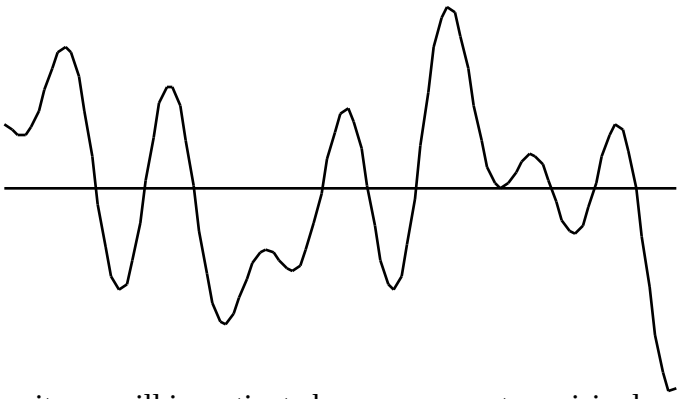
Unit X
Fourier Transforms

Unit X
Fourier Transforms

In elementary physics, we are used to seeing oscillatory and wave phenomena described by simple harmonic functions, sines and cosines. So, when we are confronted with data such as the following, we are often baffled, and expect that this must come from some sort of chaotic system, since it does not appear to be very simple!



In this unit, we will investigate how we can get surpisingly complicated wave patterns from simple superpositions of sines and cosines, and learn how to analyze those patterns numerically to obtain their harmonic (sine and cosine) components. This sort of analysis is extremely useful for systems that are at least nearly governed by linear, harmonic differential equations (e.g. the harmonic oscillator equation, or the wave equation), and of more limited utility in non-linear systems. Nevertheless, as good physicists, we must learn well the problems that we can actually solve, and then only reluctantly move on to those problems we can only partly solve, or only gain rough approximate solutions.

Guidebook Entry X.1: Fourier Synthesis

Let's create a spreadsheet that allows us to add several different sinusoidal functions together, which is called Fourier Synthesis. Let's imagine that we are talking about oscillations in time (although we could just as well consider oscillations in space). Each component function is of the character

$$f(t) = A\sin(\omega t + \varphi).$$

You will create columns for t, three component functions, and the sum which we can call F, but before doing that, make space in the top rows to store the values of the constants of the component functions, that is, A, ω, and φ. Then label and fill the t column from 0 to 10 in steps of 0.1. Fill the top entries in the next three columns with functions that are of the form above, and finally, create the sum column. Make a graph of the resultant sum function.

Choose A = 1 for the first component function, with $\omega = 1$ and $\varphi = 0$. Let A = 0 for each of the other component functions. Verify that the graph is what you expect.

What value of φ should give a resultant graph that is a pure cosine function?

Try this--does it work?

Show that you can produce a pure sine wave of frequency $\omega = 1$ and arbitrary phase by adding a sine wave ($\varphi = 0$) to a cosine wave ($\varphi = \pi/2$) with differing amplitudes. In particular, try the combinations:

Sine amplitude	Cosine amplitude	Phase shift?
1	0	
0.4	0.8	
0.7	0.7	
0.8	0.4	
0	1	

Can you adjust the amplitudes to get the phase shift past 90º? Explain.

The amplitude of the resulting wave should be equal to the square root of the sum of the squares of the component amplitudes, if you are adding a pure sine to a pure cosine of the same frequency. Can you verify this for a couple of examples?

Now try adding the following component waves:

	wave 1	wave 2	wave 3
Amplitude	2	2	2
Frequency	1	3	4.6
Phase	1	0	3

Compare the resulting graph to the mysterious one on the first page. Do they look similar? Save this data; you will use it in the next activity.

In the last exercise, you found that you could produce seemingly very erratic wave patterns through the superposition of simple waves. In the next exercise, we will learn how to start with an erratic looking wave pattern, and break it apart into its component waves.

Guidebook Entry X.2: Fourier Analysis

The decomposition of a wave into its component pieces depends on the fact that one sine wave integrated against another sine wave over a suitably large interval is zero, unless the two frequencies are the same:

$$\int_0^{2\pi} \sin(2n\pi t)\sin(2m\pi t)dt = \pi\delta_{mn}$$

[The expression of the frequencies this was assures an integral number of periods over the integration range, and the δ is zero unless m=n, when it becomes 1.] If m and n are different, the product of two sine waves is a rapidly oscillating function that averages to zero over a large interval. By related argument, any cosine function integrated against a sine function will yield zero. If we are going to do this by computer and obtain real numbers, then we will have to evaluate this integral numerically. Typically this is done simply by evaluating the Riemann sum. Let us then consider an example like the one above, where we have a wave over an interval from 0 to 10 in steps of 0.1. This can be analyzed in terms of frequencies from that corresponding to the base period ($\omega = 2\pi/10$) up to a frequency at which the function can go from maximum to minimum from one point to the next ($\omega = 100\pi/10 = 10\ \pi$). We will gain coefficients of both sine and cosine terms at each frequency which is a multiple of the base frequency up to the

maximum frequency. How many different coefficients then is this? How does this compare to the number of original function values?

Each cosine coefficient depends on the original function f as follows:

$$A_{\cos} = 2\sum_i \left[f(t_i)\cos(\omega t_i) \right] / N$$

where N is the number of points, or 100 in this case. There is an analogous expression for each of the sine coefficients.

Now, set up a sheet to calculate this. Create a column of t values, then a column for corresponding f values (copy these from your synthesized wave from above). Along the top, create a row of ω's, having each ω appear twice, once for the cosine term and once for the sine term. Then, in a column below each ω, introduce the element in the Riemann sum, that is, f(t)*sin(ωt) for the sine terms. At the bottom of each column, insert an expression that takes the sum of the column elements, and divides by N/2 to normalize the expression. Fill all of these across and down, making sure to keep appropriate references absolute. There also is a trick that allows you to fill alternating sine and cosine expressions if you choose to have them adjacent--ask your instructor if you need help.

Apply this analysis to your original data. Does it extract the correct frequencies? What are the discrepancies? Discuss them with your instructor.

For many applications (e.g. sound analysis), we don't care very much about the phase of the component functions (whether they are sine or cosine or both), but rather just the strength at a particular frequency. In this case, we take a power spectrum. The power is simply the square of the amplitude at the particular frequency.

Guidebook Entry X.3: Fourier Power Spectrum

Recall that the amplitude of the sum of a sine and cosine function is given by a relationship reminiscent of Pythagoras:

$$A_{total}^2 = A_{sine}^2 + A_{cosine}^2.$$

Create a new row in your spreadsheet from above just below the coefficients, and produce a set of power values. Graph this versus ω. Does your result make sense in terms of your initial amplitudes? Explain.

Do you have enough information just in the amplitudes to recreate the original function? Why or why not?

If one uses complex versions of the trigonometric functions, the form of the Fourier transform becomes much more symmetric. That is, virtually the same algorithm can be used to go from a function of time to amplitudes at each frequency as going back from amplitudes to the function. This makes some sense, since each process involves multiplying by sine and cosine functions and adding together. In this case, the distinction between Fourier synthesis and Fourier analysis is lost, and the process is generally just terms a Fourier transform.

The computation of a Fourier transform becomes rather computer intensive if the set of numbers involved gets very large, since the number of calculations goes like N^2. However, since sines and cosines in complex notation can be represented by the exponential function with an imaginary argument. This then allows us to take the multiplication of trigonometric functions and turn that into the addition of the arguments. This allowed the development of a clever calculational algorithm called the Fast Fourier Transform. The exact details of this are not important, but virually all computer Fourier analysis is done these days by Fast Fourier Transform, or FFT. Many standard books detail the algorithm (such as *Numerical Recipes*, or DeVries *Computational Physics*). We will just let Mathematica do this job for us.

It is important for you to know that the FFT is simply and algorithm for calculating the Fourier transform. It is no different than the transform you have done yourself in terms of the numbers that result. The only difference is that it is computationally

faster. It is so much faster that its common introduction was a big deal in businesses that used Fourier transforms (e.g. a variety of spectroscopy techniques, such as NMR). As a result, many people who use FFT's will swear that it produces somehow "better" numbers, and that a conventional discrete transform such as the one you did in Excel is inferior in its result, which is of course nonsense. Mathematica does not make a big deal of the fact that their routine is in fact an FFT; it is actually a bit difficult to deduce this from the manual.

Since the FFT depends on expressing functions in terms of complex exponentials the results come out as complex numbers, where the resulting amplitudes have a real part that is the cosine coeffient and the imaginary (I) part is the sine coefficient.

Guidebook Entry X.4: Mathematica and the FFT

To take a Fourier transform, we must first generate a data table. To do this, use the Mathematica command

data = Table[{x, f(x)},{x, 0.,10., 0.1}]

where f is the initial waveform we are about to transform. In this example, I have chosen the same range and spacing as our previous example. Create this table using our original function that is the sum of the three waves on page 3.

Now to actually take the Fourier transform, we simply invoke the command

Fourier[data]

and the result comes out as a table. Do this transform, and see if your results agree with your results from before.