Integral transforms

Professor Richard Harvey FBCS

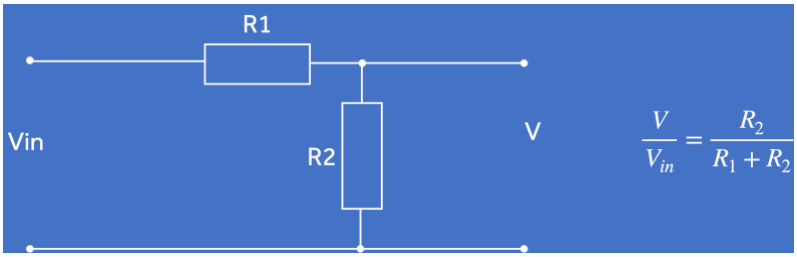
12th April 2022

Merely mummering the words “Fourier transform” are enough to send shivers through the heart of an average engineering student and in most computer science courses the words never cross the lecture hall. Even experts regard them as a somewhat recherché topic. Yet integral transforms are a vital part of modern society. We have already encountered them several times in lectures in this series: digital image and audio compression uses them; mobile telephony uses them and not mentioned so far, digital audio broadcast, digital video and many others.

The difficulty is that they are regarded as a very difficult topic – only the brightest students master integral transforms. The very word is off-putting – clearly, they contain an integral, yuck, and a difficult integral, double-yuck. So, this is one of those lectures that might be a triumph of clarity or a disaster of obfuscation and needless maths. Shall we begin?

It is an old engineering joke that if something moves, and it should not, then use gaffer tape; if something does not move, and it should, use a hammer. Indeed, engineers are sometimes portrayed in the movies as wandering around mechanised environments tapping things with a hammer. One might say that predicting what will happen to something if I hit it is indeed an antique problem. But in classical mathematics predicting the motion of a “dynamical system” is not so easy — we have to solve the differential equations associated the system given some *initial conditions* (the starting point of the system) and the *forcing function* (that might be a hammer blow but, in the case of a car suspension, then it would be the shape of the lumps and bumps in the road). Many hours of undergraduate time are spent solving such problems. I’ll show you one from electrical engineering.

This picture is known as a potential divider and, if the boxes contained resistors, then you might remember from your school days that the output voltage is a fraction of the input voltage where that fraction is determined by the size of the resistance, R_2 to the total resistance $R_1 + R_2$. Now, what if the box on the right-hand side contained not a resistor but something more complicated like a capacitor[1]. In that case our circuit would look like this.



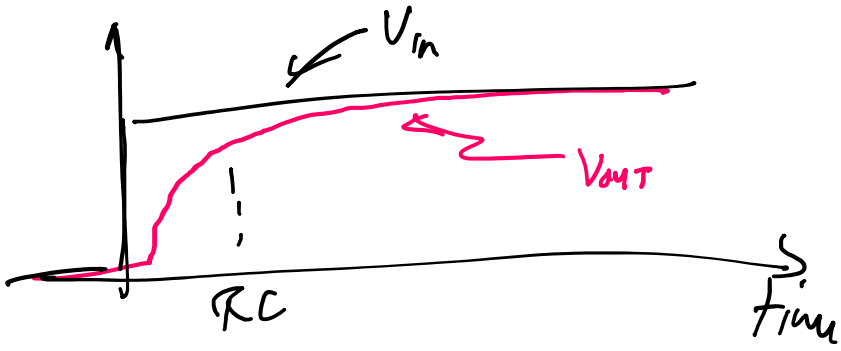
A resistor is a device where the current through it is proportional to the voltage across it, we write $V=IR$ (Ohm’s law) but a capacitor has a more complicated arrangement in which I = C dV/dt. If you unfamiliar with

[1] Don’t panic (yet) – a capacitor is a standard electrical component that can be thought of two conducting plates held very close together but not touching. No direct current can pass between the plates but if the voltage changes then electromagnetic waves couple the plates and alternating current can flow.

the notation dV/dt do not worry – it's the rate of change of V (measured in Volts per second). Looking at our little circuit above the voltage we argue that the current flowing through the resistor and the capacitor are identical (no current flows out of the circuit) so we can put these together to give a first-order ordinary differential equation:

$$RC\frac{dv(t)}{dt} + v(t) = v_{in}(t)$$

It looks a bit daunting, and it is hardly surprising that undergraduates are not very keen on such things. However, Professors are very keen on them and will often run through a gamut of possible inputs so that they can illustrate, not only how the circuit works, but also to demonstrate how clever they are. One popular input is the Heaviside unit step, named after Oliver Heaviside, and if we use that we get something that looks like this:



But solving these things is a bit of a faff. If we were interested in the response to a sin or cosine wave, we might set $V_{in} = A\cos(\omega t)$. I can tell you that this is going to get a bit messy unless we make some swift realisations. The first one is called the LTI assumption – the circuit is Linear Time Invariant. Time Invariant is easy - it means the circuit does not change over time. Linear is very useful – a system is linear if the output to the sum of two waveforms can be written as the sum of responses to the two components. One way of writing the output of an LTI system is to write

$$V_{out} = f(V_{in})$$

which reads as the output voltage V_{out} is some function $f()$ of the input V_{in}. How does this help? Well let's imagine that the input signal Vin can itself be written as the weighted sum of two simpler waveforms. Maybe we have:

$$V_{in}(t) = ax(t) + by(t)$$

In which case the output can also be written as weighted sum of the outputs due to x(t) and y(t):

$$V_{out}(t) = af\big(x(t)\big) + bf\big(y(t)\big)$$

This is a massive convenience since it might mean we do not have to solve a new differential equation for every single signal that comes into our system. Maybe, if we chose some of those component signals, what I have called $x(t)$ or $y(t)$, very carefully then we could use them as building blocks for more complex cases? But what signal?

The RC circuit I considered earlier had responses that looked like $e^{-t/RC}$ or $1 - e^{-t/RC}$ so what is we tried a component signal of the form e^{st}? Here s is just some variable yet to be defined. I've made it positive e^{st} rather than e^{-st} just to stop us having minus signs floating around the place. So, let's assume in our RC circuit above that

$$v(t) = ve^{st}, \qquad v_{in}(t) = v_{in}e^{st}$$

in which case the derivative[2] is

[2] Don't panic if you cannot remember, or do not know how to compute derivatives, as the gradient of an exponential is also an exponential multiplied by the exponent.

$$\frac{dv(t)}{dt} = sve^{st}$$

How convenient, now each component of,

$$RC\frac{dv(t)}{dt} + v(t) = v_{in}(t)$$

Has the form of something multiplied by e^{st}. Thus

$$svRCe^{st} + ve^{st} = v_{in}e^{st}$$

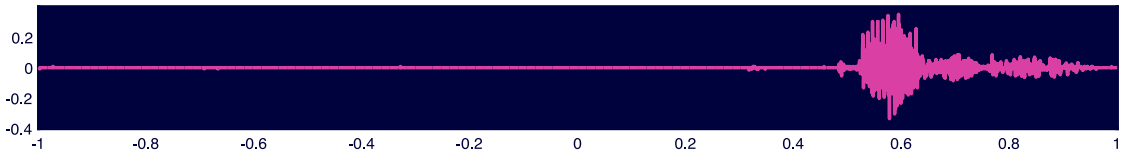
or with a bit of algebraic jiggling about

$$\frac{v}{v_{in}} = \frac{Z_C}{R + Z_C}, Z_C = 1/sC$$

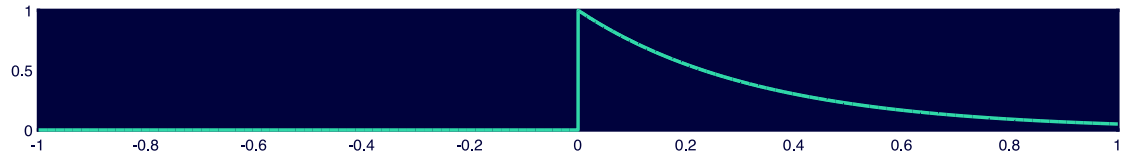
Which is exactly the form of equation I gave earlier when we were dealing with two resistors. The resistor R_2 has become something called Z_C but suddenly, instead of faffing about solving differential equations, I'm doing circuit analysis with these Z-thingies[3].

So far, so nice, but this only seems to work if my input voltage is either of the form of e^{st} or, e^{-st}, or, remembering superposition, some weighted combinations of waveforms like e^{st} or e^{-st}.

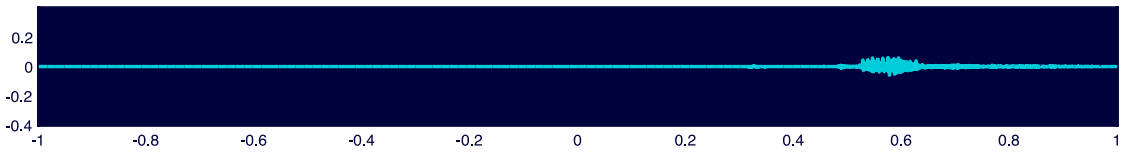
Well one way of measuring how similar two things are is to multiply them and average. Let's try that. Here is a waveform of me saying "Gresham"



and we can multiply it by our "kernel" e^{-st},



to give:



And if we average this signal then the result is a measure of how similar the "Gresham" sound is to our exponential.

Mathematicians call that average an integral and it has funny sign, like an elongated S[4],

$$F(s) = \int_0^\infty f(t)e^{-st}\,dt$$

It looks a bit daunting but let me write it more simply

$$F(s) = \mathcal{L}\{f(t)\}$$

Put simply, $F(s)$ is a measure of how similar your waveform, $f(t)$ is to e^{-st} and it's called the forward Laplace transform. It has an inverse

$$f(t) = \mathcal{L}^{-1}\{F(s)\}$$

[3] *Z* is what electronic engineers called impedance. The impedance of a capacitor is $1/sC$ and the impedance of an inductor is sL.

[4] *S* for a sum and the *dt* means a continuous sum with respect to *t*, time. In this case the sum runs from zero to infinity.

f(*t*) and *F*(*s*) come as a pair and are known as a Laplace transform pair. They are completely equivalent – you give me *F*(*s*) and I can compute *f*(*t*) or vice versa. We write

$$f(t) \leftrightarrow F(s)$$

Suddenly we free of the tyranny of differential equations. We can take any waveform, convert it into its Laplace transform, use simple tools for analysing systems and, if we want, invert the output. I have skipped some rather nasty bits – notability the computation of the inverse requires something quite interesting called a complex contour interval but, in practice this nastiness is often avoided by tables of pre-computed Laplace transforms. The engineer looks up the Laplace transform of the signal and converts the circuit or system into the Laplace domain. Computing the output now involves just simple algebra and no nasty differential equations. If needed the time-domain output signal can be computed from the inverse but many engineers work directly in the Laplace domain.

Despite being named after Pierre Laplace, the French mathematician, it seems as though its use was pioneered by the Norwegian Niels Abel. Laplace did also use transforms, but he was interested in, not differential equations but difference equations. Difference equations are the norm in computers using digital filters – instead of having time signals *f*(*t*) we tend to have sequences *f*(*n*) and instead of differentials we have differences. The equivalent transform is known as the *z*-transform.

$$f(n) \leftrightarrow F\{z\}$$

Now, I think it is fair to say that the Laplace transform, and z-transform are highly recherché devices that are extensively used by circuit designers, digital signal processing designers and control system experts. The latter is particularly important – the reason that your local chemical plant or nuclear plant does not go bang is that control engineers have very carefully converted the system into the s-domain or z-domain and made sure that the plant has stability[5].

So far, we have avoided mentioning the most famous of the transforms: the Fourier transform.

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

It looks a bit like the Laplace transform – there is an integral which is measuring how alike our waveform, *f*(*t*) is to something[6] but now that something is not e^{-st} but something slightly different $e^{-i\omega t}$. The Greek letter omega, ω, is one of the standard mathematical symbols for frequency but most people use the letter, f, measured in Hertz or cycles per second[7]. The Fourier transform is a particular tool of engineers who analyse signals or systems, and there are two bits of additional maths which help us handle some two problematic issues. The first issue is that, if the signal we are analysing is infinitely long then there is a possibility that the integral can blow up. We handle that with a lovely bit of maths known as generalised functions – we allow the answer to be infinite but only in an infinitely small interval – which are drawn as little spikes with a height that represents their area. Those arrows represent a spike of infinite height and zero width – sounds a bit weird but it works fine. Such spikes are usually named Dirac Delta functions after the great physicist, Paul Dirac – a Bristolian trained as an engineer but later turned theoretical physicist.

So, what is the Fourier transform of a cosine wave of frequency f_0? It is two spikes – one at a frequency f_0 and another at frequency $-f_0$. There is often much scratching of heads about the meaning of negative frequency among students but, once we have grown-up, we realise that the negative part of the spectrum is not needed for most circumstances, and we often plot only one-sided Fourier transforms.

The second issue is randomness - most signals have some randomness. Does the Fourier transform help in such circumstances? Yes – randomness can be measured in the time domain via a similarity measure known as *correlation* – if you measure a signal against itself you get *auto-correlation* usually designated $R(\tau)$ The Fourier transform of the autocorrelation is known as the Power Spectral Density (PSD). When people talk loosely about a spectrum they are often talking about the PSD.

---

[5] Deciding if a system is stable turns out to be very simple in the Laplace and/or z-domains. It depends entirely on the locations of where the denominator of the transfer function are zero. These locations are called *poles* and, for stability they need to be located in a region known as the "left-hand half-plane". And hence have led to series of obscure engineering jokes involving Polish people on aircraft.

[6] That something is often called the *kernel* in transform parlance.

[7] ω signifies *angular frequency* and $\omega=2\pi f$.

Symmetries abound in Fourier transform theory. When we are dealing with real signals, it is usually the case that the signal is zero before we start the experiment – thus outputs happen after the inputs. We call this causality which means that half of the Fourier transform integral is zero. One solution is to define the transform over a half interval – the unilateral transform but engineers don't like remembering a lot of things so we tend to stick with the bilateral transform but recognise that causal signals will have FTs with redundant symmetry – which is why people often sketch only half the spectrum.

In a computer system, signals are not continuous, they come in samples. And they are not infinitely long, they have a start and an end. The transform of a set of samples is known as the Discrete Fourier Transform or DFT and, in reality, when someone shows you a spectrum or a spectrogram, they will have used the Discrete Fourier Transform:

$$F(m) = \sum_{n=0}^{N-1} f(n) e^{\frac{-2\pi i n m}{N}}$$

It looks a bit daunting, but it's just the discrete version of the integral form. Again, we are multiplying our sequence *f*(*n*) by a kernel function, which is this case is $e^{-2\pi imn/N}$. Then we average it -- that's the summation. Obviously, we cannot do infinite summations in a computer so instead of running to infinity the sum stops at a number we'll call *N*. From a computational point of view, the good thing is that the exponential kernel repeats itself and this repetition means one can create fast algorithms known as Fast Fourier Transforms or FFTs. The name FFT is often used interchangeably with the DFT which is understandable – there's no point implementing a DFT slowly when Fast algorithms exist for almost all useful situations. On the slides I show some examples of a computed FFT. Note that while the Fourier transform is a useful theoretical concept the reality is that when we are computing the FFT we have sampled data and only a finite amount of it – these imperfections mean that the transform of a cosine wave is not a couple of perfect spikes but there is 'leakage' into adjacent bins[8].

FFT algorithms are used extensively in MPEG audio. We take a block of audio data, compute the FFT and look for features which the human ear might not hear due to "masking". One example of masking is that loud tones conceal the presence of others. This offers the potential to delete some acoustic information: we can compute the FFT; look for distinct tones; and delete any nearby tones on the grounds that humans cannot hear them. This is happening thousands of times per second when you record your voice, record a video or replay some audio on any one of your numerous electronic devices.

Another approach, which is favoured in image processing, is to replace the complex exponential in the DFT with a cosine – this is the Discrete Cosine Transform or DCT. It is often applied in two dimensions so now we have a two-dimensional quantity. We can get quite a good approximation to an image by completely discarding the high-end coefficients as we saw in a previous lecture on compression. This is exactly what the world's favourite image compression standard, JPEG does [1]. It dices images into 8-by-8 pixel sub-images, applies the DCT, discards small value transforms and sends only the large ones. At the received end we reconstruct the blocks via an inverse transform.

One interesting integral transform is the Radon transform. If you have ever had a CT-scan, then you may have wondered what is going on. That ring that is around your body contains sources of energy and receivers – we can measure the integral of your body in a straight line from the transmitted to the receiver. But how to reconstruct the image? Step forward the Radon transform. And it turns out that one quick way to compute the Radon transform is via the FFT.

If your head is spinning at this point then humorously I might point out that there is also an integral transform for problems with circular symmetry – the Hankel transform which uses Bessel functions but, more seriously, I should pause before we overdose on transforms and point out that the general idea behind an integral transform is to take a signal in a domain which is a bit awkward – often time or space and convert into a new domain (f, z, s etc) in which the maths is easier. In this sense transforms are quite a venerable mathematical technique. Sometimes the new domain becomes the *de facto* domain for analysis – everyone is quite familiar with an audio spectrum in which high frequencies (high pitched sounds) are on the right of the graph and low

[8] There is a whole sub-genre of signal processing that discusses this. One solution is to 'window' the data so that it fits perfectly into the N samples we have available. This is the norm when dealing with real-time data. Another approach, which is relevant when you are using the FFT as a measurement device, is to build better approximations to power spectral density.

frequencies (low pitched sounds) are on the left. Not many people are aware that they are looking at an estimate of the power spectral density which is the Fourier transform of the autocorrelation function – their intuitive understanding is enough.

But transforms also appear as algorithms that are used, not just for analysis, but for signal manipulation and information control. In this category the dominant algorithm is the FFT which is completely ubiquitous in modern IT systems. And in Gresham lectures FFTs have arisen many times: audio and speech processing relies on the Fourier transform [2] [3], image processing uses the DCT [4], control theory uses the Laplace transform, digital filters use the z-transform, digital TV and cellular phones use OFDM coding which uses the inverse FT. Transforms are absolutely everywhere. If you are watching this online then my voice is coming to you via an integral transform system, the video is compressed via an integral transform, the signal is further packed into the small amount of bandwidth using a transform system and that's without me pointing out that there is now considerable evidence that your brain does something like an integral transform when it analyses visual or audio signals.

This lecture ends with a minor pedagogical puzzle – esteemed Computer Scientists have declared that the FFT is one of, if not the, most important algorithms of modern times. Modern computers will be computing millions and millions of FFTs everyday – a substantial part of the substantial carbon emissions of IT are due to the FFT. There are great number of efficient algorithms for computing the FFT [5]. But the FFT is not part of the standard Computer Science syllabus. If you are one of my students who will learn it, provided you choose the right modules, but its not guaranteed. For some, this is a triumph of abstraction: our libraries and operating systems are now so good that programmers do need to be bothered with the tedious minutiae of the DFT algorithm but for others it is another example of the danger of modern systems – how can we be sure that they are doing what we think they are doing? To answer that question, I think it would be helpful to look at how a program that you are I write interacts with the hardware. That 'glue" between the program and the hardware is called the Operating System and it is the topic of the next lecture.

References and Further

[1] G. Hudson, A. Léger, B. Niss and I. Sebestyén, "JPEG at 25: Still Going Strong," *IEEE Multimedia,* vol. 24, no. 2, pp. 96-103, 2017.

[2] R. Harvey, ""It from BIT: the science of information"," 23 October 2018. [Online]. Available: https://www.gresham.ac.uk/lectures-and-events/it-from-bit-science-of-information. [Accessed 19 Nov 2021].

[3] R. Havey, "Speech processing: how to wreck a nice peach," Gresham College, 27 Nov 2018. [Online]. Available: https://www.gresham.ac.uk/lectures-and-events/speech-processing. [Accessed 5 April 2022].

[4] R. Harvey, "Compression," Gresham College, 23 November 2021. [Online]. Available: https://www.gresham.ac.uk/lectures-and-events/compression. [Accessed 5 April 2022].

[5] R. Blahut, Fast algorithms for signal processing, Cambridge: Cambridge University Press, 2010.