Aim ↬ To understand and implement the Continuous Time Fourier Series and its inverse in order to analyze and synthesize periodic signals using MATLAB.

Software Required ↬ MATLAB

Theory ↬

The Continuous Time Fourier Series is a powerful mathematical tool used to express a periodic function as an infinite sum of sines and cosines. This decomposition is based on the principle that any periodic function, regardless of its shape, can be represented by a series of sinusoidal components, each with specific amplitudes and frequencies.

Introduced by Jean-Baptiste Joseph Fourier in the early 19th century, this theory has become fundamental in fields such as signal processing, electrical engineering, and applied mathematics. By representing a complex periodic function as a series of simple waves, we can more easily analyze, manipulate, and understand the behavior of the function.

A periodic function f(t) with period T can be written as:

$$f(t) = a_0 + \sum_{n=1}^{\infty} [a_n \boldsymbol{cos}\left(\frac{\boldsymbol{2\pi nt}}{\boldsymbol{T}}\right) + b_n \boldsymbol{sin}\left(\frac{\boldsymbol{2\pi nt}}{\boldsymbol{T}}\right)]$$

where a_0 is the average value of the function over one period, and a_n and b_n are the Fourier coefficients given by:

$$a_0 = \frac{1}{T}\int_0^T f(t)dt$$

$$a_n = \frac{2}{T}\int_0^T f(t)\boldsymbol{cos}\left(\frac{\boldsymbol{2\pi nt}}{\boldsymbol{T}}\right)dt$$

$$b_n = \frac{2}{T}\int_0^T f(t)\boldsymbol{sin}\left(\frac{\boldsymbol{2\pi nt}}{\boldsymbol{T}}\right)dt$$

These coefficients a_n and b_n quantify the contribution of each sinusoidal component to the overall function. By summing these components, we can reconstruct the original function. This series converges to the function f(t) at points where f(t) is continuous and to the average of the left-hand and right-hand limits at points of discontinuity. Fourier Series are instrumental in analyzing complex waveforms and are foundational in the study of Fourier Transforms, which generalize these concepts to non-periodic functions.

Properties of Continuous Time Fourier Series ↴

1] Linearity ↪

$$af(t) + bg(t) \leftrightarrow aC_n + bD_n$$

2] Time Shifting ↪

$$f(t - t_0) \leftrightarrow C_n e^{-jn\omega_0 t_0}$$

3] Frequency Shifting ↪

$$f(t)e^{jm\omega_0 t} \leftrightarrow C_{n-m}$$

4] Time Scaling ↪

$$f(at) \leftrightarrow C_n|_{\frac{T_0}{a}\,\&\,a\omega_0}$$

5] Complex Conjugate ↪

$$f^*(t) \leftrightarrow C^*{}_{-n}$$

6] Convolution ↪

$$f(t) * g(t) \leftrightarrow C_n . D_n . T$$

7] Multiplication in Time Domain ↪

$$f(t).g(t) \leftrightarrow C_n * D_n$$

8] Differentiation in Time Domain ↪

$$\frac{d^n f(t)}{dt^n} \leftrightarrow (jn\omega_0)^n C_n$$

9] Integration in Time Domain ↪

$$\int_{-\infty}^{t} f(\tau)\, d\tau \leftrightarrow \frac{C_n}{jn\omega_0}$$

10] Parseval's Power Property ↪

$$P = \frac{1}{T}\int_{T} |f(t)|^2 dt = \sum_{n=-\infty}^{\infty} |C_n|^2$$

11] FS of Periodic Impulse Train ↪

$$\sum_{k=-\infty}^{\infty} \delta(t - kT) \leftrightarrow C_n = \frac{1}{T}$$

Wave	Series	Fourier Series Grapher
Square wave	sin(x)+sin(3x)/3+sin(5x)/5+...	$\frac{sin((2n-1)*x)}{2n-1}$
Sawtooth	sin(x)+sin(2x)/2+sin(3x)/3+...	$\frac{sin(nx)}{n}$
Pulse	sin(x)+sin(2x)+sin(3x)+...	$sin(nx)*0.1$
Triangle	sin(x)-sin(3x)/9+sin(5x)/25+...	$\frac{(-1)^n*sin((2n-1)*x)}{(2n-1)^2}$

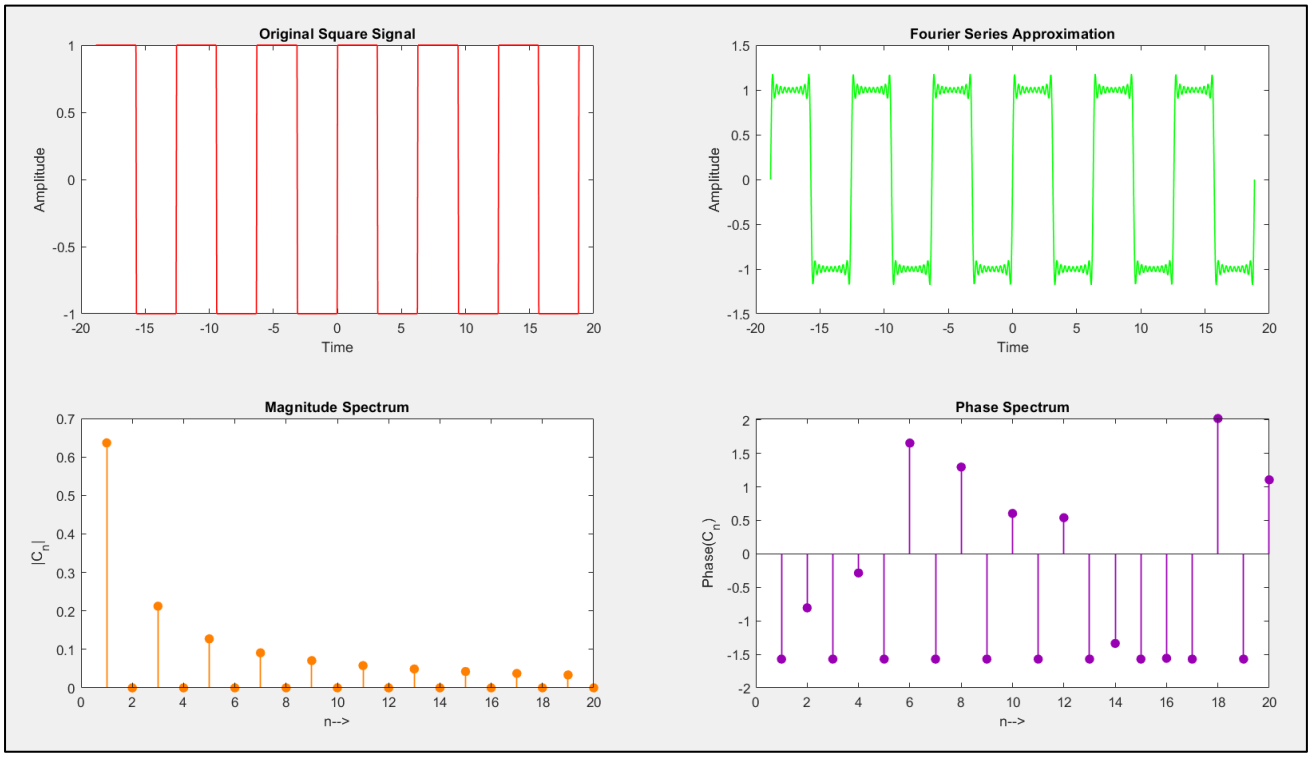
The Inverse Fourier Series allows us to reconstruct the original periodic function from its Fourier coefficients. Using the coefficients a_0, a_n, and b_n we can accurately recreate and study the original periodic function. The Fourier Series and its inverse are key tools for analyzing complex waveforms and are the basis for Fourier Transforms, which extend these ideas to non-periodic functions.

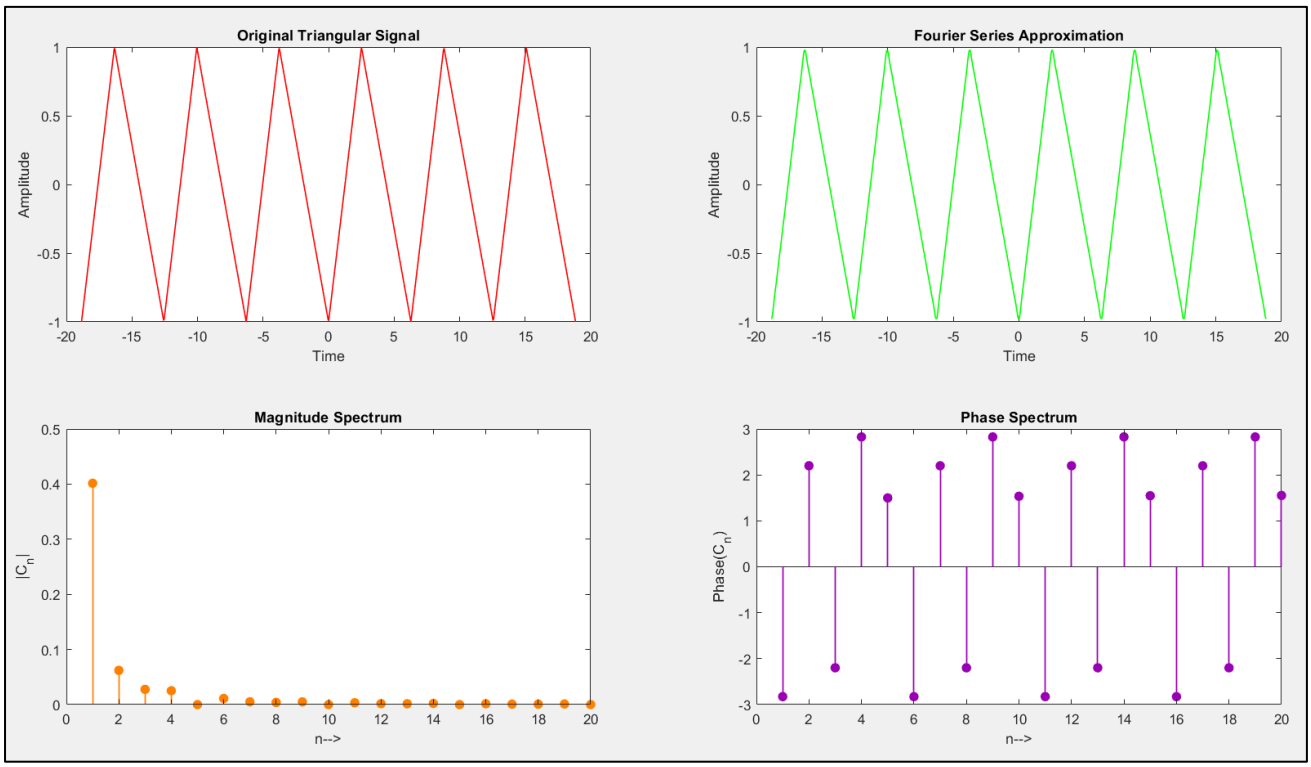
Code ↬

```
%Continuous Time Fourier Series
T = 2 * pi;
N = 20;
t = linspace(-3*T, 3*T, 1000);

signals = {
    @(t) square(t), 'Square Signal';
    @(t) sawtooth(t, 0.4), 'Triangular Signal';
};

for k = 1:size(signals, 1)
    f = signals{k, 1};
    signalName = signals{k, 2};

    a0 = (1/T) * integral(@(t) f(t), 0, T);
    f_approx = zeros(size(t));
    f_approx = f_approx + a0;

    an_values = zeros(1, N);
    bn_values = zeros(1, N);
    C_n = zeros(1, N);
```

```
    for n = 1:N
        an = (2/T) * integral(@(t) f(t) .* cos((2*pi*n*t)/T), 0, T);
        bn = (2/T) * integral(@(t) f(t) .* sin((2*pi*n*t)/T), 0, T);
        an_values(n) = an;
        bn_values(n) = bn;

        C_n(n) = (an / 2) - (1i * bn / 2);

        f_approx = f_approx + an * cos((2*pi*n*t)/T) + bn * sin((2*pi*n*t)/T);
    end

    magnitude = abs(C_n);
    phase = angle(C_n);

    figure;
    subplot(2,2,1);
    plot(t, f(t), 'r', 'LineWidth', 1);
    xlabel('Time');
    ylabel('Amplitude');
    title(['Original ', signalName]);

    subplot(2,2,2);
    plot(t, f_approx, 'g', 'LineWidth', 1);
    xlabel('Time');
    ylabel('Amplitude');
    title('Fourier Series Approximation');

    subplot(2,2,3);
    stem(1:N, magnitude, 'filled', 'Color', [1, 0.5, 0], 'LineWidth', 1);
    xlabel('n-->');
    ylabel('|C_n|');
    title('Magnitude Spectrum');

    subplot(2,2,4);
    stem(1:N, phase, 'filled', 'Color', [0.6, 0, 0.7], 'LineWidth', 1);
    xlabel('n-->');
    ylabel('Phase(C_n)');
    title('Phase Spectrum');
end
```

Output ↬


Original Square Signal
Amplitude
Time
Fourier Series Approximation
Amplitude
Time
Magnitude Spectrum
|C_n|
n-->
Phase Spectrum
Phase(C_n)
n-->


Original Triangular Signal
Amplitude
Time
Fourier Series Approximation
Amplitude
Time
Magnitude Spectrum
|C_n|
n-->
Phase Spectrum
Phase(C_n)
n-->

Result ↬

The Continuous Time Fourier Series and its inverse were implemented in MATLAB to analyze and synthesize periodic signals. The Fourier Series was applied to a square wave and a triangular wave, and the resulting approximations closely matched the original signals.

Conclusion ↬

The experiment demonstrated the effectiveness of the Fourier Series in representing complex periodic functions using simple sinusoidal components. This method is essential in signal processing and electronics2 engineering for analyzing and understanding the frequency components of signals.

Precautions ↬

- Ensure integration accuracy by using a sufficient number of points in the time vector for precise Fourier coefficients.
- Choose an appropriate number of coefficients to balance accuracy and efficiency.
- Be aware of MATLAB's numerical limits, especially for discontinuous functions.