Fourier Analysis of Non-linear Signal

(A study on signal reconstruction using Fourier Series)

A Dissertation submitted to

JAMIA MILLIA ISLAMIA

New Delhi

in partial fulfillment of the requirement for the course of

Numerical Analysis and Programming

in

Physics

by

Shashikant Kumar (Roll:24MPH035, Sem:3rd)

Department of Physics
Jamia Millia Islamia, New Delhi.

November 21, 2025

ACKNOWLEDGEMENT

I take this opportunity to thank **Dr. Anver Aziz** and **Dr. Syed Rashid Ahmed** for their valuable guidance, encouragement and help in succesful completion of this project. I am also thankful to my friends like **Tirupati Bala Maddheshiya** and **Amarjeet Yadav** for their help and insights.

I would also like to thank all the teaching and non-teaching staff of the Department of Physics, Jamia Millia Islamia, for their kind cooperation.

Shashikant Kumar

CERTIFICATE

This is to certify that the project work entitled **Fourier Analysis of Non-linear Signal** shows a work done by **Shashikant Kumar**, in partial fulfillment of requirement for the award of the degree of Master of Science in Physics at Jamia Millia Islamia, New Delhi, under my supervision and guidance.

I hereby approve it for submission to the university for the award of Master of Science in Physics. Further, to the best of my knowledge, this report has not been submitted to any other institution for the award of any degree or diploma.

Dr. Anver Aziz,
Department of Physics,
Jamia Millia Islamia New Delhi.

Dr. Syed Rashid Ahmed,
Department of Physics,
Jamia Millia Islamia New Delhi.

ABSTRACT

The Fourier transform is widely used for frequency analysis, but non-linear signals produce additional spectral components such as harmonics and intermodulation products. This work highlights how non-linear operations alter signal spectra and discusses limitations of classical Fourier analysis. Alternative methods, including wavelet and time-frequency techniques, are briefly considered for more accurate characterization of non-linear behavior.

CONTENTS

✓ Introduction

The Fourier transform is one of the most powerful mathematical tools for analyzing signals and physical systems. It provides a way to decompose a time-domain signal into a spectrum of sinusoidal components, making it possible to study the frequency content and behavior of systems in a wide range of scientific and engineering applications. While the classical Fourier transform theory is well-established for linear and time-invariant systems, the analysis of nonlinear signals and nonlinear systems introduces significant challenges. Nonlinear signals do not follow the principle of superposition, meaning their response to a combination of inputs cannot be determined by analyzing each input separately. As a result, their Fourier spectra often contain new frequency components—such as harmonics, sub-harmonics, and intermodulation products—that are absent in the original input signal.

Understanding and interpreting these spectral components is crucial in many modern technologies, including communications, biomedical engineering, seismology, and electronic instrumentation.

A nonlinear signal may arise from intrinsic properties of a physical process, or it may be introduced by nonlinear devices, such as diodes, transistors, power amplifiers, and optical modulators. For example, in electronic communication systems, nonlinearities in high-power amplifiers cause spectral spreading and distortion, leading to interference and reduced signal quality. Similarly, in speech processing, the vocal tract behaves as a nonlinear acoustic system, generating harmonics essential for speech recognition and synthesis. In mechanical systems, nonlinear vibrations produce complex frequency patterns that assist in fault diagnosis and structural health monitoring.

When a nonlinear system is excited by a pure sinusoidal signal, its Fourier transform often reveals harmonic frequencies at integer multiples of the original frequency. This behavior is a hallmark of nonlinear operations such as squaring, clipping, and amplitude modulation. For example, if a signal is passed through a square-law device, the resulting spectrum contains both a direct current (DC) component and a component at twice the input frequency. If multiple signals are present, nonlinear interaction generates intermodulation frequencies (sum and difference components), which are critical when studying multi-tone or broadband signals.

Traditional linear Fourier analysis can describe these frequency components, but it does not always capture the full dynamics of nonlinear systems. Therefore, researchers have developed advanced analytical methods including the Volterra series, Hilbert transform, wavelet transform, Wigner–Ville distribution, and higher-order spectral analysis (bispectrum and trispectrum). These methods reveal additional information such as phase coupling, energy transfer between frequencies, and nonlinear correlations.

The Fourier transform of nonlinear signals has extensive applications in wireless communication, radar systems, power electronics, medical imaging (EEG, ECG), optical fiber communication, audio processing, and machine learning-based signal classification. As modern systems push toward higher performance, faster data rates, and stronger signal integrity, understanding nonlinear spectral behavior has become increasingly important.

In summary, the Fourier transform of nonlinear signals is an essential field of study that extends classical frequency analysis to real-world systems where nonlinear effects cannot be ignored. Its ability to identify harmonic and intermodulation components provides deep insight into complex signal behaviors, enabling advancements in science, industry, and technology.

✓ Fourier Transform Basics

The Fourier Transform is a fundamental mathematical tool used to convert a signal from the time domain into the frequency domain. While a time-domain signal shows how a quantity varies with time, the frequency-domain representation shows the different frequencies present in the signal and their amplitudes. This transformation is extremely useful in signal processing, communication, image analysis, audio engineering, and many other scientific fields.

The continuous-time Fourier Transform of a signal x(t) is defined as:

$$x(\omega) = \int_{-\infty}^{+\infty} x(t)\, e^{-j\omega t} dt$$

equation indicates that the transform decomposes a signal into sinusoidal components of different frequencies. The inverse Fourier Transform reconstructs the original signal from its spectral representation.

Objective: Analyze and reconstruct periodic signals using Fourier series.
Signals Studied:

- Sawtooth Wave
- Square Wave
- Triangle Wave

Tools: Fortran 90 for computation, Gnuplot for visualization.

Fourier series represents a periodic function as a sum of sine and cosine terms:

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

Coefficients are calculated as:

$$a_n = \frac{2}{T} \int_0^T f(t) \cos\left(\frac{2\pi n t}{T}\right) dt, \quad b_n = \frac{2}{T} \int_0^T f(t) \sin\left(\frac{2\pi n t}{T}\right) dt$$

a_0 represents the average (DC component) of the signal:

$$a_0 = \frac{2}{T} \int_0^T f(t) dt$$

By summing up the terms in the Fourier series, the signal can be reconstructed:

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

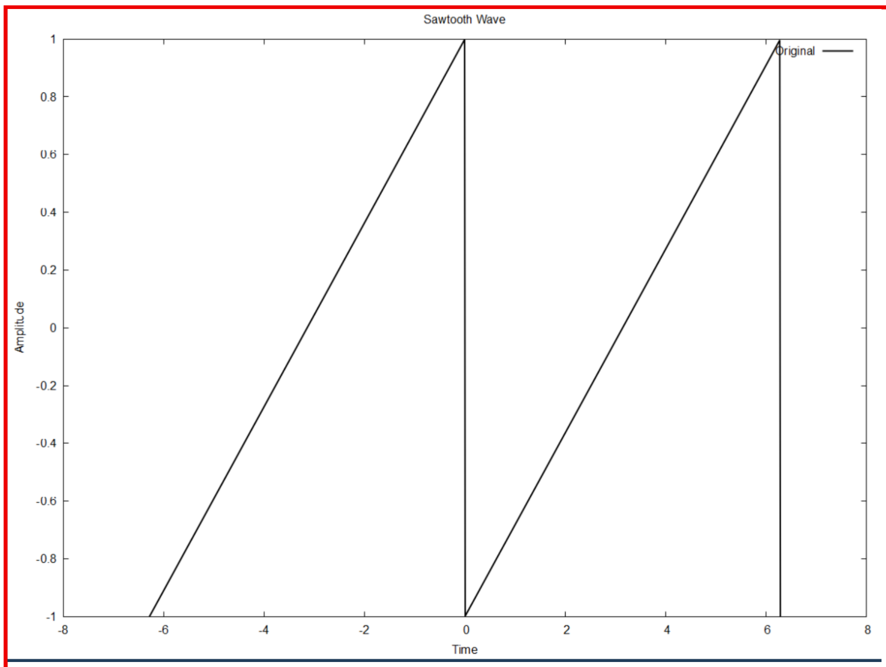
The accuracy of the reconstruction improves with the number of terms (N).

Smooth signals converge faster due to rapid decay of Fourier coefficients.

Discontinuous signals, such as square and sawtooth waves, exhibit slower convergence.

```fortran
program fourier_analysis
    implicit none
    integer, parameter :: dp = selected_real_kind(15, 307)
    integer, parameter :: n_points = 1000
    integer, parameter :: n_terms = 50
    real(dp), parameter :: pi = 3.14159265358979323846_dp
    real(dp), parameter :: period = 2.0_dp * pi
    real(dp), parameter :: A = 1.0_dp

    real(dp), dimension(n_points) :: t_array
    real(dp), dimension(n_points) :: sawtooth_signal, square_signal, triangle_signal
    real(dp) :: a0_sawtooth, a0_square, a0_triangle
    real(dp), dimension(n_terms) :: an_sawtooth, bn_sawtooth
    real(dp), dimension(n_terms) :: an_square, bn_square
    real(dp), dimension(n_terms) :: an_triangle, bn_triangle
    real(dp), dimension(n_points) :: sawtooth_approx, square_approx, triangle_approx
    real(dp) :: dt
    integer :: i, n

    ! Generate time array
    dt = 4.0_dp * pi / (n_points - 1)
    do i = 1, n_points
        t_array(i) = -2.0_dp * pi + (i - 1) * dt
    end do

    ! Generate signals
    do i = 1, n_points
        sawtooth_signal(i) = sawtooth_wave(t_array(i))
        square_signal(i) = square_wave(t_array(i))
        triangle_signal(i) = triangle_wave(t_array(i))
    end do
    ! Compute Fourier coefficients for all signals
    call compute_fourier_coefficients(sawtooth_signal, n_terms, t_array,
a0_sawtooth, an_sawtooth, bn_sawtooth)
    call compute_fourier_coefficients(square_signal, n_terms, t_array, a0_square,
an_square, bn_square)
    call compute_fourier_coefficients(triangle_signal, n_terms, t_array,
a0_triangle, an_triangle, bn_triangle)

    ! Reconstruct signals and write data
    do n = 1, n_terms
        sawtooth_approx = reconstruct_fourier_series(a0_sawtooth, an_sawtooth(1:n),
bn_sawtooth(1:n), n, t_array)
        square_approx = reconstruct_fourier_series(a0_square, an_square(1:n),
bn_square(1:n), n, t_array)
        triangle_approx = reconstruct_fourier_series(a0_triangle, an_triangle(1:n),
bn_triangle(1:n), n, t_array)

        call write_nth_approximation('sawtooth_n' // trim(str(n)) // '.dat',
t_array, sawtooth_approx)
        call write_nth_approximation('square_n' // trim(str(n)) // '.dat', t_array,
square_approx)
        call write_nth_approximation('triangle_n' // trim(str(n)) // '.dat',
t_array, triangle_approx)
    end do

    ! Write original signals
    call write_data('sawtooth_original.dat', t_array, sawtooth_signal)
    call write_data('square_original.dat', t_array, square_signal)
    call write_data('triangle_original.dat', t_array, triangle_signal)

    ! Write Fourier coefficients
    call write_coefficients('sawtooth_coefficients.dat', an_sawtooth, bn_sawtooth)
    call write_coefficients('square_coefficients.dat', an_square, bn_square)
    call write_coefficients('triangle_coefficients.dat', an_triangle, bn_triangle)
```

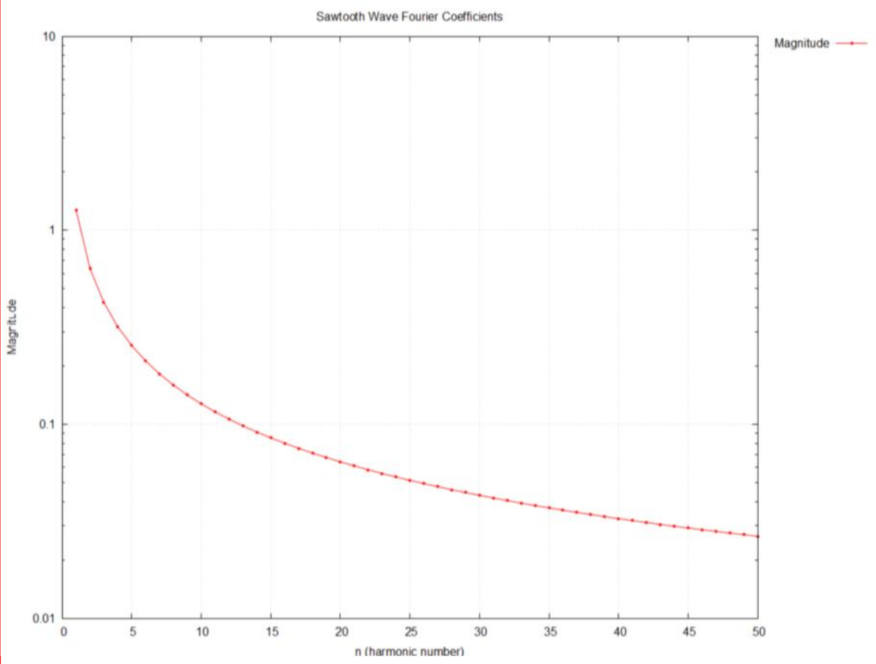
```
contains
    function sawtooth_wave(t) result(y)
        real(dp), intent(in) :: t
        real(dp) :: y, t_normalized
        t_normalized = modulo(t, period)
        y = (2.0_dp * t_normalized / period) - 1.0_dp
    end function sawtooth_wave

    function square_wave(t) result(y)
        real(dp), intent(in) :: t
        real(dp) :: y, t_normalized
        t_normalized = modulo(t, period)
        if (t_normalized < period/2.0_dp) then
            y = 1.0_dp
        else
            y = -1.0_dp
        end if
    end function square_wave

    function triangle_wave(t) result(y)
        real(dp), intent(in) :: t
        real(dp) :: y, t_normalized
        t_normalized = modulo(t, period)
        if (t_normalized < period/2.0_dp) then
            y = 4.0_dp * t_normalized/period - 1.0_dp
        else
            y = 3.0_dp - 4.0_dp * t_normalized/period
        end if
    end function triangle_wave

    function reconstruct_fourier_series(a0, an, bn, n_terms, t) result(y)
        real(dp), intent(in) :: a0
        real(dp), dimension(:), intent(in) :: an, bn, t
        integer, intent(in) :: n_terms
        real(dp), dimension(size(t)) :: y
        integer :: n

        y = a0 / 2.0_dp
        do n = 1, n_terms
            y = y + an(n) * cos(2.0_dp * pi * n * t / period) + &
                bn(n) * sin(2.0_dp * pi * n * t / period)
        end do
    end function reconstruct_fourier_series
    subroutine compute_fourier_coefficients(signal, n_terms, t, a0, an, bn)
        real(dp), dimension(:), intent(in) :: signal, t
        integer, intent(in) :: n_terms
        real(dp), intent(out) :: a0
        real(dp), dimension(:), intent(out) :: an, bn
        integer :: n, i
        real(dp) :: dt

        dt = t(2) - t(1)
        a0 = sum(signal) * dt / period

        do n = 1, n_terms
            an(n) = 0.0_dp
            bn(n) = 0.0_dp
            do i = 1, size(t)
                an(n) = an(n) + signal(i) * cos(2.0_dp * pi * n * t(i) / period) *
dt
                bn(n) = bn(n) + signal(i) * sin(2.0_dp * pi * n * t(i) / period) *
dt
            end do
            an(n) = 2.0_dp * an(n) / period
            bn(n) = 2.0_dp * bn(n) / period
        end do
    end subroutine compute_fourier_coefficients

    subroutine write_nth_approximation(filename, t, approx)
        character(len=*), intent(in) :: filename
        real(dp), dimension(:), intent(in) :: t, approx
        integer :: i, unit

        open(newunit=unit, file=filename, status='replace')
        do i = 1, size(t)
            write(unit, *) t(i), approx(i)
        end do
        close(unit)
    end subroutine write_nth_approximation
    subroutine write_data(filename, t, signal)
        character(len=*), intent(in) :: filename
        real(dp), dimension(:), intent(in) :: t, signal
        integer :: i, unit

        open(newunit=unit, file=filename, status='replace')
        do i = 1, size(t )
            write(unit, *) t(i), signal(i)
        end do
        close(unit)
    end subroutine write_data

    subroutine write_coefficients(filename, an, bn)
        character(len=*), intent(in) :: filename
        real(dp), dimension(:), intent(in) :: an, bn
        integer :: i, unit

        open(newunit=unit, file=filename, status='replace')
        do i = 1, size(an)
            write(unit, *) i, an(i), bn(i)
        end do
        close(unit)
    end subroutine write_coefficients

    function str(n) result(string)
        integer, intent(in) :: n
        character(len=20) :: string
        write(string, '(I0)') n
        string = adjustl(string)
    end function str

end program fourier_analysis
```



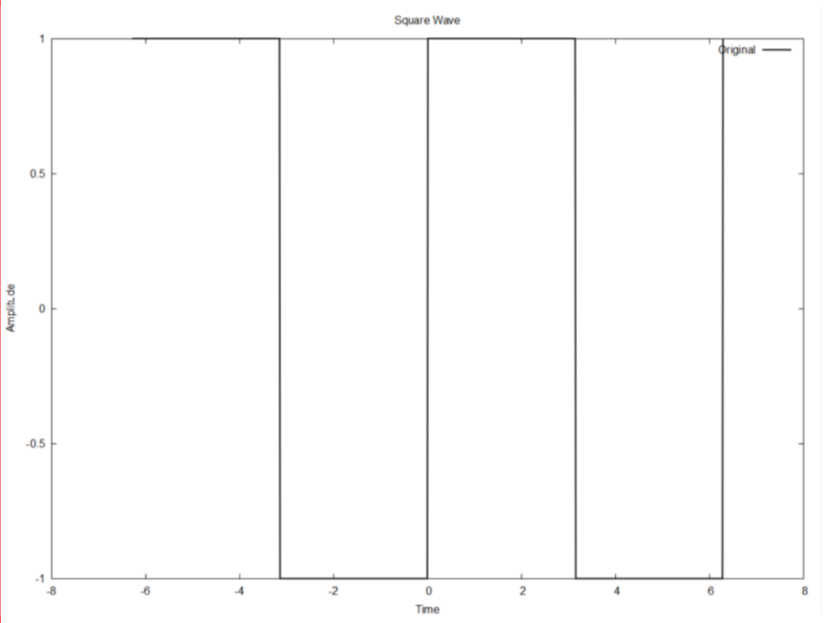
- The sawtooth wave is a periodic non-linear signal with discontinuities at the period end.
- Functional form:

$$f(t) = \frac{2A}{T}\left(t - \frac{T}{2}\right)$$

For $0 \leq t < T$



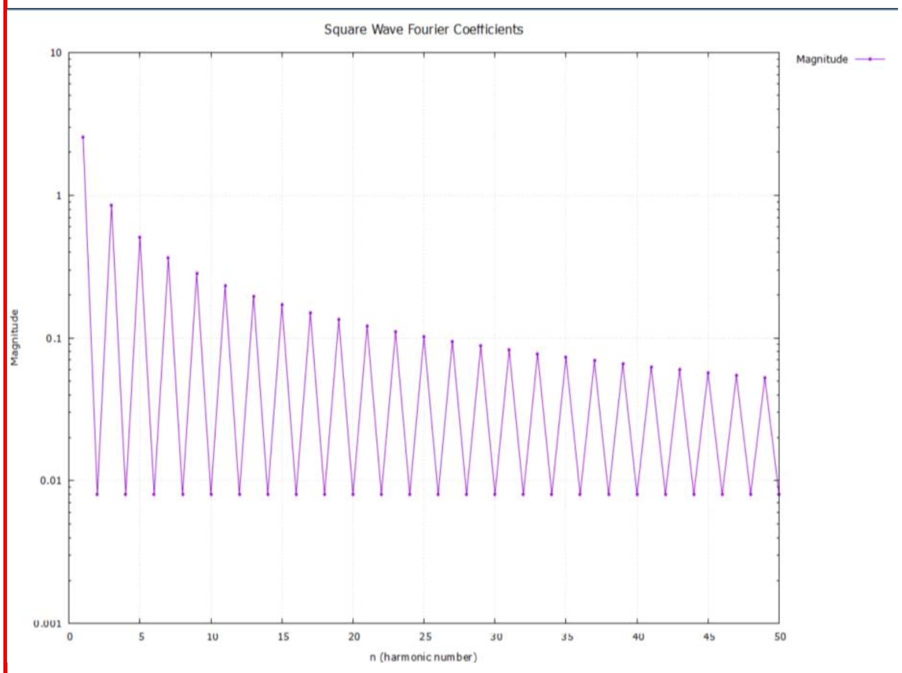
- Coefficients decay linearly, with odd harmonics contributing most to the series.
- The amplitude of each coefficient decreases as 1/n.



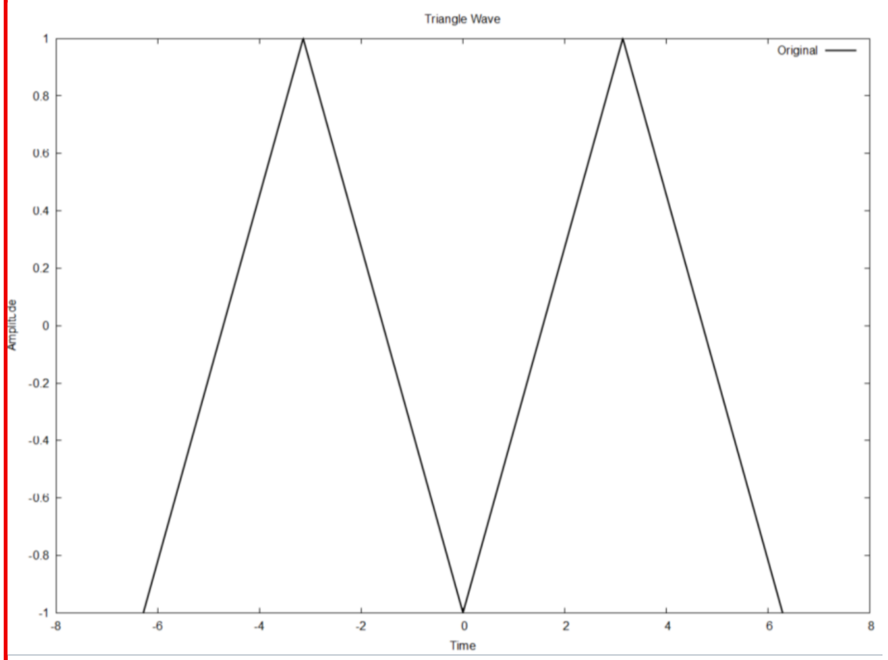
- Square wave alternates between two levels with sharp transitions.
- Functional form:

$$f(t) = \begin{cases} A, & 0 \leq t < \dfrac{T}{2} \\ -A, & \dfrac{T}{2} \leq t < T \end{cases}$$

- Only odd harmonics contribute to the Fourier series.



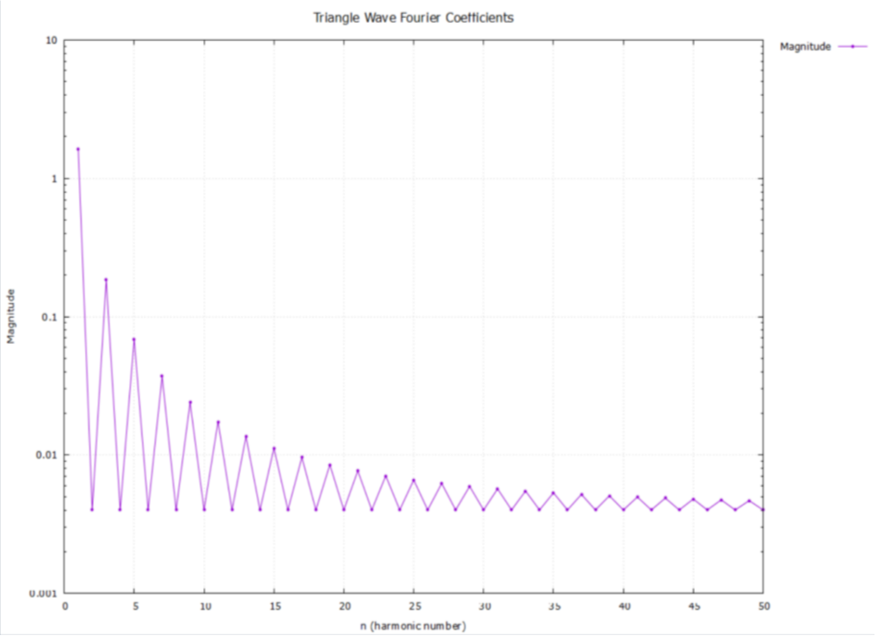
- The square wave has nonzero coefficients only for odd harmonics.
- Coefficients for even harmonics are zero.



- The triangle wave is a periodic, piecewise non-linear signal.
- Functional form:

$$f(t) = \frac{4A}{T^2}\left(T - \left|t - \frac{T}{2}\right|\right)$$

- Fourier series converges faster than for the square wave, but still requires many terms for a good approximation.



- The coefficients decay faster than those of the square wave.
- The amplitude of the coefficients is proportional to $1/n^2$.

✓ Applications

Application of Fourier Transform of Non-Linear Signals

Non-linear signals generate harmonics, intermodulation products, and spectral spreading. The Fourier Transform helps analyze these effects in many real systems. Major applications include:

1. Communication Systems

Power amplifiers, mixers, modulators introduce non-linear distortion. Fourier analysis helps identify intermodulation interference, adjacent channel interference, and bandwidth expansion.

2. Audio & Speech Processing

Clipping, overdrive, and saturation in amplifiers produce harmonics. Fourier Transform is used for noise reduction, audio quality enhancement, and effects design.

3. Power Electronics & Electrical Systems

Converters, inverters, and motor drives generate harmonic currents. Fourier analysis enables power quality monitoring, filter design, and THD calculation.

✓ Conclusion

- ❖ Fourier series effectively reconstructs periodic signals.
- ❖ Different signals (sawtooth, square, triangle) show varied convergence rates.
- ❖ Practical applications in signal processing and audio synthesis.