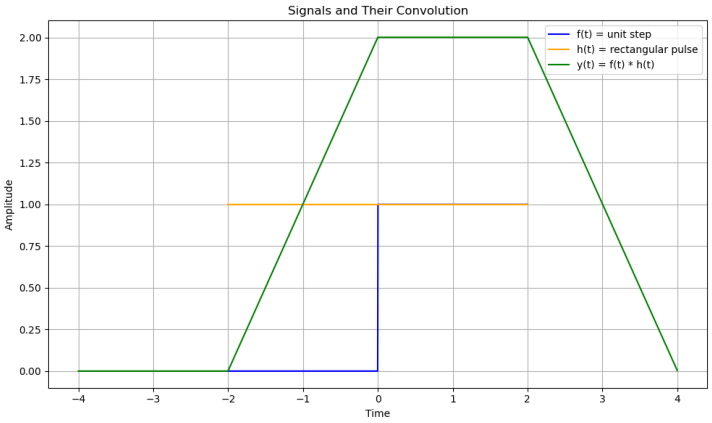



Figure 2: T = 2

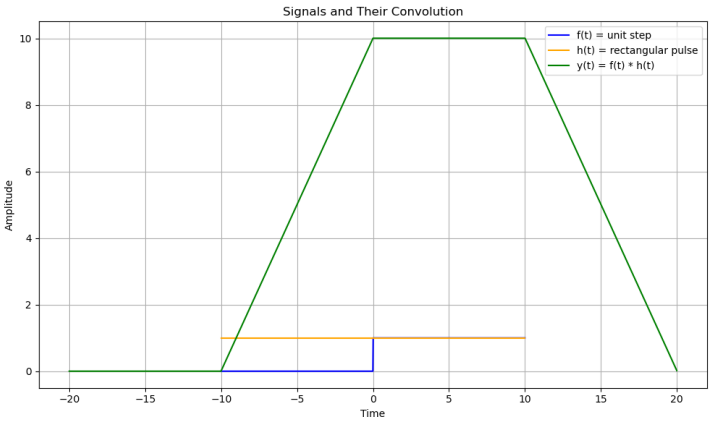



Figure 3: T = 10

# 5 Convolution of sinusoidal function

The convolution is defined as:

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

Given,

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

Which implies in convolution,

$$\begin{aligned} -T \le t - \tau \le T \\ t - T \le \tau \le t + T \end{aligned}$$

$\implies h(t-\tau) = 1$ in $(t-T, t+T)$

Thus, the convolution integral becomes:

$$y(t) = \int_{t-T}^{t+T} f(\tau)\,d\tau$$

For $f(t) = \sin(\omega t)$, the convolution becomes:

$$y(t) = \int_{t-T}^{t+T} \sin(\omega\tau)\,d\tau$$

$$\begin{aligned} y(t) &= \int_{t-T}^{t+T} \sin(\omega\tau)\,d\tau \\ &= \left[-\frac{1}{\omega}\cos(\omega\tau)\right]_{t-T}^{t+T} \\ &= -\frac{1}{\omega}\left[\cos(\omega(t+T)) - \cos(\omega(t-T))\right] \end{aligned}$$

Using the trigonometric identity:

$$\cos A - \cos B = -2\sin\left(\frac{A+B}{2}\right)\sin\left(\frac{A-B}{2}\right)$$

Let:

$$\begin{aligned} A &= \omega(t+T) \\ B &= \omega(t-T) \end{aligned}$$

Then:

$$\begin{aligned} \frac{A+B}{2} &= \omega t \\ \frac{A-B}{2} &= \omega T \end{aligned}$$

Substituting:

$$\begin{aligned} y(t) &= -\frac{1}{\omega}\left[-2\sin(\omega t)\sin(\omega T)\right] \\ &= \frac{2}{\omega}\sin(\omega T)\sin(\omega t) \end{aligned}$$

$$y(t) = \frac{2}{\omega}\sin(\omega T)\sin(\omega t)$$

$$f(t) = A_0 + \sum_{n=1}^{\infty} A_n \cos\left(\frac{n\pi t}{L}\right) + \sum_{n=1}^{\infty} B_n \sin\left(\frac{n\pi t}{L}\right)$$

By substitution of $f(t)$ and and reducing the integral we get

$$y(t) = \int_{t-T/2}^{t+T/2} \left[ A_0 + \sum_{n=1}^{\infty} A_n \cos\left(\frac{n\pi\tau}{L}\right) + \sum_{n=1}^{\infty} B_n \sin\left(\frac{n\pi\tau}{L}\right) \right] d\tau$$

**DC Term**

$$\int_{t-T/2}^{t+T/2} A_0 \, d\tau = A_0 T$$

**Sine part**

$$B_n \int \sin\left(\frac{n\pi\tau}{L}\right) d\tau = -\frac{L}{n\pi} \cos\left(\frac{n\pi\tau}{L}\right)$$

Evaluating from $t - T/2$ to $t + T/2$:

$$B_n \cdot \frac{2L}{n\pi} \cos\left(\frac{n\pi t}{L}\right) \sin\left(\frac{n\pi T}{2L}\right)$$

**Cosine part**

$$\begin{aligned} y_{\cos}(t) &= \int_{t-T}^{t+T} A_n \cos\left(\frac{n\pi\tau}{L}\right) \, d\tau \\ &= A_n \left[ \frac{L}{n\pi} \sin\left(\frac{n\pi\tau}{L}\right) \right]_{t-T}^{t+T} \\ &= A_n \cdot \frac{L}{n\pi} \left[ \sin\left(\frac{n\pi(t+T)}{L}\right) - \sin\left(\frac{n\pi(t-T)}{L}\right) \right] \end{aligned}$$

Using the trigonometric identity:

$$\sin C - \sin D = 2 \cos\left(\frac{C+D}{2}\right) \sin\left(\frac{C-D}{2}\right)$$

Let:

$$\begin{aligned} C &= \frac{n\pi(t+T)}{L} = \frac{n\pi t}{L} + \frac{n\pi T}{L} \\ D &= \frac{n\pi(t-T)}{L} = \frac{n\pi t}{L} - \frac{n\pi T}{L} \end{aligned}$$

Then:

$$\begin{aligned} \frac{C+D}{2} &= \frac{n\pi t}{L} \\ \frac{C-D}{2} &= \frac{n\pi T}{L} \end{aligned}$$

Substituting:

$$\begin{aligned} y_{\cos}(t) &= A_n \cdot \frac{L}{n\pi} \left[ 2\cos\left(\frac{n\pi t}{L}\right) \sin\left(\frac{n\pi T}{L}\right) \right] \\ &= A_n \cdot \frac{2L}{n\pi} \sin\left(\frac{n\pi T}{L}\right) \cos\left(\frac{n\pi t}{L}\right) \end{aligned}$$

$$y(t) = 2A_0 T + \sum_{n=1}^{\infty} \frac{2L}{n\pi} \sin\left(\frac{n\pi T}{L}\right) \left[ A_n \sin\left(\frac{n\pi t}{L}\right) + B_n \cos\left(\frac{n\pi t}{L}\right) \right]$$

## 5.1 Convolution of a Signal with a Rectangular Kernel

Given a signal $f(t)$ and a rectangular kernel $h(t)$ defined as:

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

The convolution $y(t) = (f * h)(t)$ is:

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

## 5.2 General Convolution Expression

The kernel $h(t - \tau)$ is 1 when $-T \leq t - \tau \leq T$, i.e., $t - T \leq \tau \leq t + T$. Thus:

$$y(t) = \int_{t-T}^{t+T} f(\tau)\, d\tau$$

## 5.3 Analysis for Various $T$ and $f(t)$

- **Large $T$:** More smoothing of $f(t)$, attenuates high frequencies.
- **Small $T$:** Less smoothing, $y(t) \approx f(t)$.
- **Constant $f(t) = c$:** $y(t) = 2Tc$.
- **Linear $f(t) = at + b$:**

$$\begin{aligned} y(t) &= \int_{t-T}^{t+T} (a\tau + b)\, d\tau \\ &= 2aTt + 2Tb \end{aligned}$$

### 5.3.1 Modified Kernel ($t > 0$)

Define:

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

The convolution becomes:

$$y_{\text{modified}}(t) = \int_{t-T}^{t} f(\tau)\, d\tau$$

**Difference:**

- Original: Integrates symmetrically around $t$ ($[t-T, t+T]$).
- Modified: Only integrates past values ($[t-T, t]$), making it causal.

### 5.3.2 Shifted Kernel by $\tau_0$

Shifted kernel:

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

The convolution is:

$$\begin{aligned} y_{\text{shifted}}(t) &= \int_{t-\tau_0-T}^{t-\tau_0+T} f(\tau)\, d\tau \\ &= y(t-\tau_0) \end{aligned}$$

**Interpretation:** Shifting the kernel by $\tau_0$ shifts the output by $\tau_0$.

## Specific Cases for $f(t)$

**Case 1:** $f(t) = \sin(\omega t)$

$$\begin{aligned} y(t) &= \int_{t-T}^{t+T} \sin(\omega\tau)\, d\tau \\ &= \frac{2\sin(\omega T)}{\omega} \sin(\omega t) \end{aligned}$$

For shifted kernel:

$$y_{\text{shifted}}(t) = \frac{2\sin(\omega T)}{\omega} \sin(\omega(t-\tau_0))$$

**Case 2: Fourier Series Representation**

Let:

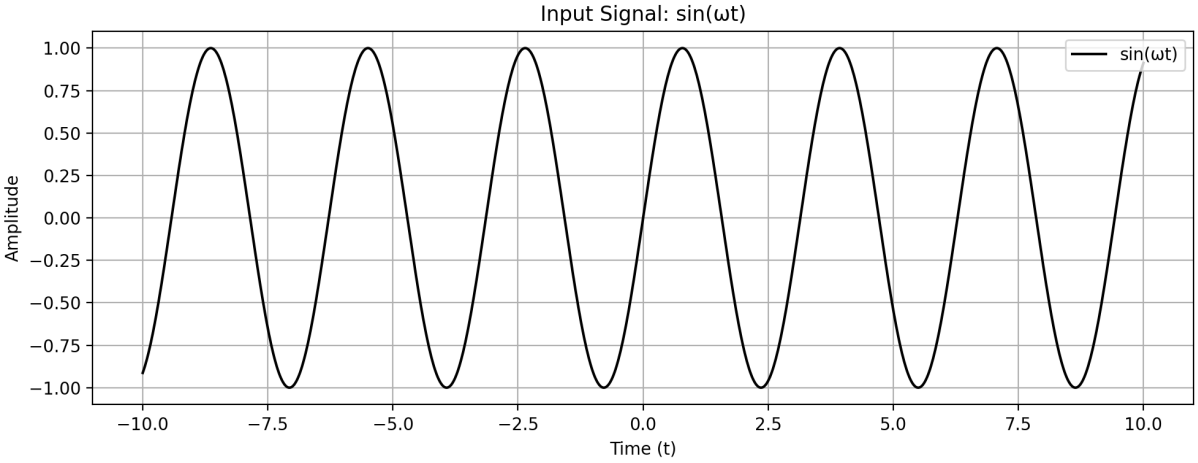
$$f(t) = A_0 + \sum_{n=1}^{\infty} A_n \cos\left(\frac{n\pi t}{L}\right) + \sum_{n=1}^{\infty} B_n \sin\left(\frac{n\pi t}{L}\right)$$

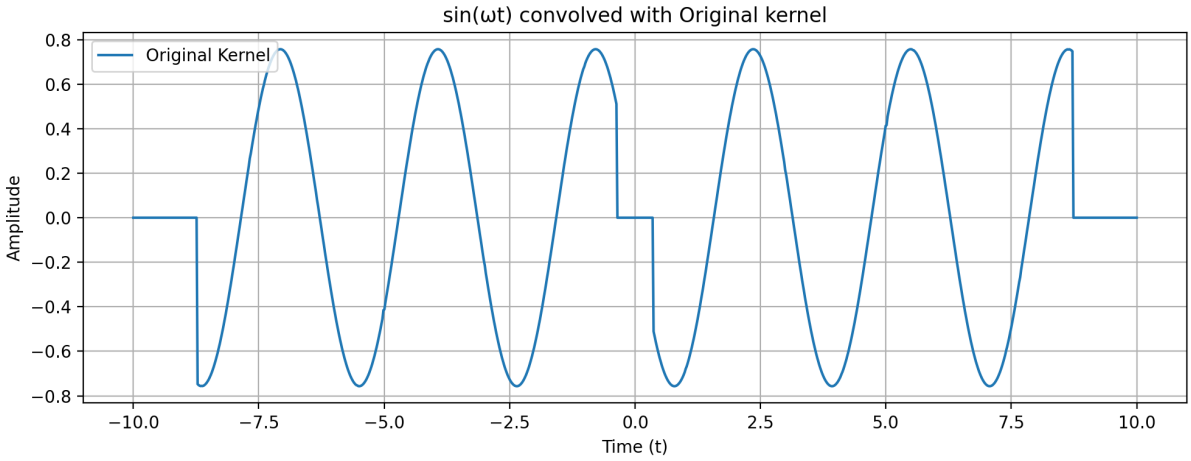
Then:

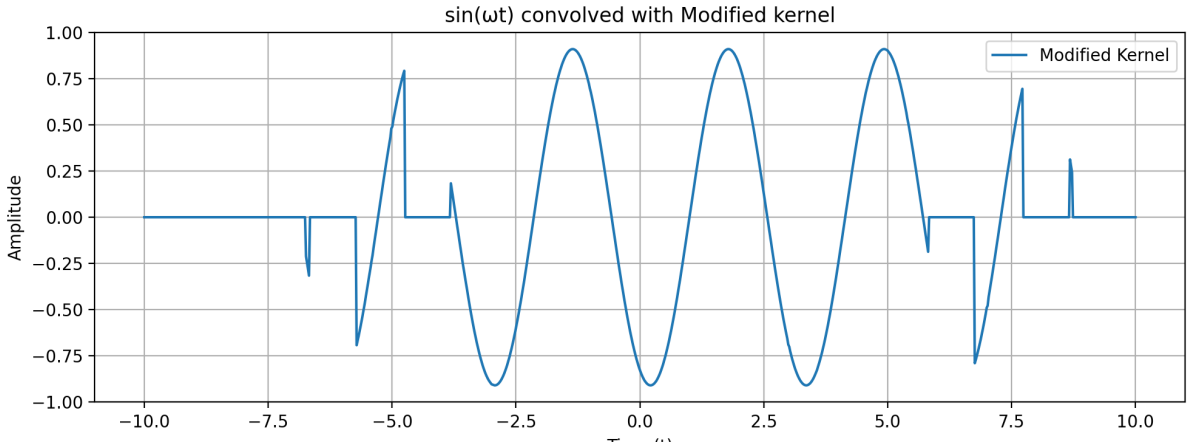
$$y(t) = 2TA_0 + \sum_{n=1}^{\infty} \frac{2L}{n\pi} \sin\left(\frac{n\pi T}{L}\right) \left[A_n \cos\left(\frac{n\pi t}{L}\right) + B_n \sin\left(\frac{n\pi t}{L}\right)\right]$$

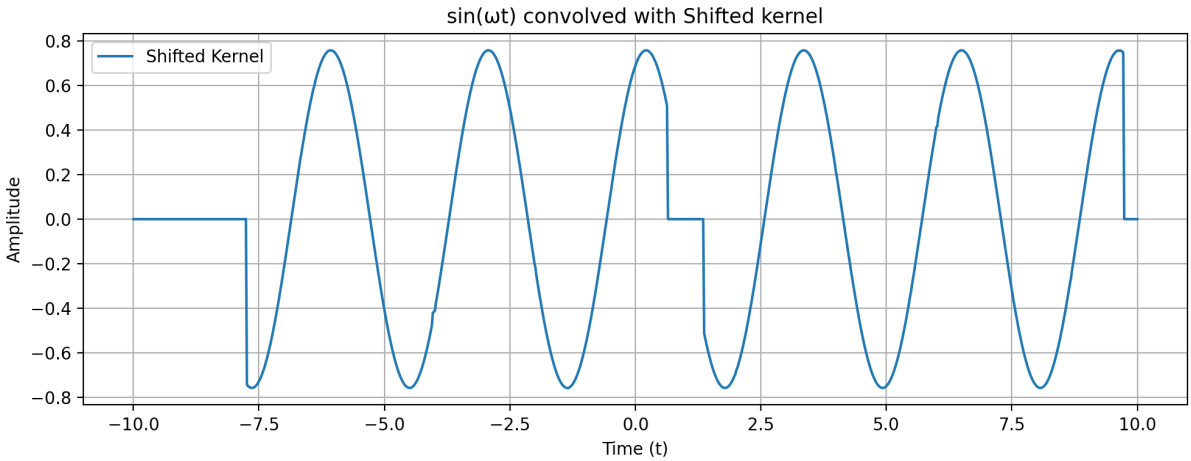
For shifted kernel:

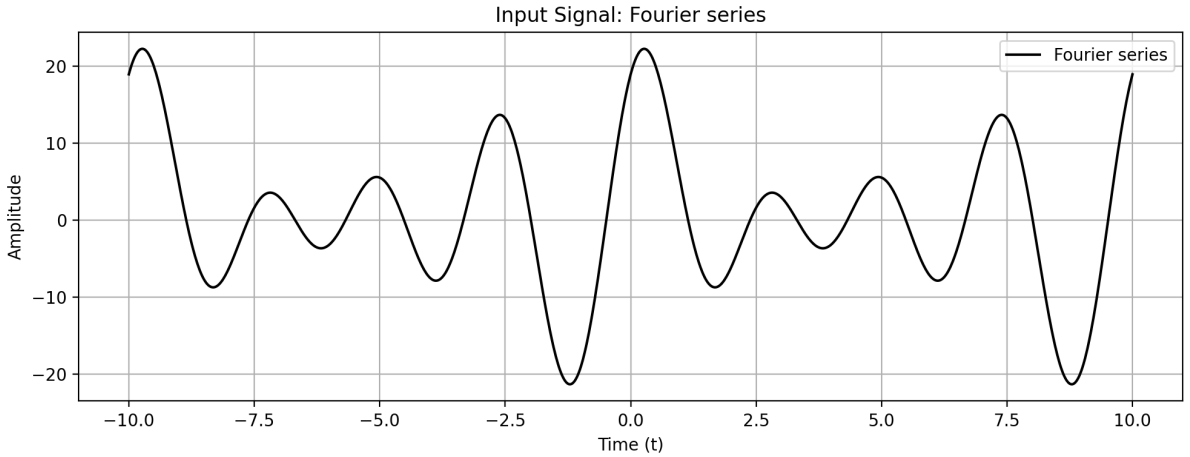
$$\begin{aligned} y_{\text{shifted}}(t) = 2TA_0 &+ \sum_{n=1}^{\infty} \frac{2L}{n\pi} \sin\left(\frac{n\pi T}{L}\right) \\ &\times \left[A_n \cos\left(\frac{n\pi(t-\tau_0)}{L}\right) + B_n \sin\left(\frac{n\pi(t-\tau_0)}{L}\right)\right] \end{aligned}$$

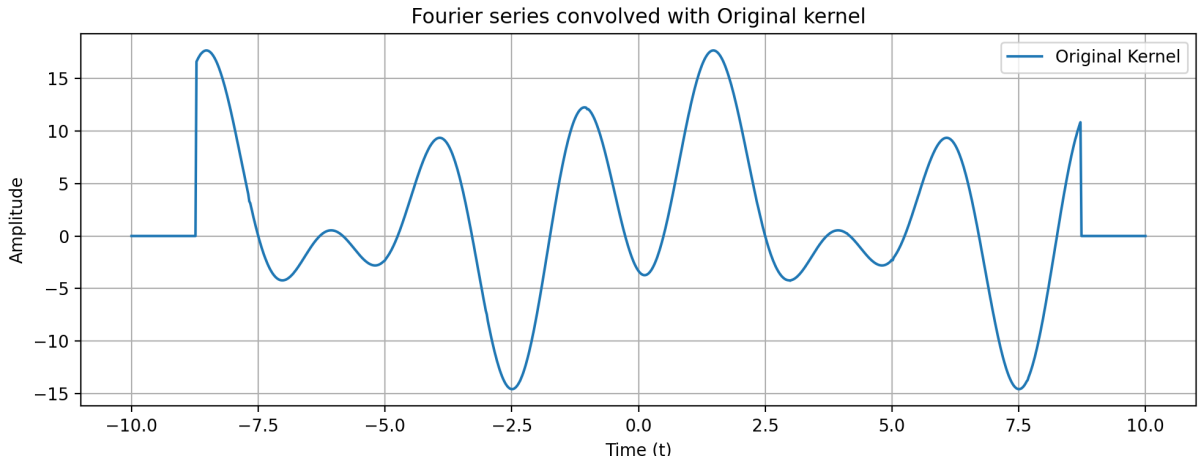

Input Signal: sin(ωt)
sin(ωt)
Amplitude
Time (t)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

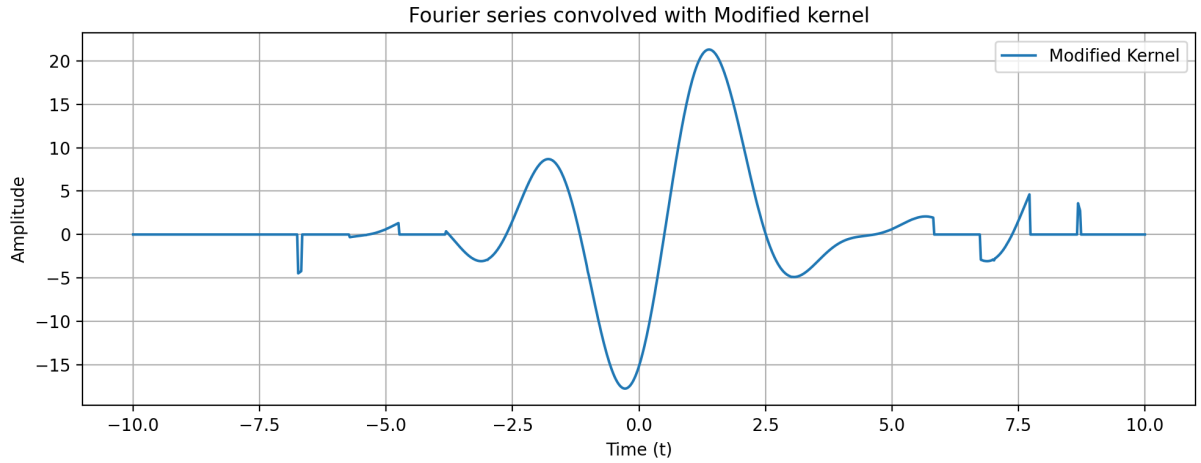

sin(ωt) convolved with Original kernel
Original Kernel
Amplitude
Time (t)
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

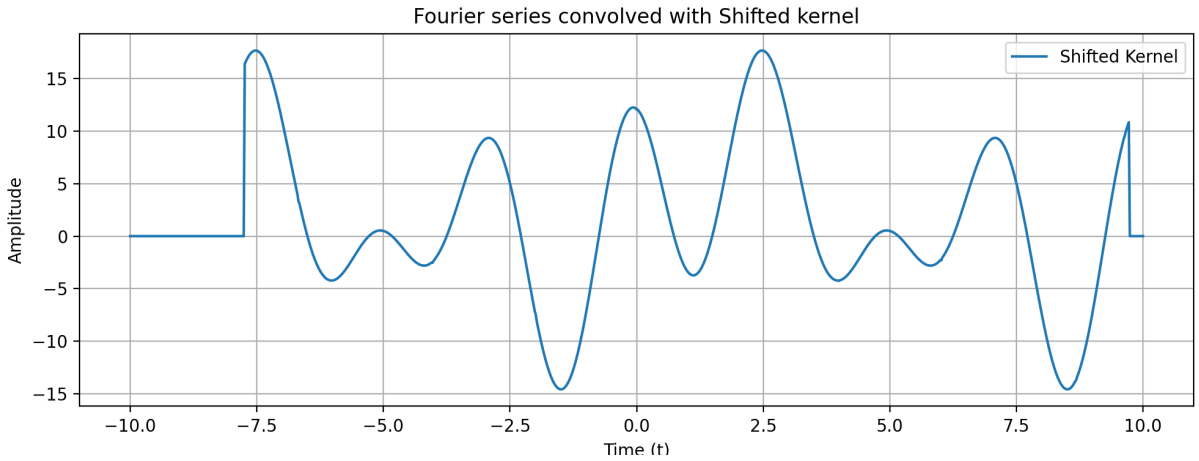

sin(ωt) convolved with Modified kernel
Modified Kernel
Amplitude
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


sin(ωt) convolved with Shifted kernel
Shifted Kernel
Amplitude
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Time (t)


Input Signal: Fourier series
Fourier series
Amplitude
20
10
0
−10
−20
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Time (t)


Fourier series convolved with Original kernel
Original Kernel
Amplitude
Time (t)


Fourier series convolved with Modified kernel
Modified Kernel
Amplitude
Time (t)


Fourier series convolved with Shifted kernel
Shifted Kernel
Amplitude
Time (t)

# 6 Convolution of linear function

Given kernel,

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

and the function

$$f(t) = at + b$$

Let $y(t) = x(t) * f(t)$, then

$$\begin{aligned} y(t) &= h(t) * f(t) \\ &= f(t) * h(t) \\ &= \int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau \end{aligned}$$

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

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

Therefore, the convolution becomes,

$$\begin{aligned} y(t) &= \int_{t-T}^{t+T} (at+b) * 1 d\tau \\ y(t) &= \int_{t-T}^{t+T} (at+b) d\tau \\ y(t) &= \frac{a}{2}[(t+T)^2 - (t-T)^2] + b[t+T-(t-T)] \\ y(t) &= 2atT + 2bT \end{aligned}$$

This integration can be numerically computed and can be seen to be the following - (we are using the values of and b as 1)

For different values of T, the change in $y(t)$ is depicted in the following -
It can be seen that as $T$ increases, the slope of the output is increasing but it is still

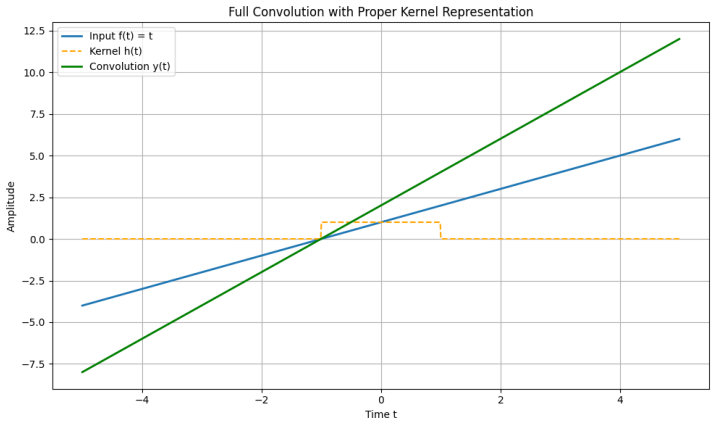



Figure 4: T = 1

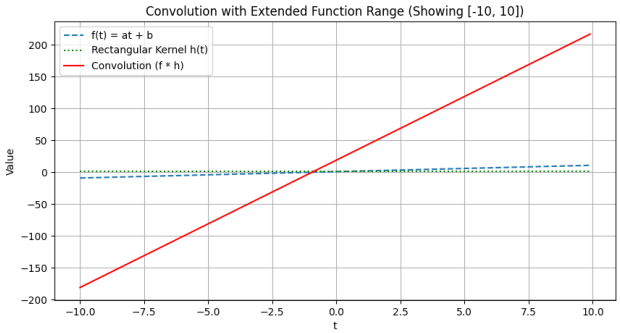



Figure 5: T = 1

linear.

## 6.1 Considering the kernel for $t > 0$

The response of the system for different values of $T$ are as follows - Modified kernel is,

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

and the function

$$f(t) = at + b$$

Let $y(t) = x(t) * f(t)$, then

$$\begin{aligned} y(t) &= h(t) * f(t) \\ &= f(t) * h(t) \\ &= \int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau \end{aligned}$$

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

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

Therefore, the convolution becomes,

$$\begin{aligned} y(t) &= \int_{t-T}^{t+T} (at+b) * 1 d\tau \\ y(t) &= \int_{t-T}^{t+T} (at+b) d\tau \\ y(t) &= \frac{a}{2}[(t)^2 - (t-T)^2] + b[t-(t-T)] \\ y(t) &= atT - \frac{a}{2}T^2 + bT \end{aligned}$$

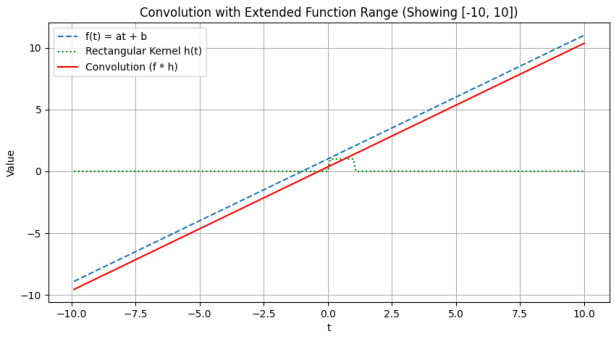



Figure 6: T = 1,A=1,B=1

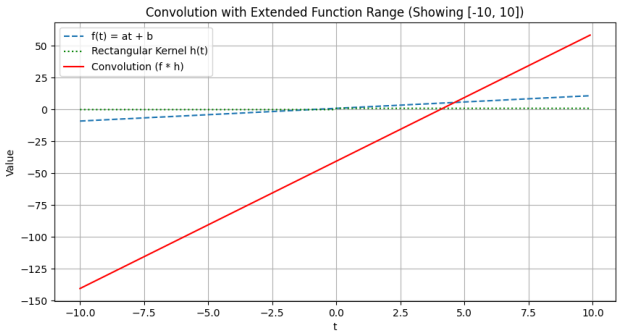



Figure 7: T = 10,A=1,B=1

## 6.2 Shifting the kernel by $t_0$

Shifting of kernel means moving the kernel to the left or right by an amount, i.e., if the given kernel is shifted by an amount $t_0$, then it becomes,

$$h(t) = \begin{cases} 1, & \text{for } -T \leq t - t_0 \leq T \\ 0, & \text{otherwise} \end{cases}$$

$$h(t) = \begin{cases} 1, & \text{for } -T + t_0 \leq t \leq T + t_0 \\ 0, & \text{otherwise} \end{cases}$$

$$f(t) = at + b$$

Let $y(t) = x(t) * f(t)$, then

$$\begin{aligned} y(t) &= h(t) * f(t) \\ &= f(t) * h(t) \\ &= \int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau \end{aligned}$$

$$h(t-\tau) = \begin{cases} 1, & \text{for } -T + t_0 \leq t - \tau \leq T + t_0 \\ 0, & \text{otherwise} \end{cases}$$

$$\begin{aligned} y(t) &= \int_{t-T}^{t+T} (at+b) * 1 d\tau \\ y(t) &= \int_{t-T}^{t+T} (at+b) d\tau \\ y(t) &= \frac{a}{2}[(t)^2 - (t-T)^2] + b[t - (t-T)] \\ y(t) &= 2a(t-t_0)T + 2bT \end{aligned}$$

Here we can observe that the solution is shifted by $t_0$ as the kernel is shifted by $t_0 = 10$ for the case where T=A=B=1, Shifting of the kernel by 10 units results in shifting of convoluted function by 10 units.

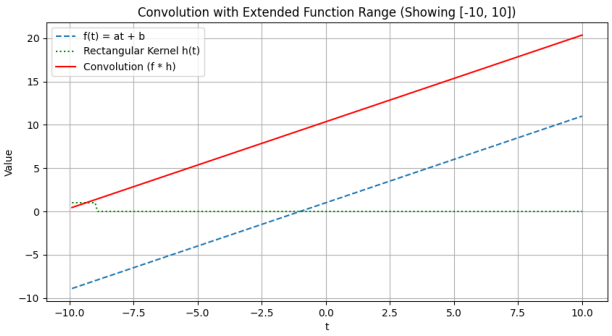



Figure 8: $t_0 = 10$

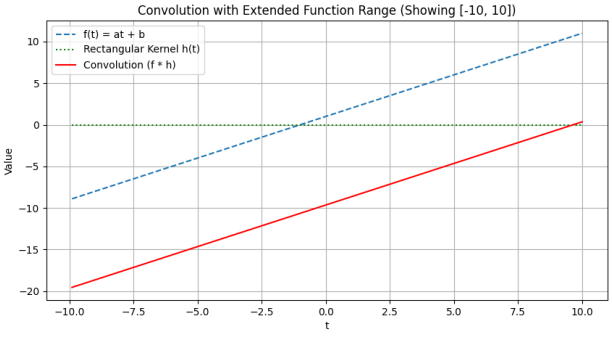



Figure 9: $t_0 = -10$

# 7 Convolution of Exponential Functions

Let's choose $f(t) = e^{-at}$ for $a > 0$ as our input signal.

The convolution is:

$$y(t) = \int_{-\infty}^{\infty} e^{-a\tau} \cdot h(t-\tau) d\tau \tag{4}$$

Since $h(t-\tau)$ is non-zero only when $-T \leq t-\tau \leq T$, or $t-T \leq \tau \leq t+T$, the effective integration limits are:

$$y(t) = \int_{t-T}^{t+T} e^{-a\tau} d\tau \tag{5}$$

Computing this integral:

$$y(t) = \int_{t-T}^{t+T} e^{-a\tau} d\tau \tag{6}$$

$$= \left[ -\frac{1}{a} e^{-a\tau} \right]_{t-T}^{t+T} \tag{7}$$

$$= \frac{e^{-at}}{a} (e^{aT} - e^{-aT}) \tag{8}$$

Using the definition of hyperbolic sine, we can write:

$$y(t) = \frac{2\sinh(aT)}{a} e^{-at} \tag{9}$$

# 8 Convolution with Hyperbolic Function

Let's consider $f(t) = \sinh(bt)$ for $b > 0$.

The convolution is:

$$y(t) = \int_{-\infty}^{\infty} \sinh(b\tau) \cdot h(t-\tau) d\tau = \int_{t-T}^{t+T} \sinh(b\tau) d\tau \tag{10}$$

Computing this integral:

$$y(t) = \int_{t-T}^{t+T} \sinh(b\tau) d\tau \tag{11}$$

$$= \left[ \frac{1}{b} \cosh(b\tau) \right]_{t-T}^{t+T} \tag{12}$$

$$= \frac{1}{b} (\cosh(b(t+T)) - \cosh(b(t-T))) \tag{13}$$

Using the identity $\cosh(A) - \cosh(B) = 2\sinh(\frac{A+B}{2})\sinh(\frac{A-B}{2})$:

$$y(t) = \frac{1}{b} (2\sinh(bt)\sinh(bT)) \tag{14}$$

$$= \frac{2\sinh(bT)}{b} \sinh(bt) \tag{15}$$

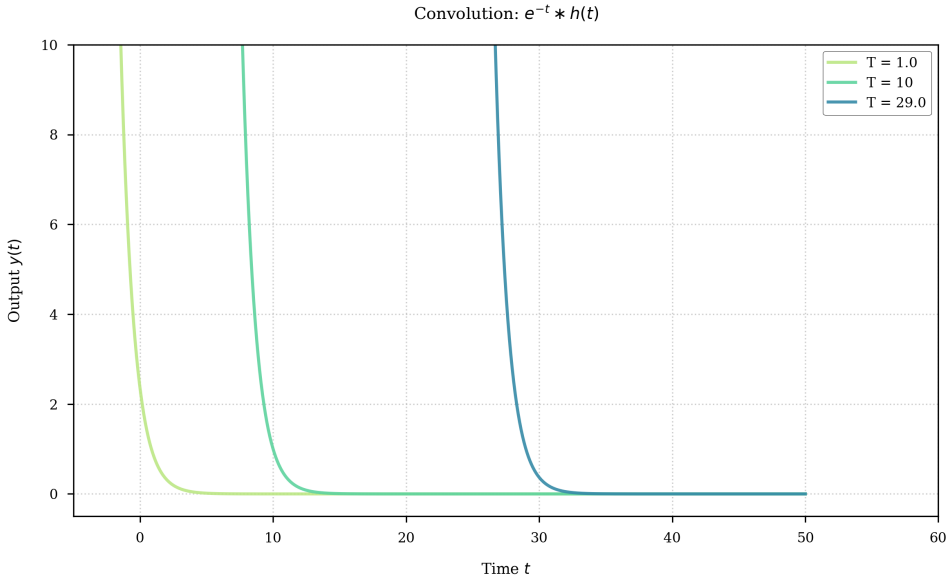



Figure 10: Convolution of $f(t) = e^{-t}$ with a rectangular kernel for varying values of $T$. As $T$ increases, the output signal amplitude scales by $\frac{2\sinh(aT)}{a}$ but retains the exponential decay nature. For large values of $T$ (e.g., $T = 29.0$), the output achieves relatively higher amplitudes (approximately 8) than smaller $T$ values, showing the amplification property of the width of the rectangular kernel without losing the exponential nature of the original function.

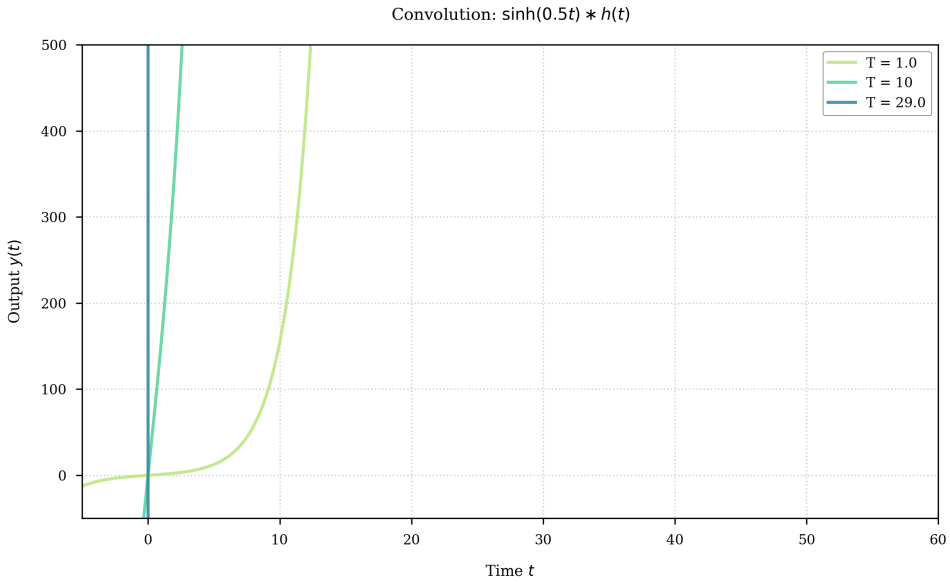



Figure 11: Convolution of $f(t) = \sinh(0.5t)$ with a rectangular kernel of varying $T$. The output retains the hyperbolic sine form but with increased magnitude that varies with $\frac{2\sinh(bT)}{b}$. With an increase in $T$, the amplitude increases significantly, reaching around $\pm 100$ for $T = 29.0$. In contrast to the exponential scenario, this output increases both positively and negatively, illustrating the way the wider rectangular kernels increase the hyperbolic nature of the original function.

## 8.1 Modified Kernel Analysis

### 8.1.1 Kernel for $t > 0$ (Part a)

For part (a), we need to modify the kernel to only consider the part for $t > 0$:

$$h_m(t) = \begin{cases} 1, & \text{for } 0 \leq t \leq T \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

This is equivalent to applying a unit step function to the original kernel and adjusting the range:

$$h_m(t) = h(t + T) \cdot u(t) \tag{17}$$

For the exponential function $f(t) = e^{-at}$, we now need to introduce the step function $u(t)$ for a causal system analysis. Let's define $f_c(t) = e^{-at}u(t)$ for $a > 0$.

The convolution becomes:

$$y_m(t) = \int_{-\infty}^{\infty} e^{-a\tau}u(\tau) \cdot h_m(t - \tau)d\tau \tag{18}$$

The effective integration limits are determined by where both $\tau \geq 0$ and $0 \leq t-\tau \leq T$, or $t - T \leq \tau \leq t$ and $\tau \geq 0$:

$$y_m(t) = \int_{\max(0,t-T)}^{\min(t,t)} e^{-a\tau}d\tau \tag{19}$$

Analyzing different cases:

### 8.1.2 Case 1: $t < 0$

In this case, $\min(t, t) = t < 0$, so there's no overlap and $y_m(t) = 0$.

### 8.1.3 Case 2: $0 \leq t < T$

Here, $t - T < 0$ and $t \geq 0$, so:

$$y_m(t) = \int_0^t e^{-a\tau}d\tau \tag{20}$$

$$= \frac{1}{a}(1 - e^{-at}) \tag{21}$$

### 8.1.4 Case 3: $t \geq T$

Both $t - T \geq 0$ and $t > 0$, so:

$$y_m(t) = \int_{t-T}^t e^{-a\tau}d\tau \tag{22}$$

$$= \frac{1}{a}(e^{-a(t-T)} - e^{-at}) \tag{23}$$

$$= \frac{e^{-at}}{a}(e^{aT} - 1) \tag{24}$$

Therefore, the complete convolution result for the modified kernel is:

$$y_m(t) = \begin{cases} 0, & t < 0 \\ \frac{1}{a}(1 - e^{-at}), & 0 \le t < T \\ \frac{e^{-at}}{a}(e^{aT} - 1), & t \ge T \end{cases} \tag{25}$$

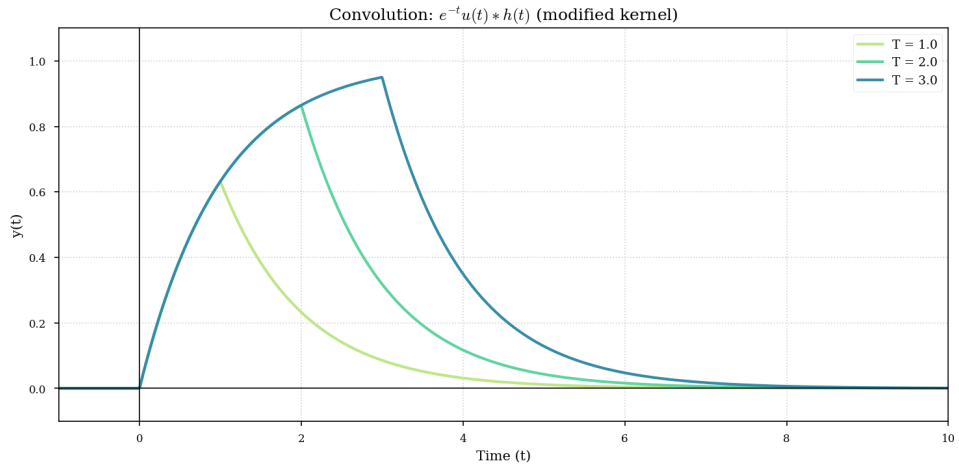



Figure 12: Convolution of $f(t) = e^{-t}u(t)$ with the modified kernel for various values of $T$. The result is zero for $t < 0$ due to causality. For $0 \le t < T$, the response rises as $(1 - e^{-at})/a$ to various maximum levels depending on $T$. For $t \ge T$, the output takes an exponential decay form $e^{-at}(e^{aT} - 1)/a$. Higher values of $T$ produce greater peak amplitudes and broader response regions. For $T = 3.0$, the peak is about 0.95 at $t \approx 3$, with exponential decay thereafter.

## 8.2 Hyperbolic Function with Modified Kernel

For the hyperbolic function with the modified kernel, a similar analysis gives:

### 8.2.1 Case 1: $t < 0$

The output is zero: $y_m(t) = 0$.

### 8.2.2 Case 2: $0 \le t < T$

$$y_m(t) = \int_0^t \sinh(b\tau)d\tau \tag{26}$$

$$= \frac{1}{b}(\cosh(bt) - 1) \tag{27}$$

### 8.2.3 Case 3: $t \ge T$

$$y_m(t) = \int_{t-T}^t \sinh(b\tau)d\tau \tag{28}$$

$$= \frac{1}{b}(\cosh(bt) - \cosh(b(t - T))) \tag{29}$$

Using the identity $\cosh(A) - \cosh(B) = 2\sinh(\frac{A+B}{2})\sinh(\frac{A-B}{2})$:

$$y_m(t) = \frac{1}{b}(2\sinh(b(t - \frac{T}{2}))\sinh(\frac{bT}{2})) \tag{30}$$

$$= \frac{2\sinh(\frac{bT}{2})}{b}\sinh(b(t - \frac{T}{2})) \tag{31}$$

Therefore, the complete convolution result is:

$$y_m(t) = \begin{cases} 0, & t < 0 \\ \frac{1}{b}(\cosh(bt) - 1), & 0 \le t < T \\ \frac{2\sinh(\frac{bT}{2})}{b}\sinh(b(t - \frac{T}{2})), & t \ge T \end{cases} \tag{32}$$

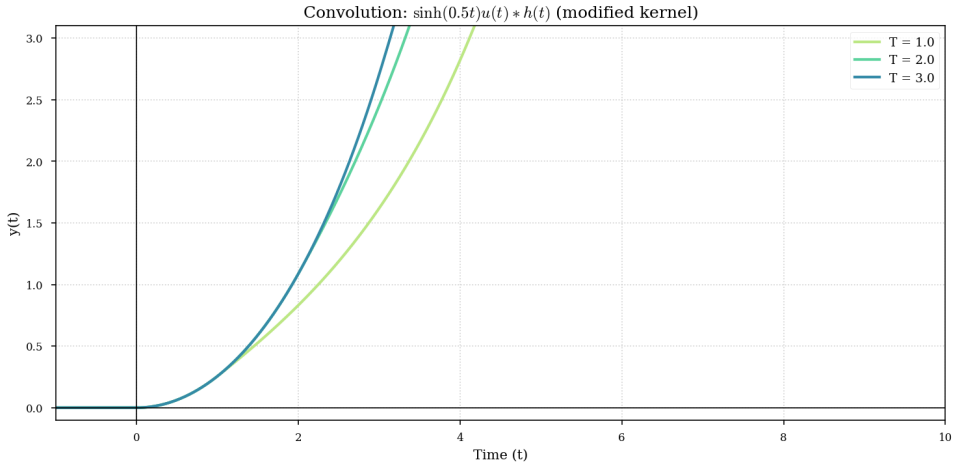



Figure 13: Convolution of $f(t) = \sinh(0.5t)u(t)$ with the modified kernel for various values of $T$. The result is zero for $t < 0$, and for $t > 0$ has exponential growth properties of the hyperbolic sine function. As $T$ grows, the rate of growth becomes faster, with the $T = 3.0$ curve having the most rapid growth. In contrast to the exponential case, these curves still rise without limit after $t = T$, with the pattern $\frac{2sinh(\frac{bT}{2})}{b}\sinh(b(t - \frac{T}{2}))$. All three curves approach each other for small $t$ but spread out considerably as $t$ grows, illustrating how the system exaggerates the input signal more with wider kernels.

## 8.3 Time-Shifted Kernel (Part b)

For part (b), we analyze a kernel shifted by a time $\tau_0$:

$$h_s(t) = \begin{cases} 1, & \text{for } -T + \tau_0 \le t \le T + \tau_0 \\ 0, & \text{otherwise} \end{cases} \tag{33}$$

This is equivalent to:

$$h_s(t) = h(t - \tau_0) \tag{34}$$

### 8.3.1 Exponential Function with Time-Shifted Kernel

The convolution with $f(t) = e^{-at}$ becomes:

$$y_s(t) = \int_{-\infty}^{\infty} e^{-a\tau} \cdot h_s(t-\tau)d\tau = \int_{-\infty}^{\infty} e^{-a\tau} \cdot h(t-\tau-\tau_0)d\tau \tag{35}$$

Using the time-shifting property of convolution, we can write:

$$y_s(t) = y(t-\tau_0) = \frac{2\sinh(aT)}{a} e^{-a(t-\tau_0)} = \frac{2\sinh(aT)}{a} e^{a\tau_0} e^{-at} \tag{36}$$

This shows that shifting the kernel by $\tau_0$ results in a time-shift of the output and an amplitude scaling by $e^{a\tau_0}$.

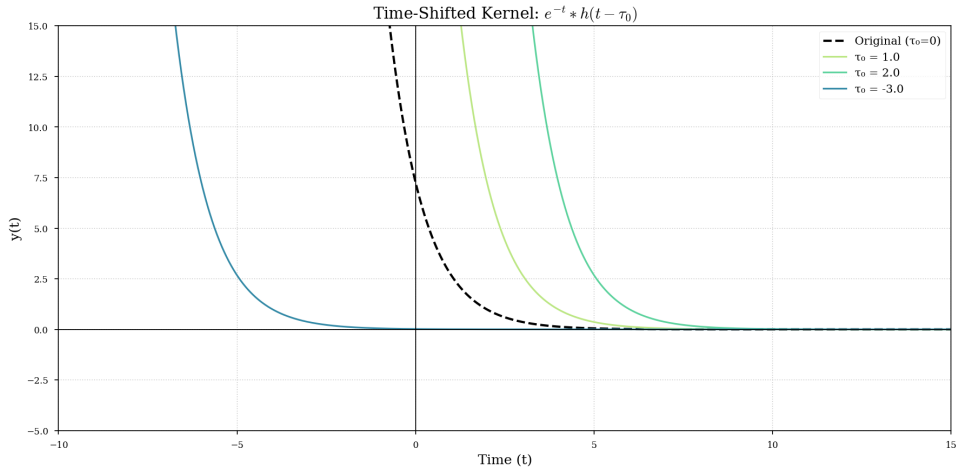



Figure 14: Convolution of $f(t) = e^{-t}$ with time-shifted rectangular kernel $h(t-\tau_0)$ for various values of $\tau_0$. The original response ($\tau_0 = 0$) is represented as a dashed line. As $\tau_0$ is increased, the whole response curve is shifted to the right by $\tau_0$ and scaled by a factor of $e^{\tau_0}$. For $\tau_0 = 1.0$, the curve is shifted right by 1 unit and scales up in amplitude. For $\tau_0 = 2.0$, the shift is 2 units with additional amplification. Similarly, for $\tau_0 = -3.0$, the shift is 3 units to the left. All curves share the typical exponential decay but different initial points and amplitudes, according to the equation $\frac{2\sinh(aT)}{a} e^{a\tau_0} e^{-at}$.

### 8.3.2 Hyperbolic Function with Time-Shifted Kernel

For the hyperbolic function $f(t) = \sinh(bt)$, the convolution with the time-shifted kernel is:

$$y_s(t) = \int_{-\infty}^{\infty} \sinh(b\tau) \cdot h_s(t-\tau)d\tau = \int_{-\infty}^{\infty} \sinh(b\tau) \cdot h(t-\tau-\tau_0)d\tau \tag{37}$$

Using the time-shifting property of convolution, the result is:

$$y_s(t) = y(t-\tau_0) = \frac{2\sinh(bT)}{b} \sinh(b(t-\tau_0)) \tag{38}$$

This represents a pure time-shift of the original response without amplitude scaling, unlike the exponential case.

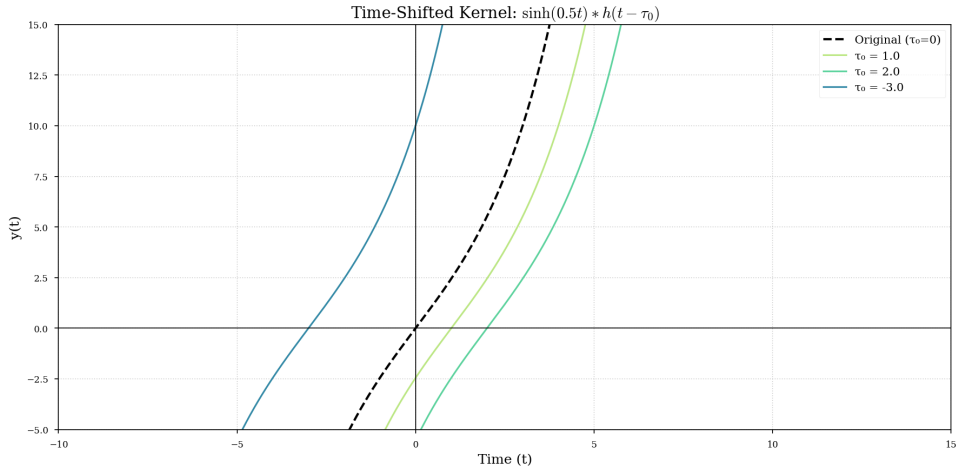



Figure 15: Convolution of $f(t) = \sinh(0.5t)$ with time-shifted rectangular kernel $h(t-\tau_0)$ for various values of $\tau_0$. The original response ($\tau_0 = 0$) is a dashed line. For increasing values of $\tau_0$, the response curves move horizontally to the right by $\tau_0$ units. For $\tau_0 = 1.0$, the curve moves right by 1 unit, for $\tau_0 = 2.0$, the move is 2 units and for $\tau_0 = -3.0$, its 3 units to the left. In contrast to the exponential case, there is no scaling of amplitude - just a simple time-shift, as seen in the parallel curves. Every curve is described by the equation $\frac{2\sinh(bT)}{b}\sinh(b(t-\tau_0))$, with the same hyperbolic growth rate but with varying x-intercepts at $t = \tau_0$.

## 8.4 Comparison of Exponential and Hyperbolic Functions with Time-Shifted Kernels

The time-shifted kernel brings out the key distinction between exponential and hyperbolic functions in convolution systems:

- For the exponential function $f(t) = e^{-at}$, a time-shift in the kernel results in both a time-shift and amplitude scaling in the output. The scaling factor $e^{atau_0}$ grows exponentially with the amount of the shift.
- For the hyperbolic function $f(t) = \sinh(bt)$, a time-shift in the kernel results only in a time-shift in the output and not in amplitude scaling. The curves do not change their shape and are merely shifted horizontally.

# 9 Convolution of sinc function

Given kernel,

$$h(t) = \begin{cases} 1, & \text{for } -T \leq t \leq T \\ 0, & \text{otherwise} \end{cases} \tag{39}$$

and the function

$$\begin{aligned} f(t) &= sinc(t) \\ &= \frac{sint}{t} \end{aligned}$$

Let $y(t) = x(t) * f(t)$, then

$$\begin{aligned} y(t) &= h(t) * f(t) \\ &= f(t) * h(t) \\ &= \int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau \end{aligned}$$

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

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

Therefore, the convolution becomes,

$$\begin{aligned} y(t) &= \int_{t-T}^{t+T} \frac{sin(\tau)}{\tau} * 1 d\tau \\ y(t) &= \int_{t-T}^{t+T} \frac{sin(\tau)}{\tau} d\tau \end{aligned}$$

This integration can be numerically computed and can be seen to be the following -

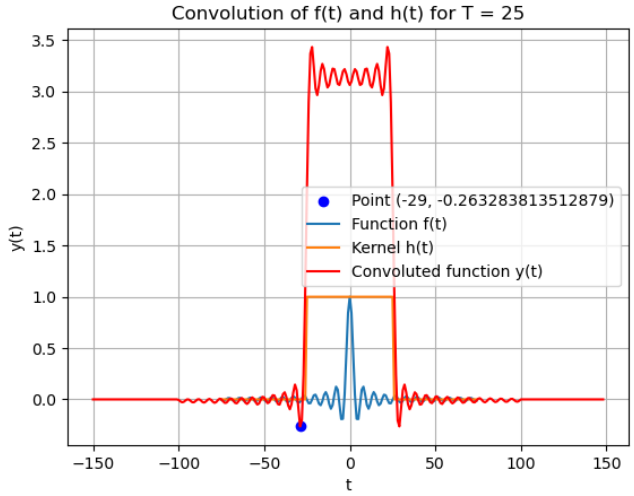



Figure 16: T = 25

For different values of T, the change in $y(t)$ is depicted in the following -

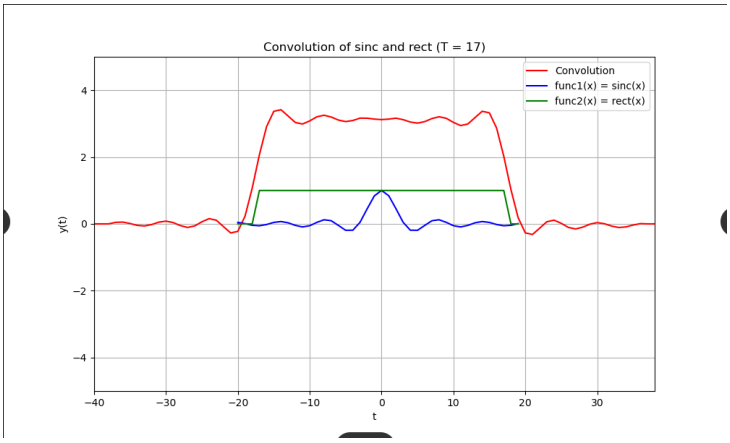



Figure 17: T = 17

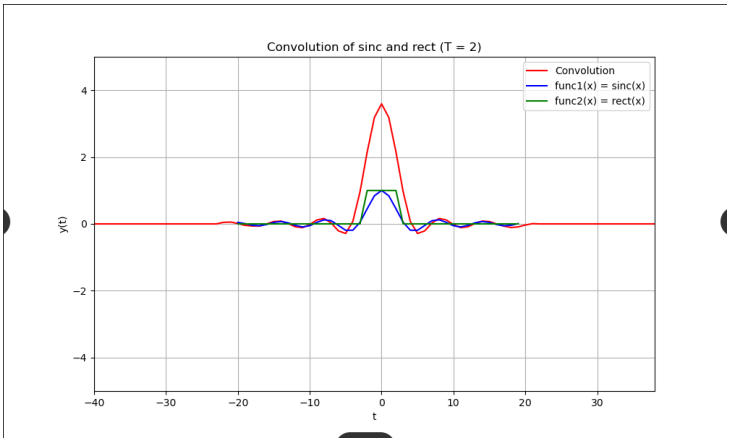



Figure 18: T = 2

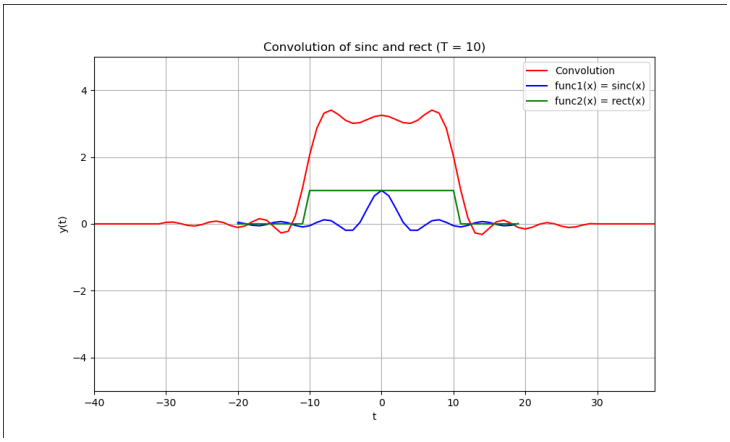



Figure 19: T = 10

It can be seen that as $T$ increases, $y(t)$ becomes more square.

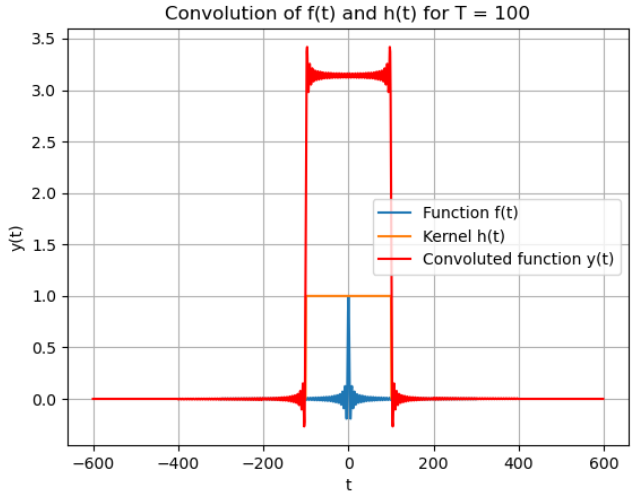



Figure 20: T = 100

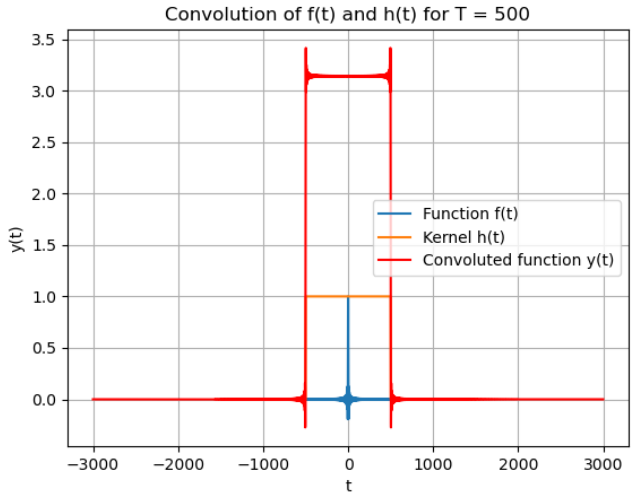



Figure 21: T = 500

## 9.1 Considering the kernel for $t > 0$

The response of the system for different values of $T$ are as follows -

It can be seen that the width of the convolution becomes half and it becomes shifted to the right.

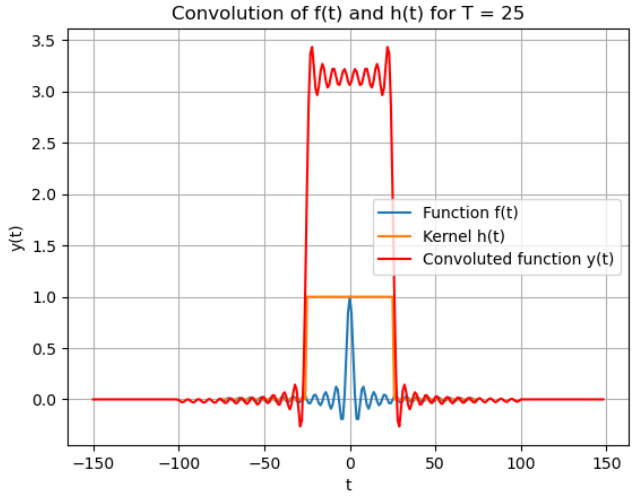



Figure 22: T = 25

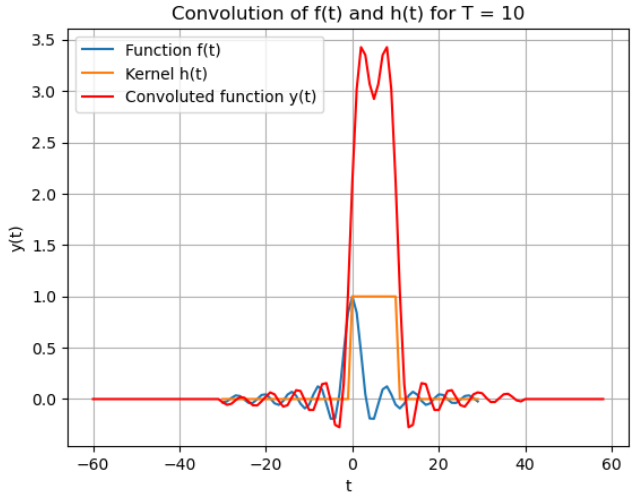



Figure 23: T = 10

## 9.2 Shifting the kernel by $t_0$

Shifting of kernel means moving the kernel to the left or right by an amount, i.e., if the given kernel is shifted by an amount $t_0$, then it becomes,

$$h(t) = \begin{cases} 1, & \text{for } -T \leq t - t_0 \leq T \\ 0, & \text{otherwise} \end{cases}$$

$$h(t) = \begin{cases} 1, & \text{for } -T + t_0 \leq t \leq T + t_0 \\ 0, & \text{otherwise} \end{cases}$$

If we are convolving a signal with the kernel, at each time $t$, we are sliding the kernel over the signal and computing how much they overlap. So, when we shift the kernel by $t_0$, we are changing when the kernel has its maximum influence. This can physically mean applying a response $t_0$ time units early (or) $t_0$ time units late depending upon the sign

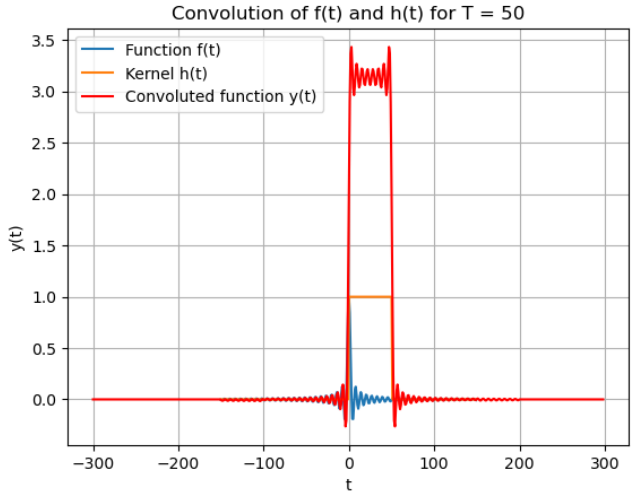



Figure 24: T = 50

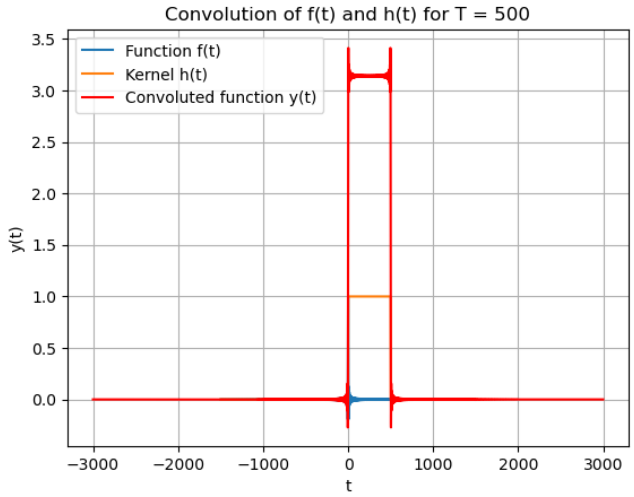
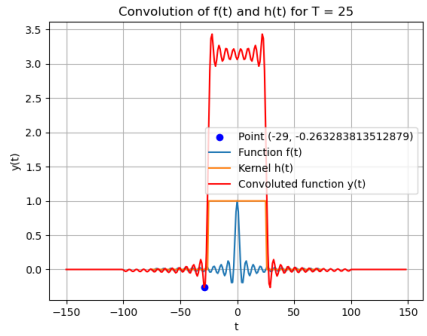



Figure 25: T = 500

of $t_0$, i.e., if $t_0 > 0$, the response is delayed and is advanced for $t_0 < 0$. This results in the convoluted signal also to be shifted by the same amount, as in figures 26 and 27 - Shifting of the kernel by 10 units results in shifting of convoluted function by 10 units.



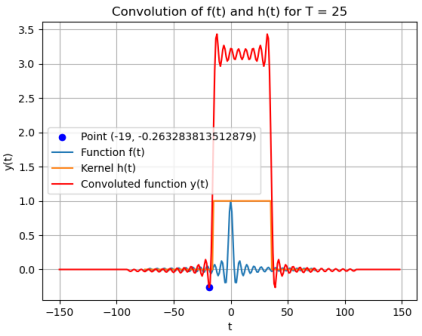
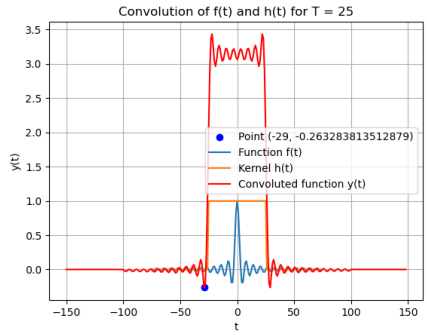




Figure 26: Shifting toward right



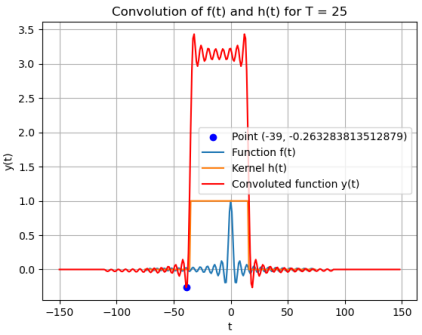




Figure 27: Shifting toward left

# 10 Numerical way to perform convolution

## 10.1 Discrete convolution

Usually when dealing with real life signals, we don't have the complete function in but rather, discreet samples of the function. So it is also useful to look at the discrete convolution, which is given by,

$$y[k] = (f * h)[k] = \sum_{m=0}^{\infty} f[m]h[m-k]$$

Discrete colvolution follow all properties continuous convolution follows. we use this to numeically compute convolution of two signals.

## 10.2 Convolution Theorem

The convolution theorem states that,

If,

$$(u * v)(t) = r(t)$$

Then,

$$\mathcal{L}\{r(t)\} = \mathcal{L}\{u(t)\}\mathcal{L}\{v(t)\}$$

This provides a much faster way to compute convolution of the signals.

## 10.3 Fast Fourier Transform (FFT) and it's inverse (IFFT)

FFT is the algorithm used to find the fourier transform of a given signal. It is a recursive algorithm with a time complexity of $\mathcal{O}(n\log(n))$. To explain the algorithm and explain how convolution theorem works, we can explain it using product of two polynomials.

Let $P_1(x)$ and $P_2(x)$ be polynomials of degree $n$. Let $\vec{p_1}$ and $\vec{p_2}$ be the vectors, whose elements are the coefficients of $P_1(x)$ and $P_2(x)$. If $P(x) = P_1(x) \times P_2(x)$, then the coefficients of $P(x)$ is the discrete convolution of $\vec{p_1}$ and $\vec{p_2}$. We can verify that with some algebra.

It is also known that, through $n$ unique points, there is a unique polynomial of degree $n$ passing through it. Using this property, let us choose $2n$ points on each of $P_1$ and $P_2$, with the x-coordinates. That will give us, $P_{1_p} = [(x_0, P_1(x_0)), (x_1, P_1(x_1), \dots)]$ and $P_{2_p} = [(x_0, P_2(x_0)), (x_1, P_2(x_1), \dots)]$, which we will call the point form of the polynomials. Now if we multiply the results, we will end up with $2n$ unique points that lie on $P(x)$ (i.e., $P_p = [(x_0, P_1(x_0) \times P_2(x_0)), (x_1, P_1(x_1) \times P_2(x_1)), \dots])$. Now if we find the polynomial that is passing through these points, that will give us the polynomial we needed all along. In a nutshell, the process of finding the unique points on each polynomial to be multiplied is FFT and finding the equation of the polynomial from it's point form is IFFT.

Now, to find the point form by chosing random values of x and computing for each of them is pretty inefficient. Thus, we will look at something symmetric, so that the computations can be reduced.

Let,

$$P(x) = a_0x^n + a_1x^{n-1} + \cdots + a_n$$
$$P(x) = P_e(x^2) + xP_o(x^2)$$
$$P(-x) = P_e(x^2) - xP_o(x^2)$$

where, $P_e(x)$ is the polynomial made only with the even terms of the original polynomial, and $P_o(x)$ is the polynomial made only with the odd terms of the polynomial, with the common $x$ term taken out. With this, finding $P(x)$ for n points become simpler, as it will be enough to calulate $P_e$ and $P_o$ n/2 points. It would be really convenient if we could do such splitting for $P_e$ and $P_o$, but $x^2$ breaks the symmetry, and requires it to be complex. Well, whatif, we make it complex? And this is how the FFT algorithm works.

In the simplest case, where $n = 2^k, k \in \mathbb{N}$, we take the $n/2^{th}$ root of 1, and recursively find $P_e$ and $P_o$.

The peudocode of FFT looks like the following:

**Algorithm 1** Fast Fourier Transform (FFT)

```
1:  procedure FFT(x)
2:      n ← |x|                                  ▷ Length of the input array
3:      if n = 1 then
4:          return x
5:      end if
6:      ω ← e^{-2πi/n}                           ▷ Note: negative exponent in Python code
7:      P_e ← (x_0, x_2, ..., x_{n-2})           ▷ Even-indexed elements
8:      P_o ← (x_1, x_3, ..., x_{n-1})           ▷ Odd-indexed elements
9:      y_e ← FFT(P_e)                           ▷ Recursive call on even-indexed elements
10:     y_o ← FFT(P_o)                           ▷ Recursive call on odd-indexed elements
11:     y ← complex array of zeros of length n   ▷ Initialize output array
12:     for i = 0 to n/2 − 1 do
13:         y_i ← y_e[i] + ω^i · y_o[i]          ▷ First half
14:         y_{i+n/2} ← y_e[i] − ω^i · y_o[i]    ▷ Second half
15:     end for
16:     return y
17: end procedure
```

Now, that we finally know how to perform FFT, we can multiply the outputs to get the convolution. But there is a problem with the above algorithm. To calculate the convolution of 2 functions with n discrete samples, our convolved output will have 2n points. But the above algorithm gives n points, and thus multiplying the n points element wise, will only give us n points in the convolved signal. So, we need to modify it such that it gives us 2n points.

Now that we have the fourier transform of our signals, and multiplied them, we have the transformed version of the convolved signal. Now we have to apply the inverse Fourier Transform over it. How are we to do it?

Fourier Transform of a discreet signal can be considered as a matrix multiplication.

**Algorithm 2** Fast Fourier Transform (FFT) - For convolution

**procedure** FFT($x$)
  $n \leftarrow |x|$ ▷ Length of the input array
  **if** $n = 1$ **then**
    **return** $(x, -x)$
  **end if**
  $\omega \leftarrow e^{-\pi i/n}$ ▷ Note: observe it's $2n^{th}$ root of unity.
  $P_e \leftarrow (x_0, x_2, \ldots, x_{n-2})$ ▷ Even-indexed elements
  $P_o \leftarrow (x_1, x_3, \ldots, x_{n-1})$ ▷ Odd-indexed elements
  $y_e \leftarrow \text{FFT}(P_e)$ ▷ Recursive call on even-indexed elements
  $y_o \leftarrow \text{FFT}(P_o)$ ▷ Recursive call on odd-indexed elements
  $y \leftarrow$ complex array of zeros of length $n$ ▷ Initialize output array
  **for** $i = 0$ to $n - 1$ **do**
    $y_i \leftarrow y_e[i] + \omega^i \cdot y_o[i]$ ▷ First half
    $y_{i+n} \leftarrow y_e[i] - \omega^i \cdot y_o[i]$ ▷ Second half
  **end for**
  **return** $y$
**end procedure**

$$\begin{bmatrix} X(\omega^0) \\ X(\omega^1) \\ \vdots \\ X(\omega^n) \end{bmatrix} = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ 1 & \omega & \ldots & \omega^{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{n-1} & \ldots & \omega^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix}$$

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ 1 & \omega & \ldots & \omega^{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{n-1} & \ldots & \omega^{(n-1)(n-1)} \end{bmatrix}^{-1} \begin{bmatrix} X(\omega^0) \\ X(\omega^1) \\ \vdots \\ X(\omega^n) \end{bmatrix}$$

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} = \frac{1}{n} \begin{bmatrix} 1 & 1 & \ldots & 1 \\ 1 & \omega^{-1} & \ldots & \omega^{-(n-1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{-(n-1)} & \ldots & \omega^{-(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} X(\omega^0) \\ X(\omega^1) \\ \vdots \\ X(\omega^n) \end{bmatrix}$$

Notice how the matrix looks very identical except that the roots of unity are inverted, and the whole matrix is divided by n. So with minor modifications to the FFT algorithm, we can perform IFFT too.

**Algorithm 3** Inverse Fast Fourier Transform

**procedure** IFFT-RECURSIVE($x$)
    $n \leftarrow |x|$ ▷ Length of the input array
    **if** $n = 1$ **then**
        **return** $x$
    **end if**
    $\omega \leftarrow e^{2\pi i/n}$ ▷ Note: opposite sign that of FFT for IFFT
    $P_e \leftarrow (x_0, x_2, \ldots, x_{n-2})$ ▷ Even-indexed elements
    $P_o \leftarrow (x_1, x_3, \ldots, x_{n-1})$ ▷ Odd-indexed elements
    $y_e \leftarrow$ IFFT-Recursive($P_e$) ▷ Recursive call on even-indexed elements
    $y_o \leftarrow$ IFFT-Recursive($P_o$) ▷ Recursive call on odd-indexed elements
    $y \leftarrow$ complex array of zeros of length $n$ ▷ Initialize output array
    **for** $i = 0$ to $n/2 - 1$ **do**
        $y_i \leftarrow y_e[i] + \omega^i \cdot y_o[i]$ ▷ First half
        $y_{i+n/2} \leftarrow y_e[i] - \omega^i \cdot y_o[i]$ ▷ Second half
    **end for**
    **return** $y$
**end procedure**
**procedure** IFFT($X$)
    $y \leftarrow$ IFFT-Recursive($X$)
    **return** $y/|X|$ ▷ Scale by $1/n$
**end procedure**

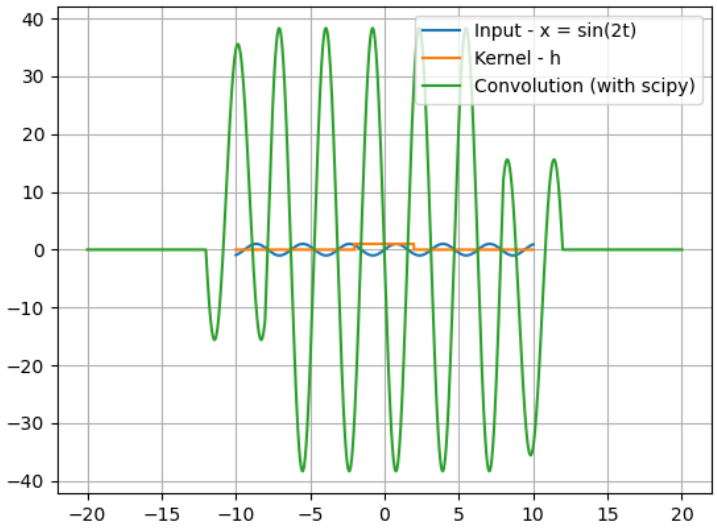



Figure 28: Convolution by scipy

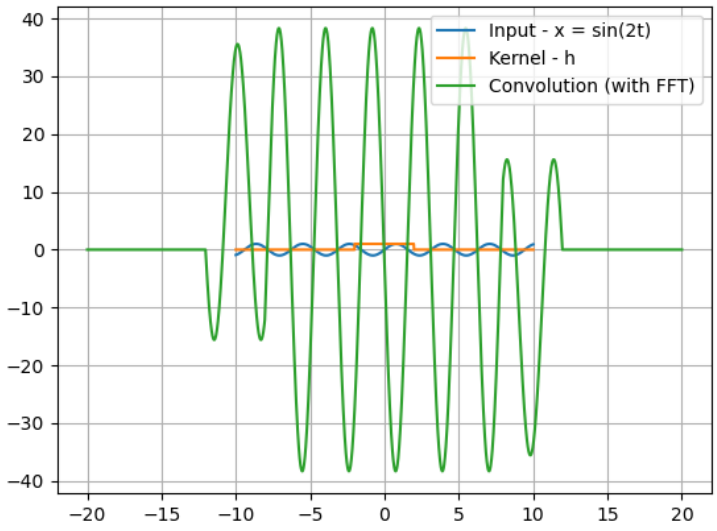



Figure 29: Convolution with python-implementation of FFT

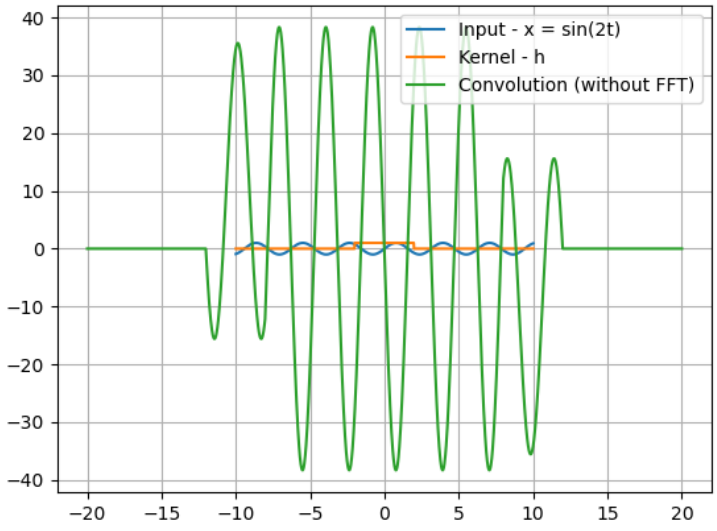



Figure 30: Convolution without FFT and IFFT algorithm

As we can see, the convolution matches.

Now comparing the runtime of each algorithm, `Discreet Convolution: 0.33714938163757324`
`Scipy's Discreet Convolution 0.0010726451873779297`
`Discreen Convolution (FFT): 0.03110957145690918`

Scipy's implementation is the fastest, as it is a professional tool with a lots of optimizations made. But as we can see, for the $n = 1024$, FFT algorithm made convolution almost 10 times faster than regular convolution. Approximately, the time complexity of regular convolution is $\mathcal{O}(n^2)$ and FFT is $\mathcal{O}(n \log n)$, which approximately matches.

# 11 Applications of convolution - Filters ( Image processing )

## 11.1 Gaussian filter

1. A Gaussian filter is a type of digital filter that utilizes a Gaussian distribution function to smooth and reduce noise in signals or images. It achieves this by averaging neighboring values with weights based on a Gaussian curve, giving more weight to the central value and less to the outer values.

2. One dimensional Gaussian -

$$G_1(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{40}$$

   Two-dimensional Gaussian -

$$G_2(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{41}$$

3. **How it works ?**

   - A Gaussian filter, is a low-pass filter, that uses a kernel that represents the weighting of neighboring pixels.
   - The kernel is then convolved with the image ( i.e., it is moved across the image ), and the values of the kernel are multiplied by the corresponding pixel values in the image.
   - The products are then summed, and the result is divided by the sum of the kernel values, producing a weighted average of the pixel's neighborhood.
   - The weighted average becomes the new value of the pixel at the center of the kernel's position, smoothing out the image.

4. The corresponding figure is shown below -

5. In the case of Figure 33, since the kernel itself is bell-shaped, convolving a narrow Gaussian will produce an output being very similar to the original signal, just slightly smoothed or blurred.

6. Gaussian filters are used because -

   - Smooth
   - Decay to zero rapidly
   - Simple analytical formula

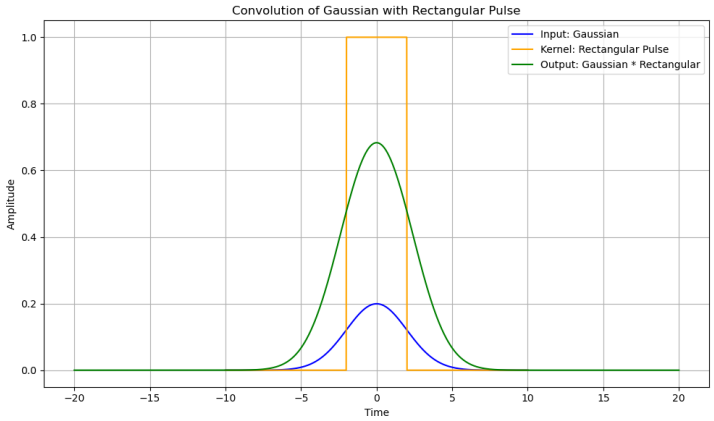



Figure 31: T = 2

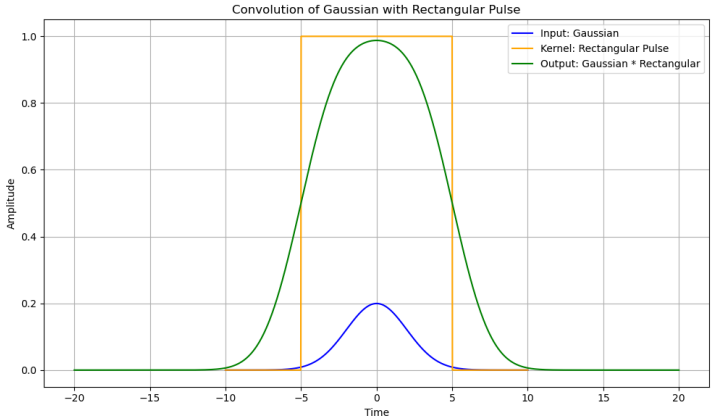



Figure 32: T = 5

- **Central Limit Theorem :** Limit of applying filters many times is some Gaussian.
- Separable

$$G_2(x, y) = G_1(x)G_1(y)$$

## 11.2 Laplacian filter

1. A Laplacian filter is a second-order derivative operator that is commonly used in image processing for edge detection and enhancement.

2. It is often implemented using a convolution kernel, which is a small matrix that is slid over the image and multiplied with the surrounding pixels to produce a new pixel value.

3. The Laplacian filter, when combined with a Gaussian blur ( Laplacian of Gaussian ), can be used to detect edges in an image.

4. Image corresponding to this is given below -

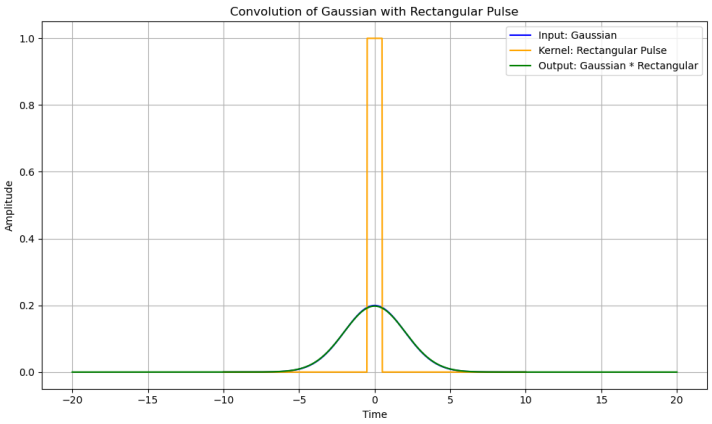



Figure 33: T = 0.5

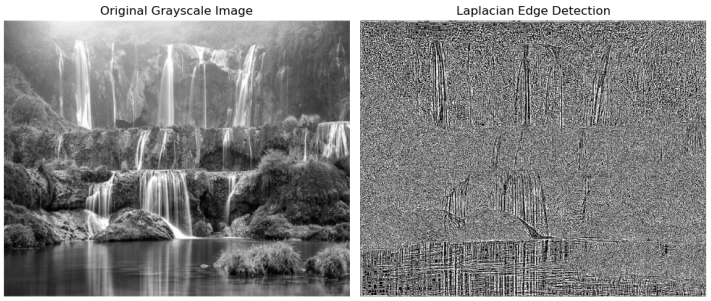



Figure 34: Edge detection using convolution

# 12 References

https://en.wikipedia.org/wiki/Gaussian$_filter$
https://www.sciencedirect.com/topics/engineering/laplacian-filter
https://www.cs.princeton.edu/courses/archive/fall16/cos429/notes/cos429$_f$16$_lecture$03$_filtering.pdf$
https://people.iith.ac.in/seshadri/Courses/DETT/DETT-2025-M.html

# 13 Github

Refer this link for the entire documentation -
https://github.com/me-coder-1204/EE1060-GQ2.git