CONVOLUTION OF EXPONENTIAL FUNCTION

clear all; %

close all;

clc;

stepSize = 0.005; % Step size

t = -5:stepSize:5; % Time axis

length\_t = length(t); % Length of time axis

midpoint = (length\_t - 1) / 2; % Midpoint of the time axis

x\_t = zeros(1, length\_t); % Original signal

y\_t = zeros(1, length\_t); % Resultant signal

h\_t = zeros(1, length\_t); % Impulse response

% Generate the original signal x(t)

for i = 1:length\_t

if t(i) < 0

x\_t(i) = 0; % Exponential signal starts at t=0

else

x\_t(i) = exp(-2 \* t(i));

end

end

% Generate the impulse response h(t)

for i = 1:length\_t

if t(i) <= 0

h\_t(i) = exp(4 \* t(i)); % Impulse response for t <= 0

else

h\_t(i) = 0;

end

end

% Flip the impulse response h(t)

h\_t\_flipped = fliplr(h\_t);

% Initialize the convolution loop

for i = 1:length\_t

delayed\_signal = zeros(1, length\_t); % Initialize delayed signal

To = t(i); % Delay time

% Calculate the index for the delay

if To < 0

newIndex = midpoint - i;

if newIndex > 0

delayed\_signal(newIndex:end) = h\_t\_flipped(1:(length\_t-newIndex+1));

end

else

newIndex = i - midpoint;

if newIndex > 0

delayed\_signal(1:(length\_t-newIndex)) = h\_t\_flipped(newIndex:end);

end

end

% Plot the original and delayed signals

subplot(2,1,1);

plot(t, x\_t, 'b', 'DisplayName', 'Original Signal');

hold on;

plot(t, delayed\_signal, 'r', 'DisplayName', 'Delayed Signal');

hold off;

legend;

pause(0.005);

% Multiply the signals element-wise

temp = x\_t .\* delayed\_signal;

% Numerical integration (trapezoidal rule)

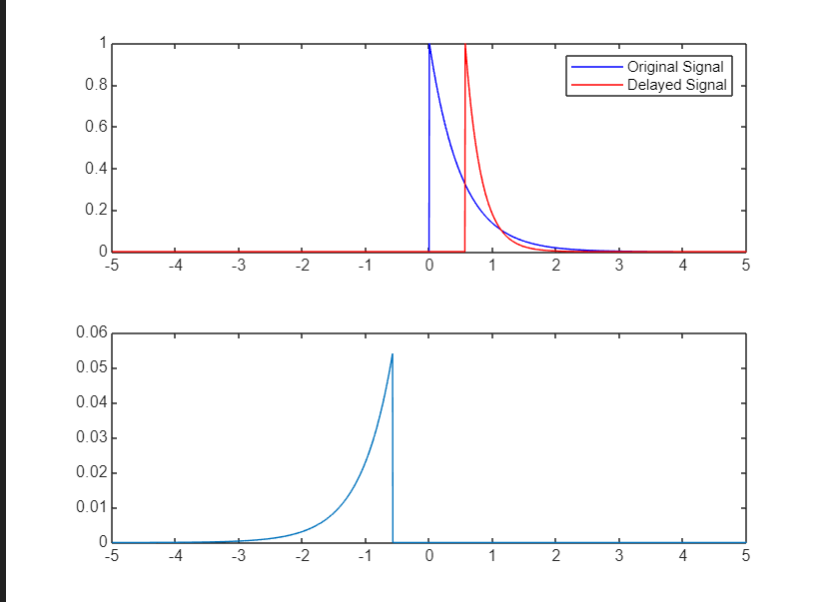
y\_t(i) = (stepSize / 2) \* (temp(1) + temp(end) + 2 \* sum(temp(2:end-1)));

% Plot the convolution result

subplot(2,1,2);

plot(t, y\_t, 'DisplayName', 'Convolution Result');

end



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CONVOLUTION OF UNIT STEP FUNCTION

clear all;

close all;

clc;

stepSize = 0.005; % Setting step size

t = -5:stepSize:5; % Creating time axis

length\_t = length(t); % Length of the time axis

midpoint = (length\_t - 1) / 2; % Midpoint of the time axis

x\_t = zeros(1, length\_t); % Original signal

y\_t = zeros(1, length\_t); % Resultant signal

h\_t = zeros(1, length\_t); % Impulse response

% Generate the original signal x(t)

for i = 1:length\_t

if t(i) < 0

x\_t(i) = 0; % Step signal starting from t=0

else

x\_t(i) = 1; % Step response for t >= 0

end

end

% Generate the impulse response h(t)

for i = 1:length\_t

if t(i) <= 0

h\_t(i) = 0; % Impulse response for t <= 0

else

h\_t(i) = 1; % Impulse response for t > 0

end

end

% Flip the impulse response h(t)

h\_t\_flipped = fliplr(h\_t);

% Loop for delaying the impulse response, multiplying with x\_t, and integrating

for i = 1:length\_t

delayed\_signal = zeros(1, length\_t); % Initialize delayed signal

To = t(i); % Delay time

index = 0;

% Find the index where t(ii) <= To

for ii = 1:length\_t

if t(ii) <= To

index = index + 1;

else

break;

end

end

% Handle the delay: shift the flipped impulse response accordingly

if To < 0

newIndex = midpoint - index; % Compute shift for negative To

for ii = 1:length\_t

if (ii + newIndex) > 0 && (ii + newIndex) <= length\_t

delayed\_signal(ii + newIndex) = h\_t\_flipped(ii);

end

end

else

newIndex = index - midpoint; % Compute shift for positive To

for ii = 1:length\_t

if (ii + newIndex) <= length\_t && (ii + newIndex) > 0

delayed\_signal(ii + newIndex) = h\_t\_flipped(ii);

end

end

end

% Plot the original and delayed signals

subplot(2, 1, 1);

plot(t, x\_t, 'b', 'DisplayName', 'Original Signal');

hold on;

plot(t, delayed\_signal, 'r', 'DisplayName', 'Delayed Signal');

hold off;

legend;

pause(0.005); % Pause for visualization

% Multiply x\_t and delayed\_signal element-wise

temp = x\_t .\* delayed\_signal;

% Numerical integration using trapezoidal rule

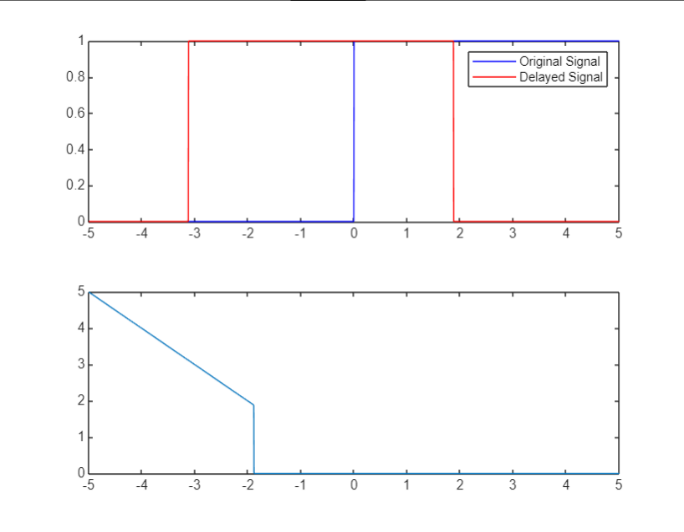
y\_t(i) = (stepSize / 2) \* (temp(1) + temp(end) + 2 \* sum(temp(2:end-1)));

% Plot the convolution result

subplot(2, 1, 2);

plot(t, y\_t, 'DisplayName', 'Convolution Result');

end



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CONVOLUTION OF TRIANGULAR WAVE

clear all;

close all;

clc;

stepSize = 0.005; % Setting step size

t = -5:stepSize:5; % Creating time axis

length\_t = length(t); % Length of time axis

midpoint = (length\_t - 1) / 2; % Midpoint of time axis

x\_t = zeros(1, length\_t); % Original signal

y\_t = zeros(1, length\_t); % Resultant signal

% Generate the original signal x(t) (Triangular shape signal)

for i = 1:length\_t

if (abs(t(i)) <= 1)

x\_t(i) = 1 - abs(t(i)); % Triangular signal for |t| <= 1

else

x\_t(i) = 0; % Zero elsewhere

end

end

% Generate the impulse response h(t)

for i = 1:length\_t

if (abs(t(i)) <= 0.5)

h\_t(i) = 1; % Impulse response for |t| <= 0.5

else

h\_t(i) = 0; % Zero elsewhere

end

end

% Flip the impulse response

h\_t\_flipped = fliplr(h\_t);

% Delay, multiply and integrate the signal

for i = 1:length\_t

delayed\_signal = zeros(1, length\_t); % Initialize delayed signal

To = t(i); % Current delay

% Calculate the index for delay based on To

if To < 0

newIndex = midpoint - i; % Adjust index for negative To

for ii = 1:length\_t

if (ii + newIndex) > 0 && (ii + newIndex) <= length\_t

delayed\_signal(ii + newIndex) = h\_t\_flipped(ii);

end

end

else

newIndex = i - midpoint; % Adjust index for positive To

for ii = 1:length\_t

if (ii + newIndex) <= length\_t && (ii + newIndex) > 0

delayed\_signal(ii + newIndex) = h\_t\_flipped(ii);

end

end

end

% Plot the original signal and delayed signal

subplot(2, 1, 1);

plot(t, x\_t, 'b', 'DisplayName', 'Original Signal');

hold on;

plot(t, delayed\_signal, 'r', 'DisplayName', 'Delayed Signal');

hold off;

legend;

pause(0.005); % Pause for visualization

% Multiply the original and delayed signals element-wise

temp = x\_t .\* delayed\_signal;

% Numerical integration using trapezoidal rule

y\_t(i) = (stepSize / 2) \* (temp(1) + temp(end) + 2 \* sum(temp(2:end-1)));

% Plot the convolution result

subplot(2, 1, 2);

plot(t, y\_t, 'DisplayName', 'Convolution Result');

end

