Aim ↬ To understand and implement the Continuous-Time Fourier Transform (CTFT) and its inverse for analyzing and synthesizing continuous-time signals using MATLAB.

Software Required ↬ MATLAB

Theory ↬

The Continuous-Time Fourier Transform (CTFT) is an essential mathematical tool in signal processing and analysis, used to transform a time-domain signal into its frequency-domain representation. This transformation provides insight into the frequency content of signals, which is crucial for applications in communications, audio processing, and systems analysis.

The CTFT of a continuous-time signal f(t) is defined as:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$$

where:

- F(ω) is the Fourier transform of f(t), representing the signal in the frequency domain.
- ω is the angular frequency, related to the frequency f (in Hz) by ω=2πf.

This integral transforms the signal from the time domain (where it is a function of time t) to the frequency domain (where it is a function of angular frequency ω). The Fourier Transform decomposes the signal into its constituent sinusoidal components, each characterized by a specific frequency, amplitude, and phase.

Properties of CTFT ↴

1] Linearity ↪

$$F\{af_1(t) + bf_2(t)\} = aF_1(\omega) + bF_2(\omega)$$

2] Time Shifting ↪

$$F\{f(t - t_0)\} = F(\omega)e^{-j\omega t_0}$$

3] Frequency Shifting ↪

$$F\{f(t)e^{j\omega_0 t}\} = F(\omega - \omega_0)$$

4] Convolution ↪

$$F\{f(t) * g(t)\} = F(\omega) \cdot G(\omega)$$

5] Duality ↪

$$F\{f(t)\} = F(\omega) \quad \text{implies} \qquad F\{F(t)\} = 2\pi f(-\omega)$$

6] Time Scaling ↪

$$F\{f(at)\} = \frac{1}{|\,a\,|} F\left(\frac{\omega}{a}\right)$$

7] Differentiation in Time Domain ↪

$$F\left\{\frac{d^n f(t)}{dt^n}\right\} = (j\omega)^n F(\omega)$$

8] Integration in Time Domain ↪

$$F\left\{\int_{-\infty}^{t} f(\tau)\, d\tau\right\} = \frac{F(\omega)}{j\omega}$$

9] Parseval's Energy Theorem ↪

$$E = \int_{-\infty}^{\infty} |f(t)|^2 dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} |F(\omega)|^2 d\omega$$

CTFT of Basic Signals ↴

x(t)	X(s)
δ(t)	1
u(t)	$\frac{1}{j\omega} + \pi\delta(\omega)$
$e^{at}u(t)$	$\frac{1}{j\omega - a}$
rect(t/τ)	$\tau\,\text{sinc}\left(\frac{\omega\tau}{2\pi}\right)$
tri(t/τ)	$\tau\left[\frac{\sin^2\left(\frac{\omega\tau}{2}\right)}{\left(\frac{\omega\tau}{2}\right)^2}\right]$
e^{-at^2}	$\sqrt{\frac{\pi}{a}}\, e^{-\frac{\omega^2}{4a}}$

$\cos(\omega_0 t)$	$\pi[\delta(\omega-\omega_0)+\delta(\omega+\omega_0)$
$\sin(\omega_0 t)$	$\frac{\pi}{j}[\delta(\omega-\omega_0)+\delta(\omega+\omega_0)$

Inverse Fourier Transform ↴

The Inverse Fourier Transform allows the reconstruction of the original time-domain signal from its frequency-domain representation:

$$f(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty}F(\omega)e^{j\omega t}d\omega$$

This inverse transform sums all the sinusoidal components (with their respective amplitudes and phases) to reconstruct the original signal. The ability to move back and forth between the time and frequency domains makes the Fourier Transform a versatile tool in engineering and physics.

In practical applications, signals are often sampled and finite in duration, which leads to the use of the Discrete Fourier Transform (DFT) and its fast computation algorithm, the Fast Fourier Transform (FFT). However, the underlying theory and interpretation remain grounded in the continuous Fourier Transform.

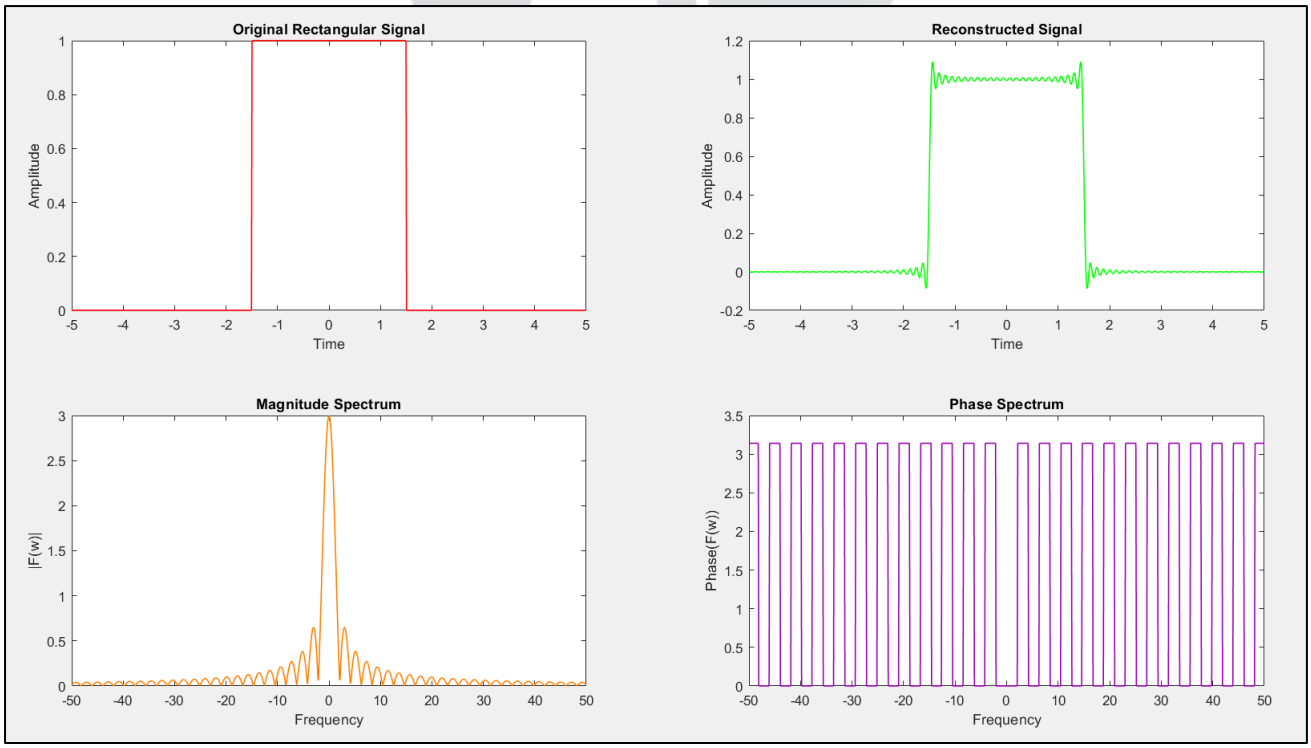
Code ↬

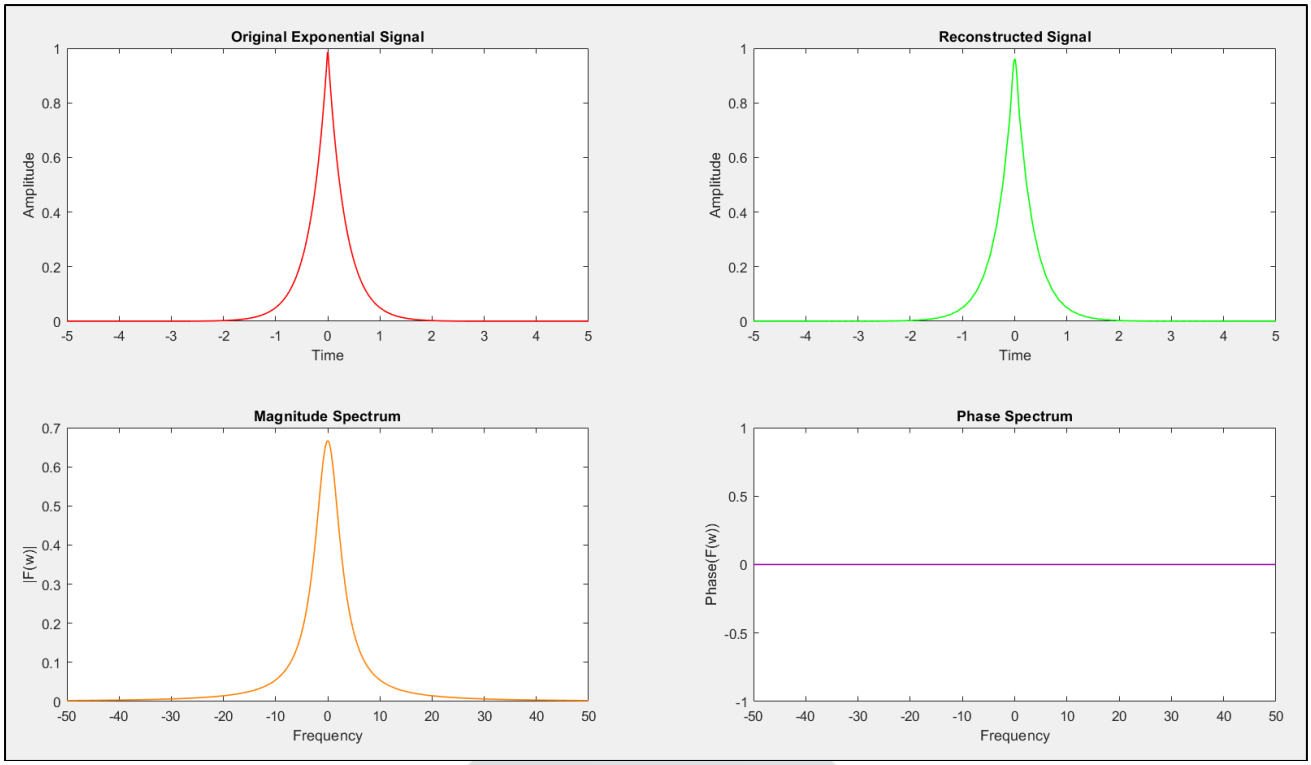
```
%Continuous Time Fourier Transform
syms t w a T

T = 3;
a = 3;

f_rect = rectangularPulse(-T/2, T/2, t);
f_exp = exp(-a * abs(t));

signals = {f_rect, 'Rectangular Signal', [];
          f_exp, 'Exponential Signal', a};

for i = 1:size(signals, 1)
    f = signals{i, 1};
    name = signals{i, 2};
    param = signals{i, 3};
```

```
    if isempty(param)
        FT = fourier(f, t, w);
    else
        FT = fourier(subs(f, a, param), t, w);
    end

    disp(['Fourier Transform of ', name, ':']);
    disp(simplify(FT));

    FT_num = matlabFunction(FT, 'Vars', w);

    Fs = 1000;
    L = 1000;
    t0 = linspace(-5, 5, L);
    w0 = linspace(-50, 50, L);

    if isempty(param)
        f0 = double(subs(f, t, t0));
    else
        f0 = exp(-param * abs(t0));
    end

    FT_vals = FT_num(w0);

    mag = abs(FT_vals);
    ph = angle(FT_vals);

    f_remake = zeros(1, L);
    for k = 1:L
        f_remake(k) = (1/(2 * pi)) * integral(@(w) real(FT_num(w) .* exp(1i * w * t0(k))), -50,
50);
    end

    figure;

    subplot(2,2,1);
    plot(t0, f0, 'r', 'LineWidth', 1);
    xlabel('Time');
    ylabel('Amplitude');
    title(['Original ', name]);
```

```
    subplot(2,2,2);
    plot(t0, f_remake, 'g', 'LineWidth', 1);
    xlabel('Time');
    ylabel('Amplitude');
    title('Reconstructed Signal');

    subplot(2,2,3);
    plot(w0, mag, 'Color', [1, 0.5, 0], 'LineWidth', 1);
    xlabel('Frequency');
    ylabel('|F(w)|');
    title('Magnitude Spectrum');

    subplot(2,2,4);
    plot(w0, ph, 'Color', [0.6, 0, 0.7], 'LineWidth', 1);
    xlabel('Frequency');
    ylabel('Phase(F(w))');
    title('Phase Spectrum');
end
```

Output ↬






```
Command Window | Workspace

>> Continuous_Time_Fourier_Transform
Fourier Transform of Rectangular Signal:
(2*sin((3*w)/2))/w

Fourier Transform of Exponential Signal:
6/(w^2 + 9)
```

Result ↬

The code successfully computes and plots the Fourier Transforms and their inverses for a square wave and an exponential decay function. The original and reconstructed signals match well, confirming accurate transformation.

Conclusion ↬

The CTFT and its inverse effectively analyze and synthesize continuous-time signals, enabling accurate frequency domain analysis and signal reconstruction.

Precautions ↬

- Ensure proper sampling to avoid aliasing.
- Normalize reconstructed signals to match the original amplitude.
- Verify integration and transformation calculations for accuracy.