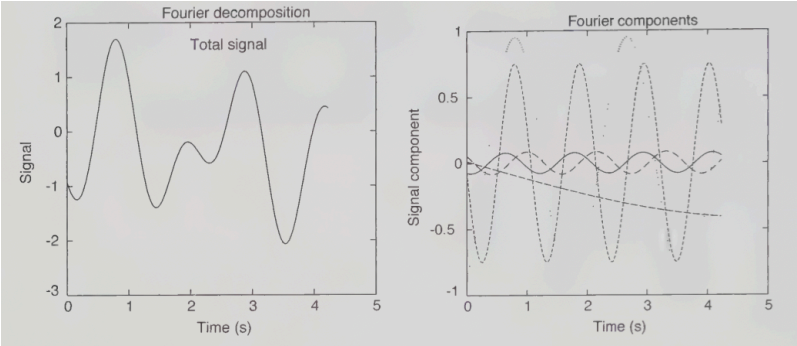
Appendix C
The Fourier Transform

C.1) Theoretical Background

The Fourier transform is a powerful but often under-taught tool in physics.
It allows any signal to be represented as a sum of sine waves.



Left: Hypothetical signal; right: individual sine waves whose sum yields the signal at left.
Even complex signals can be broken down into simpler frequency components.
The signal y(t) exists in the time domain, and its transform Y(f) exists in the frequency domain.

$$y(t) = \sum_{j=1}^{5} y_j \sin(2\pi f_j t + \phi_j) \qquad Y(f) = \int_{-\infty}^{\infty} y(t)\, e^{2\pi i f t}\, dt = \int_{-\infty}^{\infty} y(t)\, e^{i\omega t}\, dt$$

The Fourier transform (forward) converts time-domain data to frequency-domain data.
The inverse transform brings it back to the time domain.
The transform uses complex exponentials, combining sine and cosine.
A signal and its transform contain the same information, just expressed differently.
Fourier analysis is especially useful for analyzing signals like sound.
The transform reveals which frequencies are present in the signal and in what amounts.
If a system is linear (like many physical systems), its total response to a signal is the sum of its responses to each frequency component, this relies on superposition.
Fourier methods apply widely in physics, including sound, waves, quantum mechanics, and heat flow.

C.2) Discrete Fourier Transform

In practice, we usually work with discrete data, not continuous functions.
The Discrete Fourier Transform (DFT) lets us compute the Fourier transform from evenly spaced data points.

$$y_m = \frac{1}{N} \sum_{n=0}^{N-1} Y_n\, e^{-2\pi i m n/N} \qquad \text{(C.5)}$$

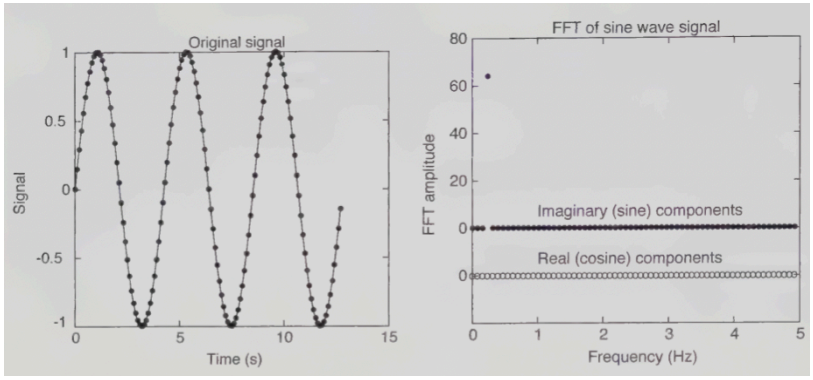
$$Y_n = \sum_{m=0}^{N-1} y_m\, e^{2\pi i m n/N}\,, \qquad \text{(C.6)}$$

The formulas (C.5) and (C.6) define the forward and inverse DFTs.
Performing a forward DFT followed by an inverse DFT returns the original data.
The DFT operates in two equivalent domains: time (y_m) and frequency (Y_n).

The time step Δt doesn't explicitly appear in the DFT formulas because of equal spacing.
Physical signals are usually real-valued, but their DFTs are often complex.
The Nyquist frequency, fNyquist=1/(2Δt), is the highest frequency that can be uniquely captured.
The sampling theorem states that sampling a signal at least twice per cycle is enough to capture that frequency component.
Frequencies above the Nyquist limit cannot be accurately recovered and cause errors (aliasing).

C.3) Fast Fourier Transform (FFT)

Direct computation of the Discrete Fourier Transform (DFT) is slow, it takes on the order of N^2 operations.
The FFT algorithm reduces this to about NlogN, making large transforms feasible.
FFT works by exploiting repetition and symmetry in the exponential terms of the DFT to reuse computations.
It splits the original sum into parts based on even and odd indices, then recursively applies this idea.
The method relies heavily on binary representations of indices and grouping terms based on bit patterns.
Each recursive "level" of the algorithm requires about N operations, and there are log_2N levels.
In total, FFT needs about Nlog_2N steps, a huge improvement over N^2, especially for large N.
FFT is widely used in real applications, it enables fast signal processing in fields like X-ray tomography and audio analysis.
In practice, FFTs are often computed using prebuilt library routines, not written from scratch.
FFT is typically done in place, meaning the input array is overwritten with the output to save memory.
A bit-reversal step is used to reorder outputs correctly based on binary index positions.
Understanding FFT highlights how smart algorithm design can make otherwise impractical tasks efficient.
A schematic illustration of an example of this algorithm:



Left: Pure sine wave signal, y(t) = sin(2πft). These 128 points are the values of the signal that are Fourier analyzed. Right: FFT of this signal. The real (cosine) and imaginary (sine) parts of the transform are shown separately. We have plotted them as discrete points to emphasize that the number of Fourier amplitudes yielded by FFT is equal to the number of original data points. Hence we have 64 cosine and 64 sine components. Only one of the points in the transform is nonzero.

C.4) Examples: Sampling interval and number of data points

Two key parameters in a DFT/FFT are the sampling interval (Δt) and the number of samples (N).
Sampling interval determines the highest frequency (Nyquist frequency) that can be captured.
Number of samples affects the resolution and detail of the frequency domain representation.
Choosing a total sampling time that captures whole periods of the signal simplifies the FFT result.
A pure sine wave sampled cleanly over full periods yields an FFT with a single nonzero component.
Adding a phase shift to the sine wave introduces both sine and cosine components in the FFT.
Arbitrary phase in signals requires both sine and cosine terms for accurate representation.
If signal frequencies don't match discrete FFT frequencies, the FFT spreads energy across multiple nearby frequencies.
Despite this spreading, the FFT still perfectly reconstructs the original discrete data.

C.5) Examples: Aliasing

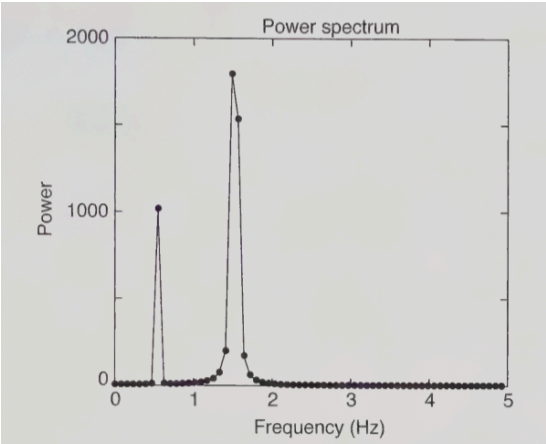
The sampling theorem states that frequencies above the Nyquist frequency ($f_{Nyquist}=1/(2\Delta t)$) can't be accurately captured.
If a signal contains frequencies higher than Nyquist, the FFT misrepresents them as lower frequencies, this is called aliasing.
Aliasing happens because high-frequency components "fold back" into the frequency range below the Nyquist limit.
In the example, a 4 Hz signal sampled with $\Delta t=0.2$ s (Nyquist = 2.5 Hz) gets aliased and shows up as a 1 Hz component in the FFT.
A key point: both the original (true) and aliased frequencies produce the same sampled points, so the FFT can't distinguish between them without more data.
To avoid aliasing, ensure your sampling rate is high enough to cover the highest frequency of interest.
In experiments, a low-pass filter can remove frequencies above Nyquist before sampling.
In simulations, use a small enough time step so all meaningful frequencies lie below Nyquist.
If you're interested in low-frequency detail, you may need a large number of time-domain samples to get good resolution.
Unwanted components can skew the transform and should be subtracted before applying the FFT.

C.6) Power Spectrum

The power spectrum shows the strength of each frequency component in a signal, ignoring phase.
It's useful when you're only interested in which frequencies are present and how strong they are, not their timing or phase.
The power spectrum is the Fourier transform of the autocorrelation function of the signal.
The autocorrelation function measures how similar the signal is to itself at different time shifts

(lags).
Peaks in the autocorrelation correspond to repeating patterns, which show up as peaks in the power spectrum at corresponding frequencies.
Practically, the power at each frequency f is proportional to |Y(f)|^2, where Y(f) is the Fourier transform.

For discrete data, power at frequency f_j is computed as: $$P_j = Y_j(\text{real})^2 + Y_j(\text{imaginary})^2$$
The power spectrum discards phase information, so it cannot be used to reconstruct the original signal.
It's commonly used in physics and engineering (analyzing signals across resistors, sound, or vibration data).



The solid symbols are the calculated values of the power.