Informal Introduction to Signal Processing and the Frequency Domain

Chris Brown

July 2010, April 2011

Contents

1 Prolog

1.1 Bad Vibrations

Your dad's been arrested on trumped-up drug and kiddie-porn charges by the Chicago police. He was about to blow the whistle on Boeing about dangerous vibrations in the Dreamliner's jet engine. You've got this datastick you found on the floor before the place was raided...maybe it has something you can use to get his message out to the world and save him by the time Fall tuition's due.

You seem to have .1 seconds of vibration data. Dad's told you the danger lies in resonances that cause peaks at 150 and 350 cycles per second. The notation says "unshielded sensor cables give 60 cycle hum. Noise pretty bad, but danger evident." You plot the data and get Fig. 1.

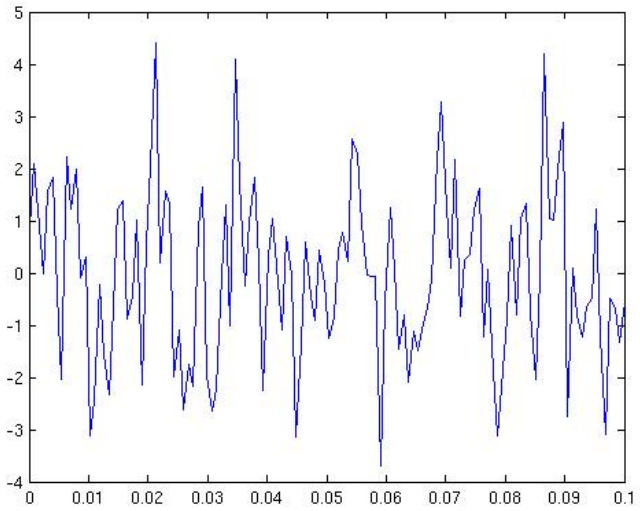


Figure 1: **A Matlab plot of vibration data.**

Yikes. Who will *that* mess convince? Can you see evidence of 150 or 350 Hz vibrations? *Or* 60? But wait! Silently blessing that dedicated old Prof. Whoever in CSC160, you call up, in order: some memories, MatLab, and the *New York Times*. See Fig. 21.

1.2 A Summer (Re)construction Job

85 Broad Street. The new Monroe County Crime Laboratory. Your internship interview is here. The place smells of new carpet, new latex paint, and old fingerprints. You knock on the pebbled glass door of Dr. Hirsch's lab; a technician says he's out, but hands you a 3" square yellow sticky. It's got a short grocery list on the smooth side. On the sticky side, it says *blur.jpg, get lic. #, 30 mins. — H.*. Oh boy: a classic "in-basket interview".

The file is data that looks like Fig. 2 when you use Matlab's `imshow` command.

Figure 2: **A Matlab imshow of blurred image.**

Yikes. Is that plate number a 2389, or a 5698, or what?.... But wait! You go to the CSC160 webpage from last year, read a bit, invoke Matlab and start typing. Lucky that little streak in the lower right is there... 25 minutes later Dr. Hirsch comes in, looks over your shoulder, and says: "You're hired. Let me show you something exciting that just came in.... What was your name again?..." You resolve to send that kindly old CSC160 prof a bottle of Laphroaig for Christmas; you hope his name's on the website somewhere. See Fig 28.

2 A Personal Preview

This is a non-mathematical introduction to several "Megacepts": universal ideas that are widely applicable theoretically, practically, and even metaphorically. They provide insight into physical phenomena, intellectual control of powerful formalisms and techniques, and have practical applications.

I haven't found an introductory treatment of this subject that does not bring in at least Sophomore-level mathematics. I claim we can understand what's going on at a level that will allow significant accomplishments, and that our understanding will make standard treatments more accessible and less intimidating.

This approach will require you to take my word for some things. They are not hard to derive and prove from a few simple mathematical formalizations of the ideas, but we only hint at formalism.

The plan is to introduce you to several concepts that are universal, powerful, general, and practically useful. I hope that later you will see the material presented formally: it's a very elegant body of work.

I hope to demonstrate the power of:

1. A few simple, basic functions.
2. The idea of a basis set of functions.
3. The ideas of a linear shift-invariant system and convolution.
4. The idea of the Fourier transformation, its properties and use.
5. Generally, The idea of an invertible transform. Representing the same thing in an equivalent but different way (using basis functions) allows useful operations that are not at all easy given the original representation. We can losslessly transform the result back to the original representation.
6. Some practical transform-domain signal-processing techniques.

If you're not put off by a few integrals, please take advantage of the embarrassing plethora of good, readable, illustrated, animated, tight, clear tutorials on this subject on the web, many from college signal-processing classes. A few sites that caught my eye are in the reference section.

3 Two Functions and an Operator

Dirac Delta: This simple but historically controversial "function", often written $\delta(t)$ or $\delta(x)$ (Fig. 3), is a powerful tool for modelling ideal versions of things like a very quick sharp input (voltage spike, clapper striking a bell, etc.) or sampling a function. Conceptually, the delta function is 0 everywhere except at 0, where it is "infinite" in just the right way that the integral over any interval containing 0 is 1. The concept can be formally defined as a limit of narrower and narrower, but higher and higher bumps that all integrate to 1. As a model of sampling, $\delta(t)$ is shifted, say by c, to give the function $\delta(t-c)$. Then the product $\delta(t-c)f(t)$ is zero everywhere but at c, where its integrated value is $f(c)$ since the integral of $\delta(t)$ there is 1. Thus the output is a scaled, shifted delta function, whose 'value' (integral, often pictured as height) is the sampled value of $f(t)$.

Sampling produces values of the function of interest (a light field projected by a camera lens, a sound waveform,...) over some range of space or time. Often these values are regularly spaced. A simplified model of sampling in one dimension is to multiply the input

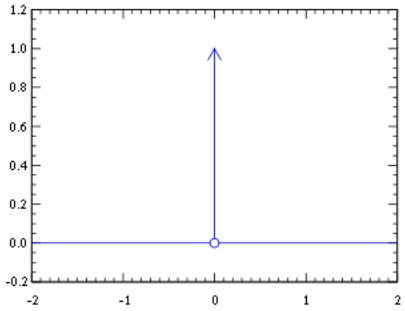
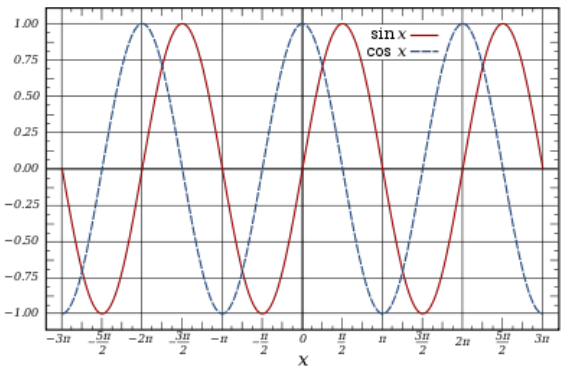


Figure 3: **Dirac Delta Function $\delta(t)$ or $\delta(x)$ has infinitesimal width, an infinitely sharp peak, value 0 everywhere but at 0 and its total integral is 1. It is also called the unit impulse function.**

function by a *Dirac Comb*, or regularly-spaced sequence of Dirac delta functions. The resulting function is an array of weighted deltas. Except for the regular spacing, the situation looks like Fig. 8.

Sinusoids (Sines and Cosines): If you don't remember sines and cosines from high school trigonometry, please go look them up. However, we care more about them as functions (of time or space) than as numbers that tell you things about the sides of triangles. Sinusoids are characterized by *amplitude, phase* and *frequency (or wavelength)* (Fig. 4). Amplitude is basically "height", wavelength (in meters, say) is the distance between repeats of the periodic function, the frequency (in inverse seconds, cycles per second, or Hertz (Hz.)) is inversely proportional to the wavelength, with the constant determined by the medium the wave is in (for electromagnetic waves it is c, the velocity of light). A shifted version of the sinusoid differs in phase: for sines and cosines, a phase shift of 2π is equivalent to one of 0.



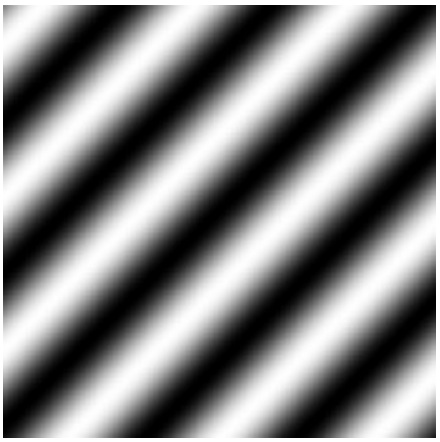


Figure 4: **Left: The sine and cosine. A sinusoid has a wavelength, a frequency inversely proportional to wavelength, amplitude (height), and phase (a shift: the cosine is out of phase with the sine by $\pi/2$ radians). Right: a 2-D sine wave grating.**

For reasons of mathematical leverage, elegance, and actually clarity, sinusoids are often represented as complex exponentials $e^{-i\omega x}$ for real ω, with e the base of natural logarithms.

We like the exponential because its derivative and integral are simply multiples of itself, which often keeps the mathematics under control. A quick refresher on complex numbers is in Appendix A.

You can look up *Taylor Series*, or you can take my word that $\sin(x), \cos(x), e^x$ can each be represented by an infinite series. It may seem that this idea is worthless, just complicating things, but it is a very common technique in the sort of applied mathematics that engineers do.

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} - \cdots$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$$

You could notice here that $\sin(0)$ is zero, $\cos(0)$ is 1, and that the $\sin(x) \approx x$ and $\cos(x) \approx 1$ for small x, as expected. The higher powers of x and the factorials work together to make the terms shrink and the series converge to a finite value.

You can also derive *Euler's Formula*

$$e^{i\theta} = \cos(\theta) + i\sin(\theta)$$

just by multiplying the sine series by i, adding it to the cosine series, and checking that the sum is the exponential. For time-varying signals, we often write $e^{i\omega t}$ where ω is angular frequency (radians per second).

For 2-D sine waves as (Fig. 4,) some trigonometric math leads to the fact that $\cos(ux+vy)$ and $\sin(ux + vy)$ are 2-D sinusoids as in the figure. Their ridges and troughs fall along the parallel lines $ux + vy = k\pi$ for integer k, and their wavelength is $2\pi/\sqrt{u^2 + v^2}$. So we can write a 2-D wave as $e^{i(ux+vy)}$.

The Dot Product (Inner Product): Fig. 5 from Wikipedia should refresh your memory of the normal 2-D interpretation.

The general version of the concept is the inner product operation between N-dimensional vectors x and y:

$$x \cdot y = \sum_{i=1}^{N} x(i)y(i). \tag{1}$$

For unit or equal-magnitude vectors, which only differ in direction, the dot product is a reasonable measure of their similarity since it is a monotonic (cosine) function of the angle between them. Another less-mathematially-friendly measure is their vector difference. The dot product can also be considered as a similarity measure even if the vectors are of

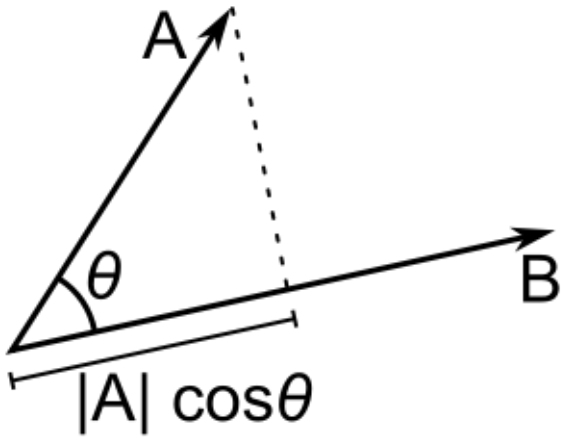


Figure 5: **The dot product.** $A \cdot B = \mid A \mid\mid B \mid \cos\theta$**, the *scalar projection* of** A **onto** B **is** $\mid A \mid \cos\theta$**.**

different magnitudes. The dot product has mathematical advantages, and is used in many current computer applications such as document classification, image understanding, and other high-dimensional matching tasks.

You may not have thought of also extending the idea to "continuous vectors", because you're not Hilbert, but who is? The formalization of inner product in a Hilbert space encompasses the idea of projecting one function on another. By analogy to Eq.(1) we can use the 'continuous sum', or integral, to define

$$f \cdot g = \int f(x)g(x)dx \tag{2}$$

as the inner product of the two functions, and can consider it a measure of how much the functions "look alike". If you consider $\int sin(x)sin(x+c)dx$, the sines are the same and the product is always positive if $c = 0$, so the integral is at a maximum. The product is always negative if $c = \pi$ (Fig. 6). At $c = \pi/2$, there are an equal number of positive and negative terms in the sum and the integral goes to zero. We can sense something of the "matching" semantics of the inner product.

Good News Department: You may see two or three other integrals in this paper, but they all look just like (are) inner products. So Eq. (2) is as bad as the math is going to get.

4 Basis Functions

When a sound enters your ear it is a possibly complex wave of pressure that varies in amplitude and frequency: When you're young you can hear from about 20-20,000 Hz. Your ear senses this broad band of frequencies with vibrations of the eardrum, which are then transmitted to the inner ear's cochlea (Fig. 7). You can see the cochlea's spiral nautilus-shell morphology, which suggests part of the mechanical process of identifying differing sound wavelengths (inverse frequencies) in the eardrum's output.

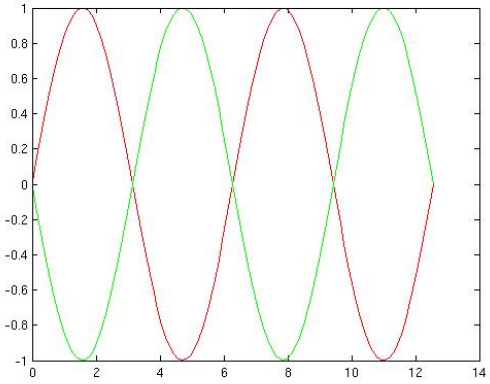


Figure 6: **Sine waves shifted by π**

The point is that the information in complex input from the eardrum is transformed by the cochlea as a set of weighted simpler components, each representing some small frequency range.

Figure 7: **The Inner Ear. Speaking metaphorically, the canals (upper left) do 3-axis inertial guidance and the cochlea (lower right) does Fourier Transformation, splitting the sound into separate frequencies.**

Moving on from amateur biology: a set of *basis functions* is simply some family of functions that are used like this: a weighted sum of them is equal to some other function of interest. *Orthogonal basis functions* have the nice property that they don't interfere with each other: their inner product (integral of the product of one with the other) is 0. It's easy to see that $\delta(t-c)\delta(t-d)$ is 0 if $c \neq d$, (otherwise it is "infinite" but in the "right" way). Orthogonality makes the inner products easy to visualize and compute.

It may seem silly to represent one function by a sum of a bunch of other ones, but there can be advantages (remember the ear?) For now, let's consider our Dirac deltas as a family of basis functions. If we're willing to consider infinite sums, it seems reasonable that we can represent a function $f(x)$ of one variable by the sum of an infinite number of weighted Dirac deltas, with the weight at x being $f(x)$ (Fig. 8).

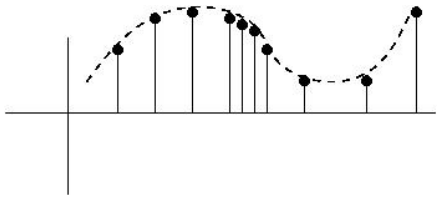


Figure 8: **A finite number N of samples gives a discontinuous function whose value is 0 almost everywhere but which agrees with the function at N places. The agreement is 'better' with denser samples. With an infinite number of samples, the constructed function agrees with the original everywhere.**

So $\delta(t)$ can form an orthogonal basis set for a one-dimensional function; maybe not a terribly elegant or even useful set, but it works. Also clearly it works for 2-D or N-D functions.

Not so obvious is that *sinusoids are an orthogonal basis set* for a big class of useful functions of arbitrary dimension (e.g. Fig. 9). The name most closely associated with the sinusoidal basis set is Fourier (as in Fourier series, Fourier Transform, Fourier analysis,...).

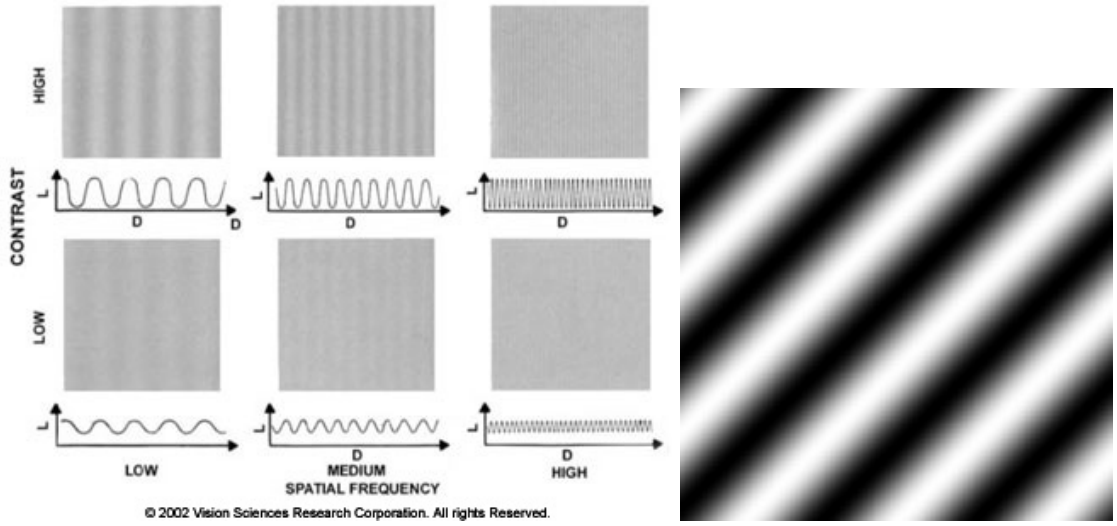


Figure 9: **Two-dimensional sinusoidal gratings are orthogonal basis functions for 2-D functions like images. Left, low-amplitude sinusoids at various spatial frequencies and one orientation. right, high-amplitude grating at another orientation.**

Applying the inner product of Eq. (2), we're claiming two sine waves of different wavelength are orthogonal: their inner product is zero:

$$\int \sin(\alpha x)\sin(\beta x) = 0, \alpha \neq \beta$$

To visualize the situation, think of one sine function above another of a different wavelength, their product, and its integral (you can easily use Matlab to create and plot a vector version of this argument). The two sine functions will sometimes be of the same sign and

sometimes of opposite sign: (in fact, half the time each.) It's really easy to believe this if one of them has a long wavelength and the other a very short one. In fact eventually all the points of positive product have to have a corresponding point of the same magnitude but negative sign. So in sum (in the integral) the positive and negative contributions cancel, the integral is 0, they're orthogonal. Proving they're a basis set I leave to your Calculus professor, but a suggestive little demonstration involves building a very sharp-edged function (a square wave) out of sine waves (Fig. 10). It isn't a proof but it's a pretty impressive accomplishment for a bunch of smooth functions to add up to a sharp one, no?

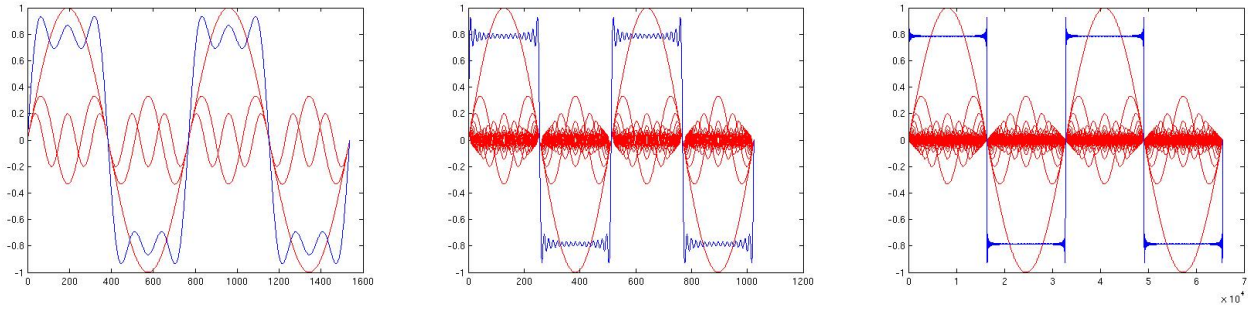


Figure 10: **Adding sine waves to get a square wave: three, 100, and 256 sines shown.**

One of the prime goals of this paper is for us to be able to think about, say a sound signal as 1) a pressure wave through time (what your eardrum or a microphone responds to) and 2) a collection of sine waves of different frequencies (what a graphic equalizer or your cochlea works with). We'll see that signals can be represented and processed in either domain and transformed between domains. The idea of such transforms is a key and ubiquitous concept: just for instance, in pure mathematics or control theory the Laplace and Z transforms can simplify solving a big class of differential and difference equations.

There are many, many sets of basis functions. Good ones for analyzing vibrations (vibrational modes) in a drum head are Bessel functions. Spherical harmonics are good for modes of magnetic fields in planets and stars, electron orbital configurations, etc. Legendre polynomials spring from basic differential equations in physics. The Laplace transform basis functions are damped exponentials.

Fig. 11 illustrates two these basis functions. There are fun animations and experiments on the web illustrating vibrational modes:
`http://paws.kettering.edu/ drussell/Demos/MembraneCircle/Circle.html`
`www.youtube.com/watch?v=v4ELxKKT5Rw`.

5 Linear and Shift-invariant Systems

5.1 An Introduction

We have used the words "linear system" to mean a system of linear equations. Henceforth in this module, a linear system is one composed of *linear operators*. This type of linear system forms the core formal object of much elegant mathematics and many practical modeling

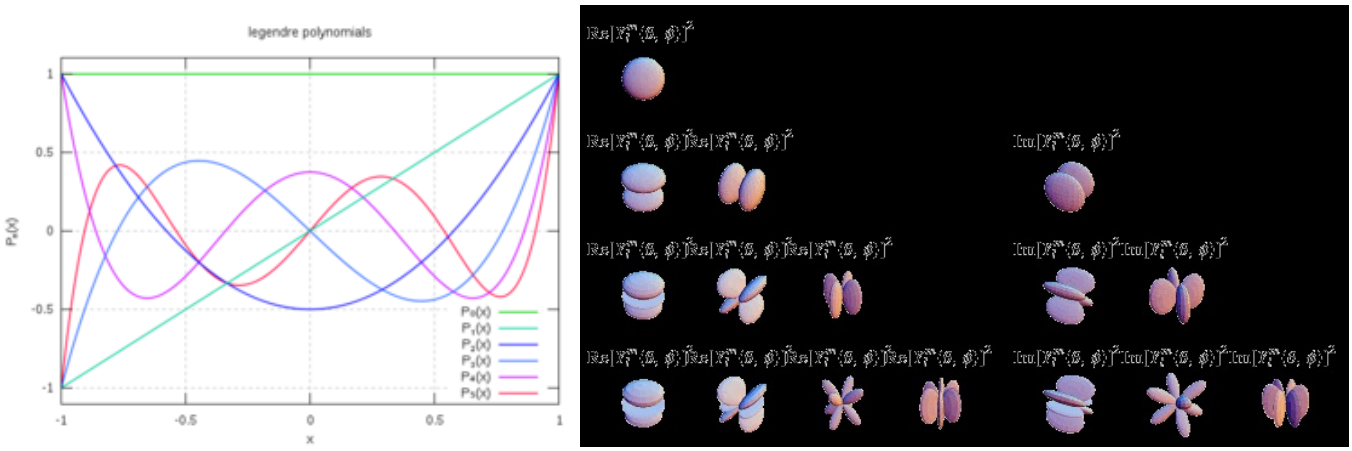


Figure 11: **Two of many orthogonal basis sets: Legendre polynomials and spherical harmonics.**

techniques. The library has many books with 'Linear Systems' in their title. For our application here you can find a responsible treatment (unlike this one) in books with 'Digital Image Processing' or 'Signal Processing' in their title (see the References Section).

Linear systems are powerful and beautiful mathematically, and approach the Platonic ideal of predictable systems that "do what you'd like or what you'd expect".

By linear operator, we mean this: let $f_1(t)$ and $f_2(t)$ be input functions $g_1(t)$ and $g_2(t)$ their corresponding output functions, and α and β be scalar weights. If we make up a new input function $\alpha f_1(t) + \beta f_2(t)$, then the output is $\alpha g_1(t) + \beta g_2(t)$. The system effectively treats each part in the input separately and independently. For a 'superposition' (addition) of weighted inputs, we get the superposition of weighted outputs.

You can check that matrix multiplication is a linear operation over vectors. Indeed in general, any linear transformation can be thought of as an appropriate "matrix multiplication" operation on an appropriately represented "input vector". (It turns out that function inputs can be viewed as infinite-dimensional vectors in a way that can be mathematically well defined. In this case, the "matrix" is a function of two variables that is multiplied by the input function inside an integral that functions as an analog to the summation in normal matrix multiplication).

Matrix multiplication is a good place to start for some important intuition. Linear systems have the above nice properties, but it is still true that if $y = Mx$ (a vector-matrix equation) *each* element of the output y can be a sum of *all* the elements in the input x, each weighted by the elements in a row of M. Thus the output may not be all that obviously related to the input.

We are going to concentrate on a very special subclass of linear systems, the linear shift-invariant (LSI) systems, to model certain physical processes, like imaging.

If x is a one-dimensional "scene" and y its image, and M models a camera, we'd want a camera C1 that did *not* mix up the whole scene somehow into each point of the image. We'd want this:

```
y              C1              x
0       1  0  0  0  0  0  0     0
0       0  1  0  0  0  0  0     0
1       0  0  1  0  0  0  0     1
2   =   0  0  0  1  0  0  0  *  2
4       0  0  0  0  1  0  0     4
0       0  0  0  0  0  1  0     0
0       0  0  0  0  0  0  1     0
```

We have the concept of basis functions and sampling using the Dirac delta. In this mind-set, we see that each row of the matrix multipication samples the input at a different point with a delta function, thus exactly reproducing it at the sample's location (which we can see shifts across the input as we go down the rows). Here we don't need an infinite number of deltas to reproduce the input exactly since vectors are discrete. If every other row (say) were all zeros, we'd undersample we'd not have a shift-invariant system.

¡p¿ It actually may be more comfortable to imagine these as infinite matrices and vectors... zeroes in both directions for the vectors and the single shifted 1 in each row of the matrix. Then each row is a shifted version of every other one, and so if we shift the input vector up or down by some amount, the output vector is similarly shifted. We have a perfect 1-D digital camera with infinite field of view that exactly reproduces the input. It's no accident that the rows are digital versions of the (shifted) delta function.

An smaller, out-of-focus camera C2 might be modeled like this: each image element is the average of up two adjacent scene elements: a slight blur. Note the annoying 'edge effect' that gives the 'wrong' answer for the last element of `y` – a potential problem for all finite versions of ideally infinite arrays.

```
 y          C2          x
1/2   =   1  1  0  0  0     0
3/2       0  1  1  0  0     1
3     =   0  0  1  1  0  *  2
2         0  0  0  1  1     4
0         0  0  0  0  1     0
```

A camera C3 that behaves nicely in the middle of its field of view but goes blurry around the edges could be:

```
y                      C3                        x
0       1/2   1/4    0    0    0    0     0      0
1/4     1/4   1/2   1/4   0    0    0     0      0
1        0     0     1    0    0    0     0      1
2   =    0     0     0    1    0    0     0   *  2
4        0     0     0    0    1    0     0      4
1        0     0     0    0   1/4  1/2   1/4     0
0        0     0     0    0    0   1/4   1/2     0
```

Now we can notice:

- C1 and C2 are even closer to the Platonic ideal of "do what you'd like or what you'd expect" because a shifted version of the same input is always the same output, shifted. They are special because their matrices are the super-simple form: just one row, repeated and shifted.
- The output vector can be considered to be indexed by the amount a matrix row has been shifted.
- C3 is a linear transform but the function does not act "the same everywhere", as do shifted rows. Here the "edges of the field" are blurred and the center isn't. C3 is not shift-invariant.
- An LSI matrix is a wasteful way to write down what's happening, though it is perfectly mathematically correct and proper. It's easy to see that each output element is a dot product of the input with a differently-shifted copy of the "one row" in the matrix. We can (and almost always do) rewrite the matrix multiplication elegantly as a vector of dot-products, each one between the input and the same (but shifted) row. The operation that creates the shifted dot products, and indexes the output by the shift, is the discrete *convolution* operation.
- This whole story works in the continuous domain, with a continuous function as input, and a shifting continuous function instead of a matrix row. Continuous addition is integration, so we obtain the (continuous) *convolution integral*.

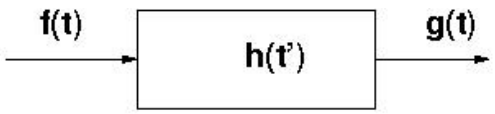
Thus a useful special class of linear operators, defined more formally below, is called "convolutions", operations that can be specified by a "weighting function" (that is, the row of the matrix or a continuous version of it) that is "swept across" the input and multiplied element (dot-product) wise at every shift to produce the output.

Fig. 12 shows that we'll consider a linear (usually LSI) system as a black box with input(s) and output(s), or input and output vectors or functions. The black box is characterized by the operation it performs on the input.

What's the black box? Linear operators can be used to represent systems like like cameras, stereo amplifiers, and even mathematical operations. If the operation is a convolution, (whose definition we will see later) the weighting function is often called the *impulse response* for input functions of time, or the *point-spread function* for functions of space.

A linear, *shift-invariant* (LSI) system produces a similarly-shifted input for a shifted version of the input: If $f(t-h)$ goes in, $g(t-h)$ comes out. The matrices representing LSI systems have a repetitive diagonal structure to them, like C1 and C2 above.

For example, one dramatically non-shift-invariant linear optical system is a kaleidoscope, which adds up variously-weighted bits of the scene to create each patch of image. The familiar colored chunks of glass bring in the interaction of different wavelengths in our color perceptions, which is a non-linear process (red plus green don't make 'reddish green'). "Non-linear optics" deals with light in nonlinear materials: the result is a breakdown of the superposition principle, and we get phenomena such as frequency-doubling or - tripling.



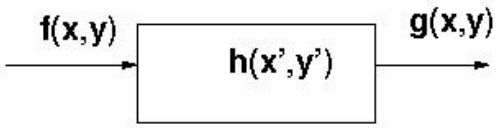


Figure 12: **Linear shift-invariant systems, one for time-domain input (like sound), one for 2-D spatial domain input (like images).** $f(t)$ **and** $f(x, y)$ **represent inputs,** $h(t\prime)$ **and** $h(x\prime, y\prime)$ **impulse response functions, and** $g(t)$ **and** $g(x, y)$ **the outputs. The box performs *convolution*.**

Nonlinear dynamic systems are famous nowadays through 'the butterfly effect' and chaos theory: small variations in input or parameters can have big effects on the output (as can the Lorenz attractor from the ODE topic in this course).

One example of a true LSI system is simply that the black box's $h(t)$ is the (linear) differential operator d/dt, which takes the derivative of the input. We remember that $d/dt(af(t + c) + bg(t + d)) = af'(t + c) + bg'(t + d)$: superposition holds, thus we get linearity. Another example: if the input is a vector (or a sum of weighted vectors), then matrix multiplication is a linear system (as we saw above, to be really shift-invariant the vectors and matrices must be infinite).

Fig. 13 shows how linear systems behave, using a camera analogy.

Note: Fig. 14 shows that a real pinhole does not produce a point image from a "point input."

Real systems are not linear: often they do approximate linearity over some range. Cameras do not have infinite fields of view. They distort the image optically in various ways both at large scales (pincushion and barrel distortion) and small (Fig. 15).

Amplifiers, speakers, our ears... all have volumes and frequencies below and above which they distort or do not respond. Light sensors, like our rods and cones, film, and CCD arrays in cameras, act the same way: they have preferred wavelengths and only a range of light amplitudes to which they respond (Fig. 16).

5.2 Properties

Again, you're not seeing proofs of any of this, nor the usual mathematical formalization: we're after visualizible concepts and intuitions.

We are lucky that we can describe precisely two very general ways to think about any linear, shift-invariant system (LSI).

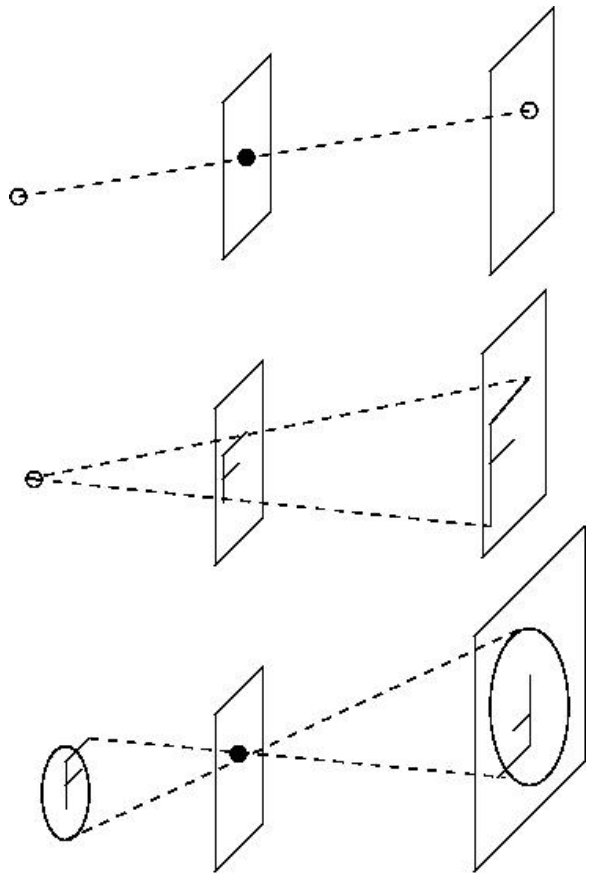


Figure 13: **The ideal pinhole camera as linear system. In terms of Fig. 12, f is the object in the world on the left, h is how the camera transforms a point (here to another ideal point or an F shape), and g is the image. In the top example, if another point is added in the scene, we expect its image to show up in the right spot, and not to interfere with the original point's image: that's linearity. The camera deals with ideal geometrical points and straight lines, not real physical imaging phenomena.**

1. It performs a *convolution* in the time or space domain of its input and its impulse-response function. In fact, if a system performs a convolution, it's LSI and *vice-versa*.

2. We'll see in Section 6 that it acts like a graphic equalizer in the frequency domain, amplifying and attenuating (and possibly phase-shifting) the various wavelengths represented in the signal.

Wikipedia's "Convolution" article has some very helpful animations and images; it states: The convolution of f and g is written $f * g$ [or as $f \otimes g$.] It is defined as the integral of the product of the two functions after one is reversed and shifted.

With apologies, here is the formal definition of the convolution, since it's a ubiquitous concept and since we can understand it now as just another inner product. The one-dimensional

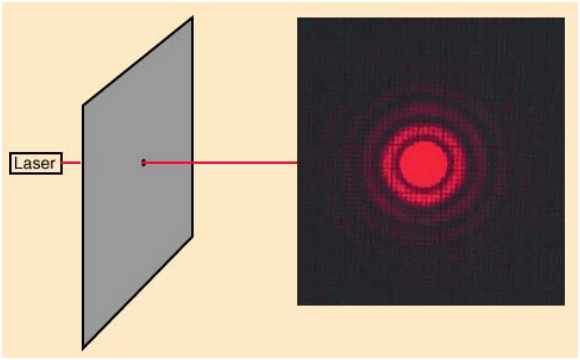


Figure 14: **Point-spread function of a real pinhole is the Airy disk, resulting from diffraction of the incoming light waves.**

discrete convolution is the short way to write the matrix multiplications in the examples of LSIs in the introduction: it is given by:

$$(f \otimes g)(t) = \sum_{i=0}^{n} f(i)g(t-i).$$

You can see that the sum computes a dot product, and t is the shift. Its one-dimensional continuous analogue is the famous *convolution integral* given by:

$$(f \otimes g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau))d\tau = \int_{-\infty}^{\infty} f(t-\tau)\, g(\tau)\, d\tau$$

Note that it's commutative. It is also an example of how a linear operator in continuous space can be described by an integral of the product of a function $f(\tau)$ with a two-parameter function of τ and an index variable t that appears externally, here $g(t-\tau)$. This is an example of the above-mentioned integral generalization of matrix multiplication. We will see another example of this generalized form shortly, in the Fourier Transform.

The use of τ here need not represent the time domain. If it does, the convolution formula describes a weighted average of the function $f(\tau)$ at the moment t, with the weighting given by $g(\tau)$ shifted by amount t. As t changes, g slides by, and its weighting function moves across the input function.

Convolution is also defined for arrays: due to the shifting and overlap, an $N-$-vector convolved with an $M-$vector has length $M + N - 1$.
In Matlab: `conv([1 2 3 4], [2 4 6])` gives `[2 8 20 32 34 24]` and in 2-D `conv2([1 1;1 1],[1 1;1 1])` is `[1 2 1; 2 4 2; 1 2 1]`. Note that the operation can be represented by multiplication of the (extended) input vector by a matrix whose rows consist of shifted copies of the (reversed) convolution weights (diagonal structure). The meaning of "shift invariance" is well illustrated by this representation.

The negative sign on one of the τs means that this function is backwards. You've seen this phenomenon in the bottom case of Fig. 13, and for many physical systems it makes sense. In Fig. 17, imagine the red function being emitted as a function of time: it starts

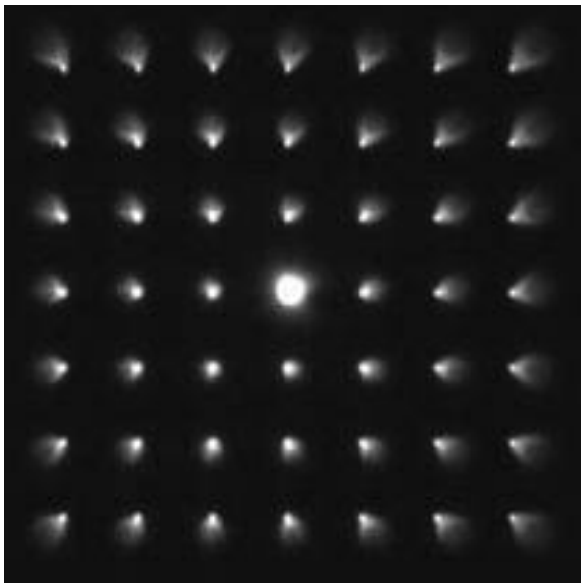


Figure 15: **In coma the 'ideal' impulse response (a point) is distorted to ellipse or comet shape, often systematic and increasing with the angle of imaged point off the optical axis of the system. In chromatic aberration, different wavelengths are focussed at different distances, hence impulse response varies with color.**

out big and decays. If this function is f sent into a system with the blue function as h, the flipping makes sense.

A closely-related concept is called *correlation*, a distant cousin of the more familiar statistical concept. The operation is usually just written as * or $\otimes$ as before, disambiguated by context if need be.

$$(f \otimes g)(t) = \int_{-\infty}^{\infty} f(\tau)g(\tau + t)d\tau = \int_{-\infty}^{\infty} f(\tau + t)g(\tau)d\tau$$

We see that the correlation is even more like the familiar dot-product or inner product of two vectors, since now the indices of the product terms are varying in the same direction. Again, the analog to matrix multiplication applies: for each t, $g(\tau+t)$ is like the $(\tau+t)$th row of an (infinite) transformation matrix. We can use the correlation to implement matching of vectors or functions, since it is made up of pure (shifted) dot products. The *autocorrelation* (correlation of a function with itself) is maximized at zero offset, and for aperiodic functions tends to drop off with distance. You can see that for an infinite picket fence (or dirac comb or sine wave), the auto correlation is itself periodic.

A simple consequence of definitions is the nice fact that if $g(t) = h(t) * f(t)$, the input $f(t) = \delta(t)$ (the Dirac delta), then for any LSI the output $g(t) = h(t)$. This is useful because we may not be able to look into the black box but if we give it an impulse as input, the output is its defining $h(t)$. This you can see from imagining a Wikipedia-animation scenario (or Fig. 17) with impulse input sliding across $h(t)$.

Another interesting and basic fact about LSI is that *sinusoids are eigenfunctions of linear, shift-invariant systems*. That means if the input $f(t)$ is a sine wave, the output $g(t)$ is a sine

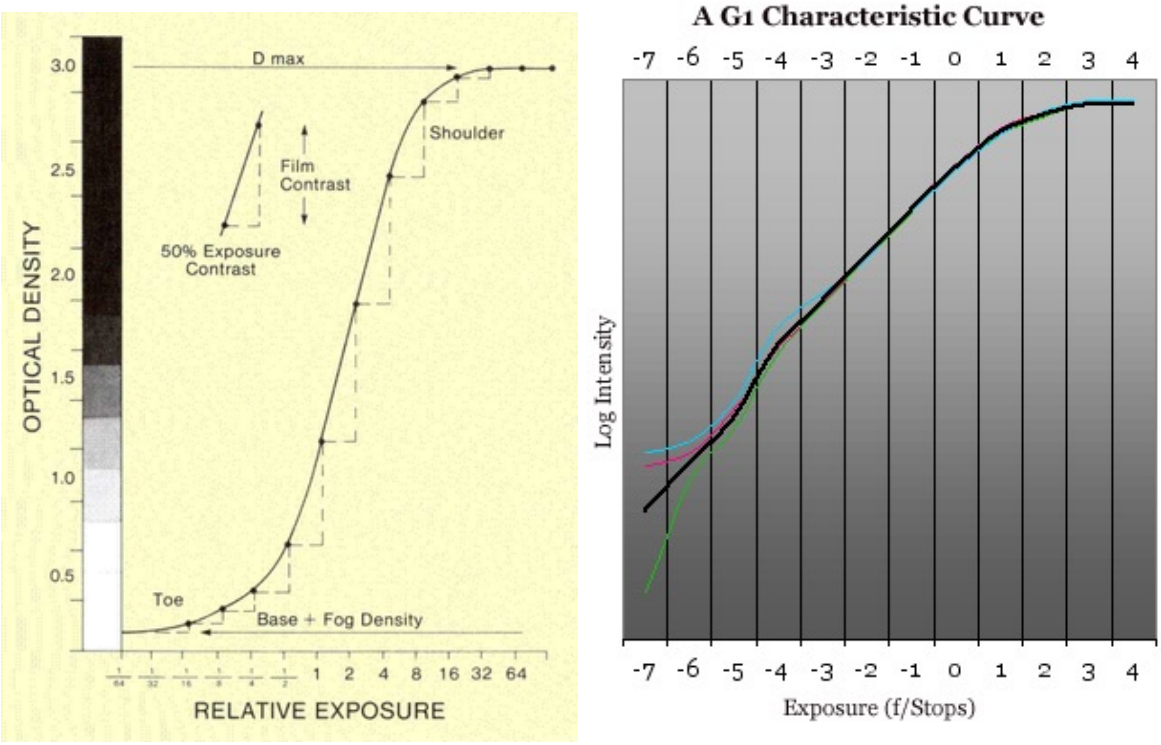


Figure 16: **The modeled response of photographic film and the actual response of one type of CCD (digital camera sensor): As is typical of real sensors, they are linear over a range at best, with nonlinear rolloff or saturation.**

wave of the same frequency *for convolution with any arbitrary* $h(t)$! Amazing but true. Output $g(t)$ may be scaled (amplified or attenuated) or shifted in phase, but it's still a sine wave of the same frequency. This property has a one-line proof based on the definition of convolution and our favorite facts that 1) the integral of e^{kx} is $(1/k)e^{kx}$ for any real and complex k, and 2) for imaginary k, e^{kx} is a sinusoid.

6 The Fourier Transform

6.1 Definition

The Fourier transform is a linear operator that we can think of think of as taking input in the time or space domain (say a sound, a voltage waveform, or an image) and producing as output an equivalent (lossless and invertible) representation of the input in a frequency or spatial-frequency domain. In other words, it decomposes the input into a number of sinusoids of varying magnitude and phase (and in two dimensions, directions). The inverse transform inverts the transformation. I can't resist showing the mathematical definition:

$$F(\nu) = \int_{-\infty}^{\infty} f(t)\, e^{-2\pi i \nu t} dt,$$

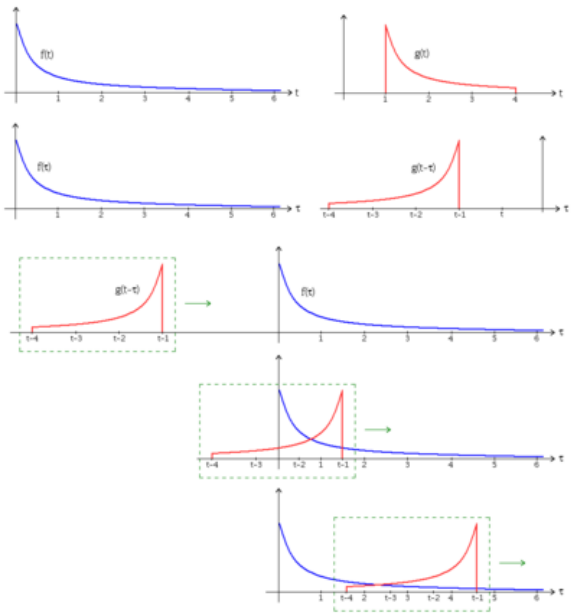


Figure 17: **The convolution operation. The red function is turned backwards (line 1) and slid past the blue one (lines 3,4,5): the output is the area under the red function *weighted by the blue one* or *vice-versa*.**

where t is time or space and ν is frequency. The inverse is simply related:

$$f(t) = \int_{-\infty}^{\infty} F(\nu)\, e^{2\pi i \nu t} d\nu.$$

(We have left out a scaling factor involving the square root of 2π needed to ensure that no net scaling occurs when the transform is followed by its inverse)

Notice that the Fourier transform (FT) is another of our inner-product integrals, analogous to matrix multiplication. This one can be read as answering the question "how much of this particular sine wave $e^{-2\pi i \nu t}$ is in this input function $f(t)$?" It is projecting $f(t)$ into the *transform basis space* of sinusoids.

Note that in general the FT is complex, but read on.

6.2 Properties

Let $\mathcal{F}$ denote the Fourier transform operator, so $\mathcal{F}\{f\}$ is the Fourier transform of $f(t)$, which we also write as $F(\nu)$.

The Fourier transform (FT) has a number of pretty properties. In no particular order:

- FT is linear. That is, the FT of a weighted sum of functions is the weighted sum of their FTs.
- FT of any symmetric (even) function is real and even: (the function is a sum of cosines).

- FT of any antisymmetric (odd) function is imaginary and odd: (function is a sum of sines).
- FT of a real function has the property that if $F(\omega) = x + iy$ then $F(-\omega) = x - iy$. Such a function is sometimes called *Hermitian*, although the term is more frequently used with respect to operators that display the same conjugate behavior under index reversal rather than sign reversal.
- Scaling or similarity: $\mathcal{F}\{f(ax)\} = \frac{1}{|a|}F\left(\frac{\nu}{a}\right)$.
- Thus a time-reversal property: $\mathcal{F}\{f(-x)\} = F(-\nu)$.
- $\mathcal{F}(\text{Gaussian})$ is a Gaussian.
- FT of a Dirac comb is a Dirac comb.
- Examples: from the scaling property and the last two properties, we see that the FT of a narrower (wider) Gaussian is a wider (narrower) one, and the FT of a higher (lower) frequency Dirac comb is a lower (higher) frequency one. The FT of a (single) Dirac function is flat: it has waves of all frequencies and the same magnitude (do the FT integral and prove it!).
- In the FT of a shifted function, the magnitude of all the components (the complex numbers) stays the same, but they rotate (their phase changes) as a function of the shift and their frequency.
- FT has an equivalent for (discrete) vector inputs. It has a very clever implementation called the Fast FT, or FFT, which we'll be using (along with *tout le monde*.

The Fast Fourier Transform (FFT): The FFT is fast because the discrete FT transform is just a matrix multiplication that normally would take N^2 multiplications to transform an N-vector (an N-long dot product for each output component). However, the FT matrix has a very elegant internal structure that can be exploited to make the multiplication faster, so FFT runs in order of $N \log N$ time. For large N, this can result in significant savings (*e.g.*, $1,000,000^2 = 10^{12}$, but $1,000,000 \log(1,000,000) = 6,000,000$.)

The FFT works fastest on arrays of length 2^n for integer n, but can be generalized beyond powers of two, and is then most efficient for lengths with small prime factors: 60 (factors 2,3,5) would be faster than 59 (a prime number), for instance, but not as fast as 64.

Signal shifts cause FT rotations: See appendix A. We can think about a complex number $z = (a + bi)$ as a vector in the real-imaginary plane with coordinates (a, b) or in polar form a magnitude $\sqrt{(a^2 + b^2)}$ and direction $\tan^{-1}(b/a)$ (or `atan2(b,a)` for us computer types.) For a complex number $conj(z) = conj(a + ib) = a - ib$, so $z \cdot conj(z)$ gives the squared magnitude of the number $(a^2 + b^2)$ (Appendix A).

On the 'properties' point above about the shifted function, the individual elements in the FT are *phasors* (see Wikipedia for an animation) (Fig. 18); a phasor is a rotating complex number that is related to sinusoids: a phase change (shift) a sinusoid is the same as rotating its phasor.

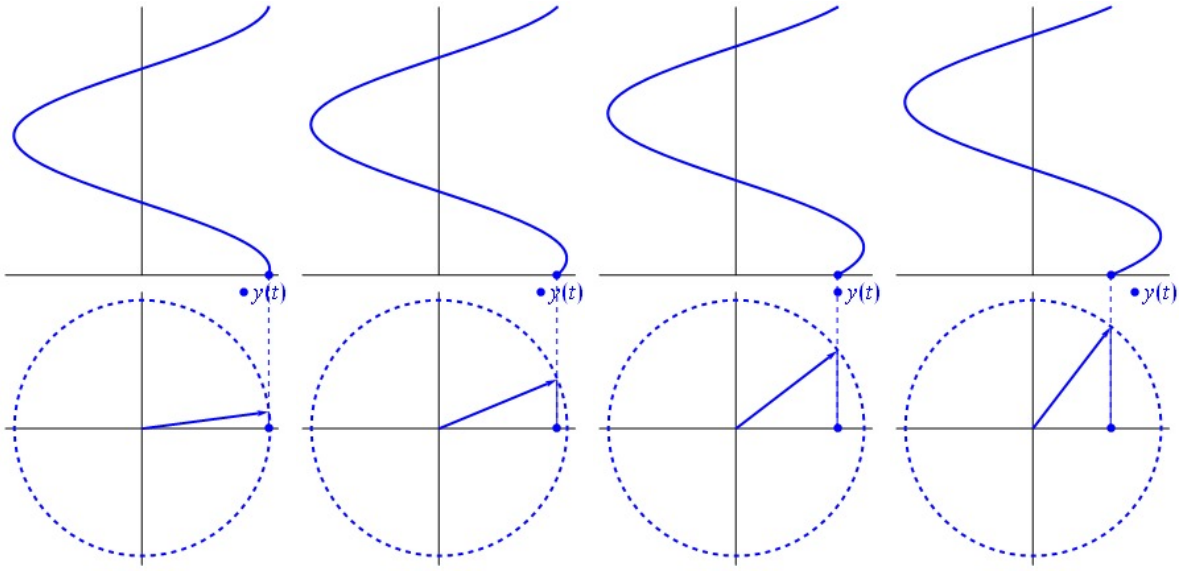


Figure 18: **A phasor. The sinusoid associated with it is shown. Note same amplitude and frequency, but wave is shifted ("starts at a later point in its cycle") as the phase angle increases. When phase difference hits $2 * pi$, wave looks exactly the same (as does the phasor).**

The power spectrum: If the FT is multiplied pointwise by its own complex conjugate, $P = conj(F) \cdot F$, every element of the product P is real and positive, and is the squared magnitude of the FT's complex phasor. P is a very important concept called the *power spectrum* of the function since it shows the power of all the sine waves in its spectrum (that appear in its Fourier Transform representation). It is normally how we visualize (plot) a FT, and we often hear power spectrum plots referred to as FT plots (examples in Appendix B).

The power spectrum (PS) tells a lot about the signal. If we get the power spectrum of the vibrations of a jet engine we can read right off it the frequencies at which it is vibrating most strongly and may choose to redesign or modify it if we don't like what we see. The PS of a signal is "de-spaced", in that information about the power at a given frequency is gathered from the whole signal.

Periodic and aperiodic functions: In the ideal continuous mathematical world, we can use infinite, periodic inputs like sines. In Matlab, we must use finite arrays as input to the FFT, and it's important to realize that the FFT *treats this array as if it were an infinite periodic array*! That is, it repeats, or wraps around (to make a ring in 1-D and a torus in 2-D). For instance, the vector [1 2 3 4 5 6 7 8 9] seems to describe a nice smooth ramp but if it is understood to be periodic, there is a disguised, sudden drop back to one: ...7 8 9 1 2 3 .... As we shall see we pay the FFT just to notice this sort of phenomenon, so we can't forget about it. We often do want the non-repeating function, as when we match by 'aperiodic correlation'. For this, simply embed the data array (say it is $N \times N$) in a $2N \times 2N$ array of zeroes and use the bigger array. The calculation of course still thinks this bigger function is periodic but the array you are interested in will not 'interfere with itself'. In Matlab the padding is easy; if `X` is an N by N array, then `fft2(X, 2*N, 2*N)` works.

FT and power spectrum coordinates: The logical coordinate system for a FT or power spectrum has the origin in the middle, with positive and negative axes as usual. Then the FT of a single cosine wave consists of two real points, one at (u,v) and one at (-u, -v), representing the fact that the wave can be thought of as going in either of two opposite directions. If the cosine is shifted along the axis its phase changes and these two phasors rotate (change phase), and when the wave shifts enough to become an odd function (the sine, in fact), the two points at (u,v) and (-u, -v) are now purely imaginary.

However, when we input a vector to an FFT program, the output has its origin in a corner, (lower left or lower right). There's still symmetry but it's confusing. Matlab has the function `fftshift`, which shifts the origin of the FFT output to the center.

In the literature we almost always see plots of the right half of the one-dimensional power spectra, for 0 and positive frequencies. For 2-D power spectra, the usual display is the entire symmetric 2-D spectrum, with origin in the center (as with `fftshift()`).

Spatial and frequency domain relationships: For images, it is often possible to connect image phenomena with PS phenomena: If the image is 'blobby', slowly-varying, with few sharp edges, the energy in the PS will be in the central, low-frequency region. If there are lots of sharp edges there will be lots of power at the high frequencies (note in Fig. 10 how we had to add more high-frequency sines to make the square wave's edges sharper). If there is an obviously oriented feature (a long, high-contrast edge or parallel lines) in the image, the PS will have a symmetrical spike in the orthogonal direction emanating from the center. A checkerboard like pattern (say a simple basket-weave texture or a wall of windows) induces a Dirac comb-like structure in the PS, etc. See Fig. 19 and Appendix B.

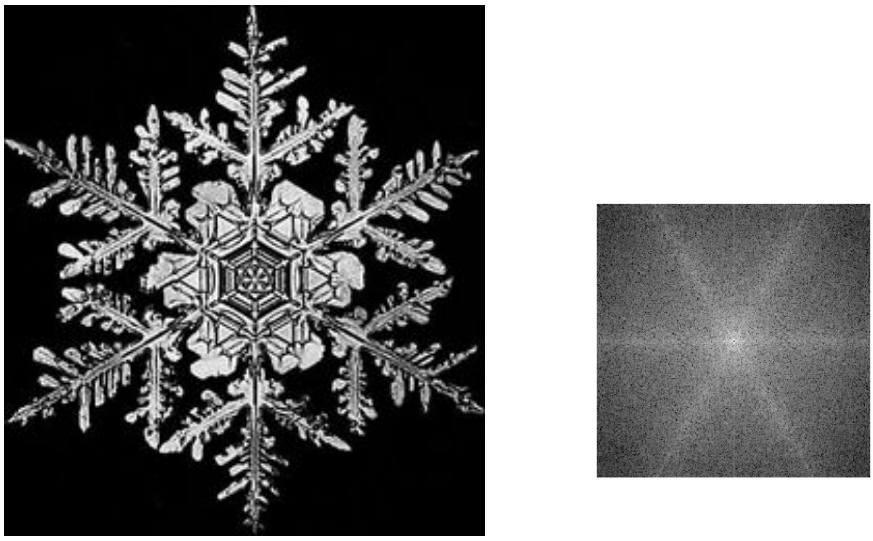


Figure 19: **An image and its power spectrum. The six-fold symmetry is evident in both.**

Shifting example: Fig. 20 shows a sine and slightly shifted sine – imagine their sum is the input. We expect the power spectrum to represent only one amplitude and frequency,; that is, to consist of only two phasors in symmetric places. We expect it to look the same for either sine since the power spectrum throws away the phase information.

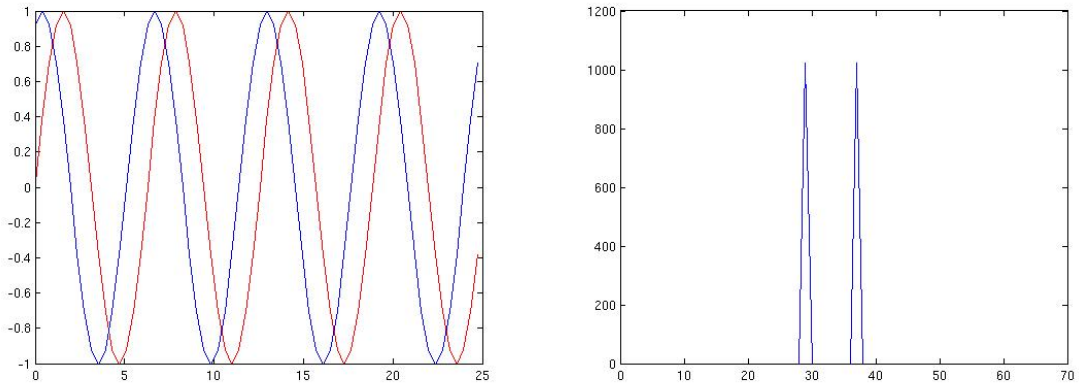


Figure 20: **Left: sines with same amplitude and frequency, different phase. Right: `fftshift`ed power spectrum of either sine wave or their sum.**

Let's dig into the actual vectors produced by the FFTs of the sines. In the (un `fftshift`ed FFT for the unshifted sine we see zeroes and...:

```
... -0.0000 - 0.0000i
  Columns 5 through 8
   0.0000 -32.0000i   0.0000 - 0.0000i...
```

Aha...there's one of the FFT phasors, purely imaginary since the sine is odd. Here's its complex conjugate, symmetrically opposite the origin:

```
Columns 61 through 64
   0.0000 +32.0000i  -0.0000 + 0.0000i ...
```

For the shifted sine, the phasors are at the same indices (5 and 61), but have rotated to create complex matrix elements:

```
  29.5641 +12.2459i
  29.5641 -12.2459i
```

The energy in this wave (the magnitude of its single-phasor FFT) has not changed.

$29.5641^2 + 12.2459^2 = 32^2 = 1024$.

Bad vibrations: The jet engine vibration data (see Fig. 1) was claimed to have some particular dominant frequencies and noise. In fact, we only need to add two or three sine waves together, even with no noise, and unless their frequencies (wavelengths) are really different it's hard to tell what the original frequencies were (try it). The made-up jet vibration data was three pure sines and a surprising amount of 0-mean Gaussian noise all added up. The power spectrum makes their contributions obvious, even to the `New York Times`.

The basic code for the power spectrum is simple:

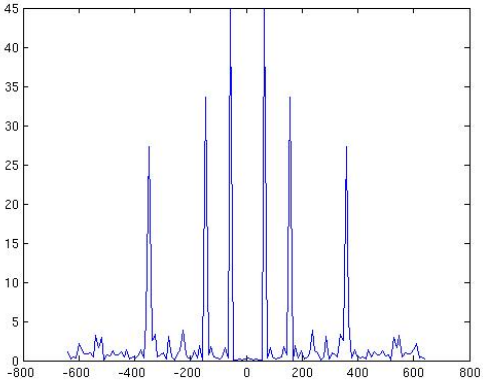


Figure 21: **Power spectrum of vibration data shown in Fig. 1. It is the sum of variously-weighted sinusoids of frequencies 60, 150, and 350 Hz. with a fair amount of 0-mean Gaussian noise. Data was a 128-long vector representing .1 second.**

```
function thePS = PowSpec1D(X,n)
 Y = fft(X,n);  % get the FT
 thePS = (Y .* conj(Y)) / n; % compute the PS
end
... % and in the calling script
plot(xaxis, fftshift(thePS));  % show me
```

In practice, we usually want to supress the 'central peak', which gives the total power and thus dominates the plot scaling, and sometimes want to emphasize detail in (usually smaller) high frequency power by plotting the logarithm of the spectrum or somesuch.

7 The Convolution and Sampling Theorems

7.1 Convolution Theorem

There are more interesting and useful FT properties than we saw in Section 5.2. Here's a big one. *Convolution in the time (space) domain is dual to elementwise multiplication in the frequency domain*. This is the *convolution theorem*.

Let $\mathcal{F}$ denote the Fourier transform operator, so $\mathcal{F}\{f\}$ and $\mathcal{F}\{g\}$ are the Fourier transforms of f and g, respectively. Then

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\} \tag{3}$$

where * denotes convolution and $\cdot$ point-wise multiplication (in Matlab, the .* operation). Also *vice-versa*:

$$\mathcal{F}\{f \cdot g\} = \mathcal{F}\{f\} * \mathcal{F}\{g\}, \tag{4}$$

And applying the inverse Fourier transform $\mathcal{F}^{-1}$ to Eq. (3), we get:

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\} \tag{5}$$

Convolution and correlation are important operations and the Convolution Theorem is a key intellectual idea with wide-ranging practical utility.

Convolution is closely related to a polynomial multiplication and even to integer multiplication: compare the convolution of vectors 1111 and 111 with the product of integers 1111 and 111 (base ten). The convolution theorem says that one way to compute a convolution of functions (vectors, matrices) A and B is to get their FTs, multiply those FTs elementwise, and take the inverse FT of the product. This method is faster for big enough inputs.

Thanks to the convolution theorem we can make a frequency-domain version of the simple block diagram of (Fig. 12). It is (Fig. 22), with the inputs and outputs being the FTs of f and g, the function in the box being the FT of h, and the operation of the box being elementwise multiplication.

Here the box acts like a graphic equalizer: the signal (say sound) comes in as the sum of sinusoids of many frequencies, which are amplified or attenuated by the equalizer (which multiplies the vector of their amplitudes and phases (that is, $F(\nu)$) elementwise by its settings (that is the vector $H(\nu)$, which is called the *Modulation Transfer Function (MTF)*.) The graphic equalizer is just an MTF. For sound, it's a linear system that's easier to think about in the frequency domain than in the temporal domain.

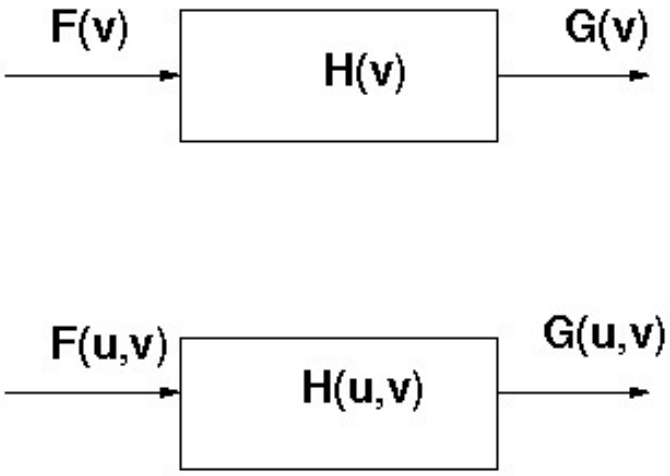


Figure 22: **Compare to Fig. 12. Here $F(\nu)$ is the FT of a signal, $H(\nu)$ is the the FT of the point spread function $h(t')$. $H(\nu)$ acts like a graphic equalizer, changing the amplitude and phase of the incoming sinusoids. The box's operation is Matlab's .*, or element-wise multiplication.**

7.2 The Sampling Theorem

It makes some sort of sense, we feel, to talk about or use a sampled version of a continuous function. We use finite-resolution images from "x-megapixel" cameras, and film was finite-resolution too. Likewise a CD is data from a sound waveform "sampled" at 44KHz. MP3 is a very interesting blend of sampling and psychophysics-based compression that not only samples sound but tosses out the frequencies that it thinks we won't miss. JPEG is similar. They are "lossy encodings", based on sampling and on tossing out less-informative frequencies in the signal.

Rising above the smog of consumer electronics trade secrets, empirical engineering hacks, and evanescent esoterica about data-encoding 'standards', consider the idealized, mathematical, formalizable question: can we *exactly* reproduce a continuous signal from a finite number of (point) samples?

We know enough now to answer that question, even if the exact algorithm isn't clear. Our approach is to think in the frequency domain: every signal is a sum of sinusoids. If we could reconstruct every sinusoid, we could reconstruct the signal exactly. If we can reconstruct a sinusoid (its frequency, amplitude, phase) from a sequence of samples, we can presumably reconstruct any lower-frequency, longer-wavelength sinusoid from the same samples (aha!!). (Managing to state this property precisely is actually a bit tricky, and the crux of a formal proof) Say there is some maximum frequency in the signal. Deconstruct the signal into sinusoids of $<=$ that frequency. Find a sampling frequency f that allows recovery of the highest-frequency sinusoid, sample the signal (thus sampling all the sinusoids) at that frequency, and reconstruct the original input from the recovered input frequencies (and phases and amplitudes).

This is a feasibility-proof, not an algorithm, but that's all we need to answer our formal question: Yes, we can reconstruct a *band-limited* signal (one with no frequencies higher than some maximum) *exactly* if we can reconstruct its highest-frequency sinusoid exactly.

Temporal Domain Argument: Imagine sampling a sine wave at its high and low points, that is twice per wavelength (at twice its frequency). If we know these are the high and low points, we know its amplitude, phase, and frequency: we're done. True, if our sampling is shifted along by 1/4 of a wavelength, sampling at the same rate, we see a nothing but zeroes: bad luck and no information about the signal. Also notice a higher-frequency sine can sneak in between the samples and not be noticed (Fig. 23).

BUT if you sample long enough at just a little faster than twice the frequency, AND you know that no higher frequencies are present (this is part of the trickiness) sooner or later you'll get samples at the maximum amplitude and you can figure out all you need to know. The *Nyquist frequency* is twice the maximum frequency of a band-limited function and we must sample more frequently than that to guarantee an error-free reconstruction (which can be accomplished by interpolating between samples with a special function). In practice with finite wave trains, 5 or 10 times the maximum frequency is a good practical sampling rate to shoot for.

Frequency Domain Argument: We use the property that the FT of the Dirac comb is another Dirac comb, and that the closer the time (space) domain peaks are together, the

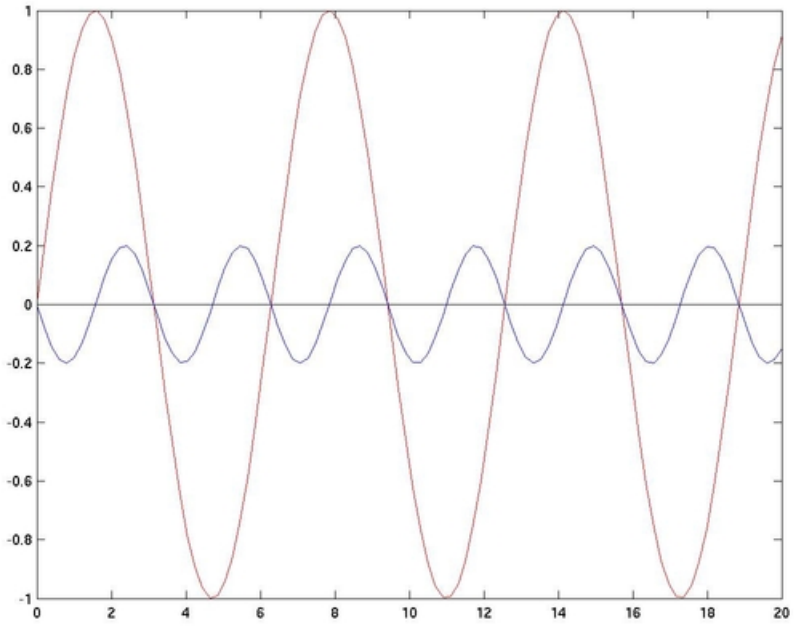


Figure 23: **The signal is the sum of the two sine waves. If we sample at twice the large (red) sine's frequency and at its high and low peaks, we can recover amplitude, phase, and frequency. The small (blue) sine is entirely missed, being 0 at every sampling location. Moral: we need to sample at least at twice the highest frequency, and we need an even higher frequency unless by luck sampling is in phase with peaks.**

farther apart the frequency domain peaks are. Well, multiplying a function $f(t)$ by the Dirac comb gives a sampled version of $f(t)$. Using the convolution theorem (Eq. (4)), and going to the transform domain, the transform of the sampled function is the convolution of $F(\nu)$ and the transformed Dirac comb. That is, it's copies of $F(\nu)$, spaced apart just as far as the peaks in the transformed Dirac comb. If those peaks are too close (the sampling Dirac comb was too coarse) the copies of $F(\nu)$ overlap and interfere with each other. If the peaks are far apart, the copies do not overlap and we can imaging snipping one out, inverse transforming it, and getting $f(t)$ back perfectly by inverse transforming one of the copies of $F(\nu)$ (Fig 24. The effects of overlap, or under-sampling, are called *aliasing*; the symptom is that higher-frequency waves are either missed entirely, or more likely are treated as (misleading) evidence of lower-frequency waves.

This subsection is meant to show the intellectual leverage we can already bring to bear on rather important signal-analysis questions. Here, we justified the universal and maybe questionable practice of representing a continuous function by a set of discrete samples. We discovered that for perfect reconstruction the function must be band-limited and the sampling must be done more often than the Nyquist frequency.

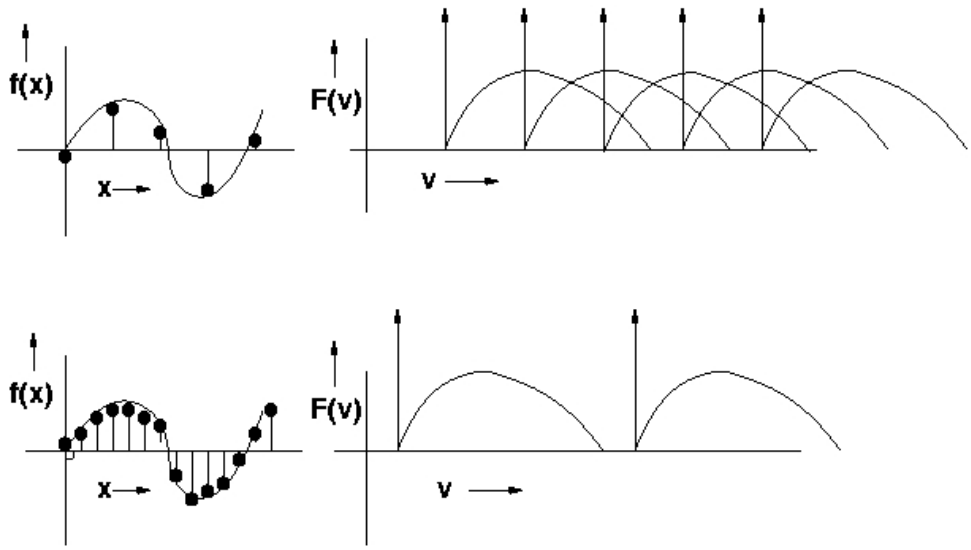


Figure 24: **On the left, sampling described as multiplication of $f(x)$ by a Dirac comb of low frequency (top) and high frequency (bottom). Fourier transforming both functions (band-limited $F(\nu)$ means the FT is finite) and applying the convolution theorem gives the FT of the product as the convolution of the FTs. The coarse (fine) Dirac comb gives a fine (coarse)-toothed transform. Here coarse (under-) sampling means the $F(\nu)$'s overlap and will add together and be impossible to sort out. Finer sampling (above the Nyquist frequency) means the $F(\nu)$s are separated and one can be "snipped out" and inverted to reconstruct $f(t)$ exactly.**

8 Three Frequency-domain Operations

8.1 Filtering

In this context, a *filter* (a very common use of the word in signal-processig) is an operation that does some signal-processing function. Filters can operate in the temporal or spatial domain (see Section 9).

Common types of filtering in the frequency domain are "band-pass" and "radial" filtering. In the first, some band of temporal or spatial frequencies is passed through and the rest are blocked. A low-pass filter throws away high frequencies (blurring the result) and a high-pass filter (usually a disk-like mask centered at the origin of frequency space) does the opposite, (sharpening the result) (Fig. 25.) A band-pass filter is the sum of a high-pass and a low-pass filter, allowing some band of frequencies to pass.

A radial filter looks like two more or less skinny pieces of pie symmetrically arranged around the (u, v) origin, masking frequencies in a range of directions. Of course radial and band-pass filters can be combined.

As Fig. 25 illustrates, filter shape is important: the fast change from 0 to 1 and back in the figure's sharp disk, or in 1-D the boxcar filter 0001111000, generate high frequencies. In turn they cause "ringing" in the results. There is an entire science of filter design, with

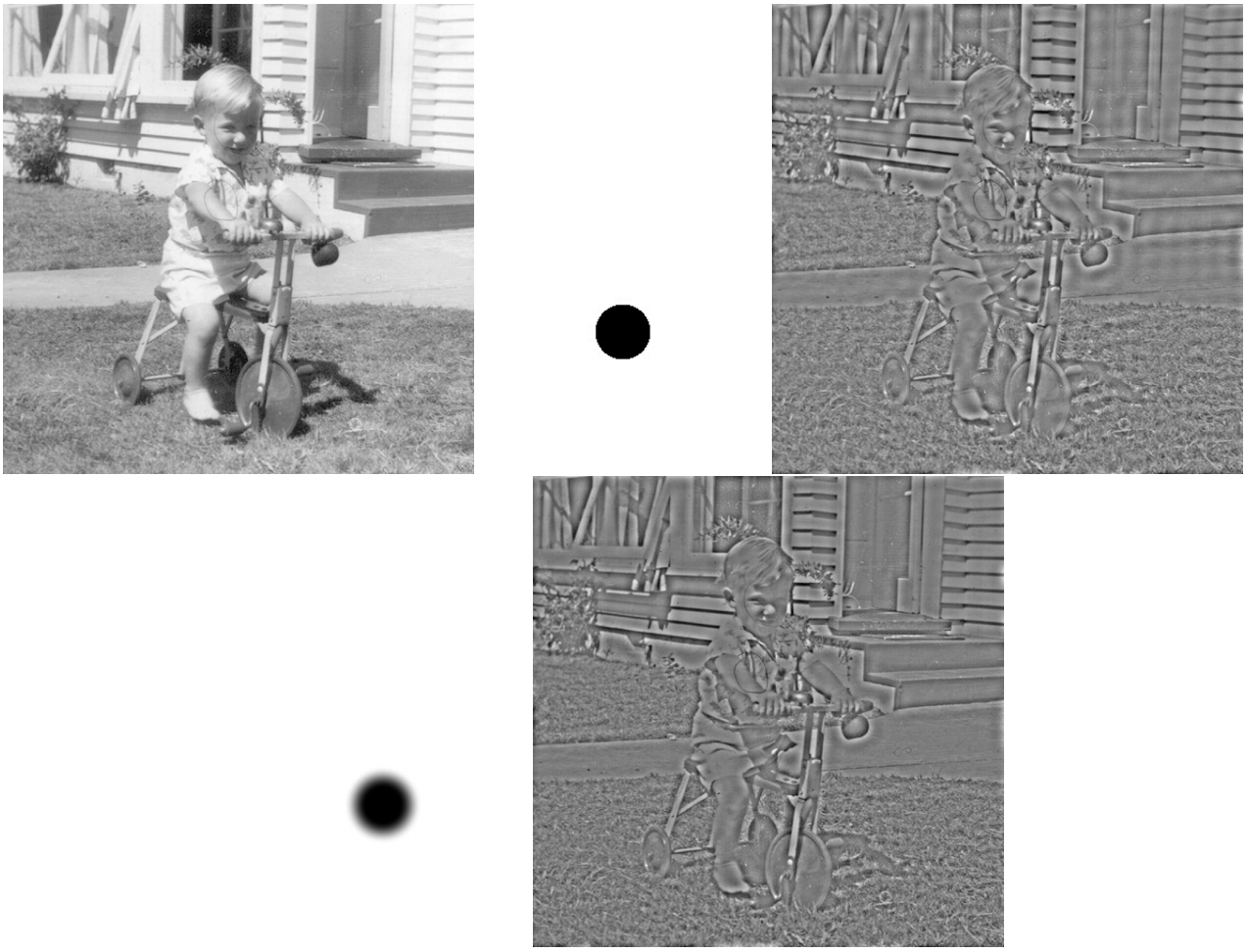


Figure 25: **Top: original image, sharp-edged high-pass filter centered at** $(u, v) = (0, 0)$**. The filtered result has ringing artifacts. Bottom: smooth-edged high-pass filter centered at** $(u, v) = (0, 0)$**: result is better.**

results like the Butterworth filter, which is designed to keep the band-pass range as flat (free of ringing) as possible. In matlab, `>> help butter` and follow the "doc butter" link.

Generally, if the FT of some unwanted signal is known, it can be masked out: problems arise because the signal you want often shares frequencies with the unwanted signal.

8.2 Matching

There is a whole science of recognizing known signals in noise (spawned when radar was being developed). Some signals are better than others for radar: a reflected sine wave or picket fence signal is ambiguous about range, an impulse signal needs a lot of power in a short time (and will be spread out when it returns). A "chirp" is a better choice, and a random string of 1's and -1's isn't bad (Fig 26).

The optimal way to detect a known signal is *correlation detection*, in which the input is

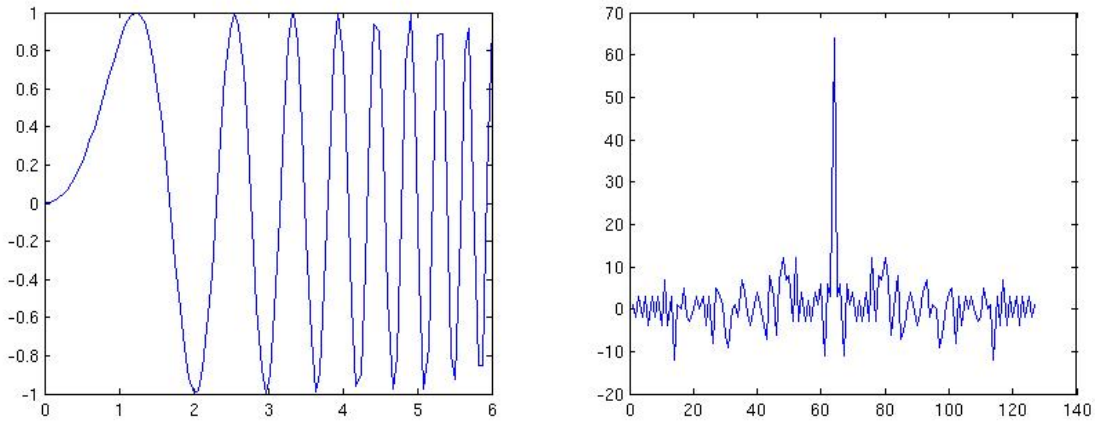


Figure 26: **A chirp (left) has good autocorrelation properties (a sharp peak). At right is the autocorrelation of a random string of 1's and -1's. In realizable functions, the sharp peak is flanked by "sidelobes".**

correlated with the known function (or convolved with its inverse). The correlation should show a peak when (or where) the known function lines up with its appearance in the input.

We met the *autocorrelation* of a function in Section 5.2. It is the output of correlation detection when the input is the known function with no noise. For instance: (remember *conv* is convolution, not correlation)
`conv([1 2 3 4],[4 3 2 1]) = [4 11 20 30 20 11 4]`. Using the time-reversal property of FTs and the convolution theorem, we get the elegant result that the power spectrum of a function is the Fourier transform of its autocorrelation.

Designing functions with good autocorrelation properties is interesting for imaging, and signal detection. "Computational imaging" cameras with strange "lenses" and built-in image-postprocessing are on the market. Another example is finding features in images to match in automatic photo-mosaicing: they must be places you can match reliably in another image – they are small areas with delta-function-like autocorrelations.

If the pattern to matched is small compared to the input, implementing correlation in the time or space domain (via the 1- or 2-D shifting dot-product method) is cheaper than FTing everything twice. Matlab has `conv(), conv2()` for such cases.

8.3 Image Reconstruction

> Within weeks of the launch of the [big, expensive, Hubble] telescope, the returned images showed that there was a serious problem with the optical system. Although the first images appeared to be sharper than ground-based images, the telescope failed to achieve a final sharp focus, and the best image quality obtained was drastically lower than expected. Images of point sources spread out over a radius of more than one arc-second, instead of having a point spread function (PSF) concentrated within a circle 0.1 arc-sec in diameter as had been specified in the

design criteria.[46] The detailed performance is shown in graphs from STScI illustrating the mis-figured PSFs compared to post-correction and ground-based PSFs.[47]

Analysis of the flawed images showed that the cause of the problem was that the primary mirror had been ground to the wrong shape. Although it was probably the most precisely figured mirror ever made, with variations from the prescribed curve of only 10 nanometers,[21] it was too flat at the edges by about 2200 nanometers (2.2 microns).[48] This difference was catastrophic, introducing severe spherical aberration, a flaw in which light reflecting off the edge of a mirror focuses on a different point from the light reflecting off its center.[49]

–Wikipedia

How do we repair in-camera optical distortion (blurring, say) of an image? The first ground-based, signal-processing fixes to the Hubble were based on the following idea, which starts with the convolution theorem statement:

$$\mathcal{F}\{f * h\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{h\}$$

Let's say the left hand side is the image from a camera with point-spread function h and scene data f. If h is an ideal Dirac delta, the output is the scene. If the camera moves during exposure, h becomes a 2-D curve (in general), and the image is spread out along this curve. Or, if the camera is not in focus its point spread function becomes a disk. The following argument is of course general, not just for images. Assume we know or can guess h.

In the equation above, simply divide (elementwise) by $\mathcal{F}\{h\}$:

$$\frac{\mathcal{F}\{f * h\}}{\mathcal{F}\{h\}} = \mathcal{F}\{f\},$$

so

$$f = \mathcal{F}^{-1}\left(\frac{\mathcal{F}\{f * h\}}{\mathcal{F}\{h\}}\right) \tag{6}$$

Here we have generated the FT of what we want (the input function) using things we know (the point-spread function and the degraded image). Taking the inverse transform of both sides recovers the original input. It almost works. A representative example is the box blur function, which formalizes either (in 2-D) defocus blur or (in 1-D) straight-line motion blur. Fig. 27 shows the problem: if $\mathcal{F}\{h\} \approx 0$, multiplying by $\frac{1}{\mathcal{F}\{h\}}$ multiplies frequencies there by huge amounts (or Inf if there is a divide by zero error). These amplified frequencies often contain a noise component, so the noise can overwhelm the signal. So some care (thresholding before multiplying, say) is needed. The Gaussian is a very friendly blur function since its FT is another Gaussian and so is always positive. Deconvolving with it simply amplifies the high frequencies, so thresholding is easy to implement and has predictable consequences (loss of fine detail).

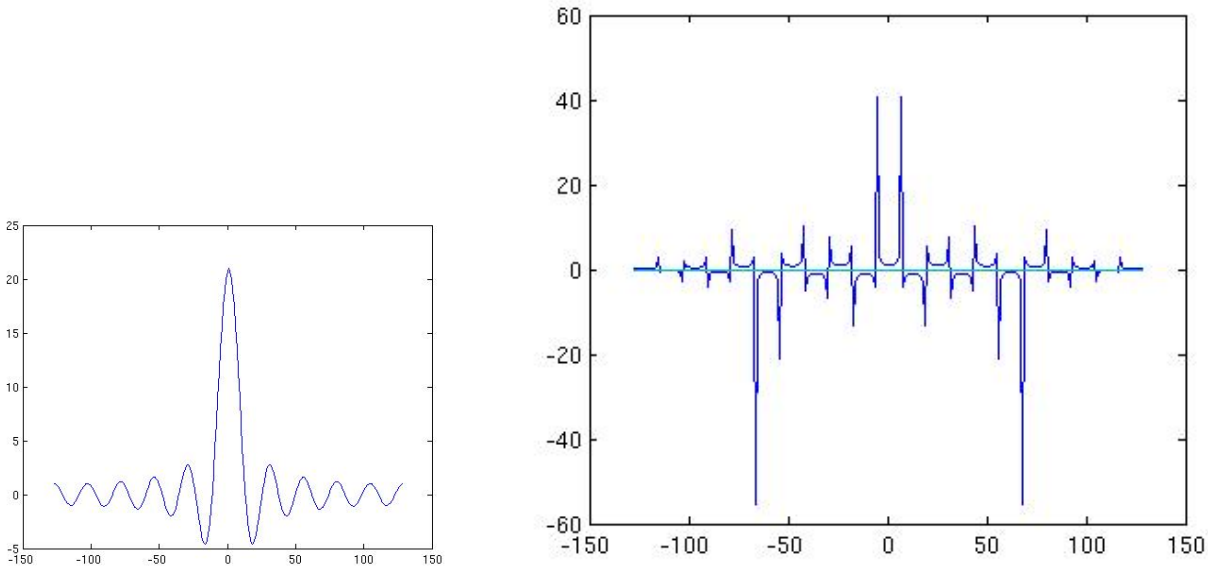


Figure 27: **The symmetric box function has a real-valued FT (on the left.) `box = [ 1 1 1 1 1 0 0 ....0 0 1 1 1 1]`: box(1) is considered the origin). 1/FT is plotted on the right: there are spikes or Infs where the FT is near or equal to 0.**

A summer (re)construction job In the crime-lab data (See Fig. 2), the PSF is the result of motion blur, and it be recovered from (that is, it exactly IS) the streak in the lower right, the image of a single bright pixel. Here are three digital versions of it: the true PSF and two approximations.

```
realpsf = [ .1 .5 .9 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 .9 .5 .1 ] % 22 long
okpsf = [ 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  ] %19 long
badpsf =  [ 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  ] %22 long
```

The results of deconvolving and un-transforming via Eq. (6) are shown in Fig 28.

Figure 28: **Deconvolving the motion-blurred image of Fig 2 with three possible streak point-spread functions (see text). Top left: badpsf; top right: okpsf (results serviceable but a little blurry still); Bottom: realpsf, the exact original point spread function.**

9 Time and Space-Domain Operations

While this tutorial is about the idea and use of transform spaces, many useful signal processing operations, in fact most, are performed in the time and space domains.

As an example, we might want to take the spatial derivative of the image to enhance edges. In a digital image, this amounts simply to subtracting one pixel's value from its neighbor's. We can do that with a convolution: we just convolve with the function (or mask, in Matlab terms) [-1, 1]. So the derivative is a linear operator (which we know, given the formula for $d/dx\ [a(f(x)) + b(g(x))]$.)

Due to one of our favorite properties of e^{at} , (its derivative is ae^{at}), it turns out that the result of putting the derivative operator into the FT is a cone-shaped function in frequency space rising from 0 at the origin and proportional to the frequency. The in frequency space, the derivative is a graphic equalizer that amplifies sinusoids by a factor proportional to their frequency. Makes sense: the derivative of slowly-varying waves is small, and of fast-varying ones is big.

Practically speaking, it is quicker and easier (takes fewer computer operations) to do this little convolution in the space domain than it does to transform, multiply, and un-transform image-sized matrices. Further, this filter like many can be implemented as a 'real time' operation rather than a 'batch' operation: d/dt can be done 'on the fly' by streaming the signal through a little circuit that subtracts neighboring data values and outputs the results. Likewise more complex filters like guitar effects are real-time circuits with a little memory.

As an experimentalist, you will be fitting mathematical models to experimental data, smoothing or interpolating data, creating statistics and visualizations. As we have seen, frequency space certainly has its uses, but unless frequencies are the issue you may never find yourself in transform space.

In sound processing, basic temporal-domain operations include overdubbing, sampling, splicing, amplitude modulation, "loop delays", and guitar effects like phasers and flangers.

Is spatial processing useful for images? Well, imagine inserting the Photoshop manual right here. So yes. Matlab has an image processing toolbox, with some very useful proven techniques, like *Mathematical morphology*, which is like convolution but the operations are non-linear set operations like AND and OR functions over the data (binary data is the easiest to think about). Morphology is excellent for cleaning up noisy images in many useful ways, and also can be used for matching.

Enough. There is much more to be learned and used, but I hope you have an inkling of why transform spaces are useful in many contexts, especially signal processing.

10 Acknowledgements

Prof. Randal Nelson did his best (within the bounds of friendship and without the possibility of a total rewrite) to minimize this document's mathematical howlers and misinformation. I'm definitely and directly responsible for all infelicities, misstatements, and mistakes. – CB.

11 Sources and References

Mathematical Morphology: Theory and Hardware (Oxford Series in Optical and Imaging Sciences); R. M. Haralick; Oxford U. Press 2010.

Digital Image Processing using Matlab; R.C. Gonzalez, R.E. Woods, S. L. Eddins; Pearson 2004.

Computer Vision and Image Processing: a practical approach using CVIPtools; S.E. Umbaugh, Prentice-Hall 1998

Digital Image Processing, 2nd Ed.; R.C. Gonzalez, R.E. Woods; Prentice-Hall 2002.

Digital Image Processing; K.R. Castleman; Prentice-Hall 1996.

Digital Image Processing,3rd Ed.; W.K. Pratt, Wiley 2001.

Traitement numerique des images; M. Kunt, G. Granlund, M. Kocher; Presses polytechniques et universitaires romandes 1993.

Brown, C.M., "Multiplex imaging with multiple pinhole cameras," *J. Appl. Physics 45*, 4 April 1974.

Brown, C.M., "An iterative improvement algorithm for coherent codes," *Optics Communications 33*, 3, 241-244, June 1980.

```
http://fourier.eng.hmc.edu/e101/lectures/Fourier_Analysis/Fourier_Analysis.h
good tutorial
http://sharp.bu.edu/~slehar/fourier/fourier.html
nice examples, good tutorial
http://mrl.nyu.edu/~dzorin/intro-graphics/handouts/filtering/node3.html
http://en.wikipedia.org/wiki/Phasor_%28sine_waves%29
http://en.wikipedia.org/wiki/Dirac_delta_function

http://terpconnect.umd.edu/~toh/spectrum/TOC.html
illustrated, browseable, 45-page essay on Signal processing in
chemical analysis, with code and examples.

http://local.wasp.uwa.edu.au/~pbourke/miscellaneous/imagefilter/

http://www1.cs.columbia.edu/CAVE/publications/pdfs/Nayar_TR11.pdf
Computational cameras, by a giant in the field.
```

Wikipedia: *Coma, Distortion, Convolution (animation), Phasor (animation), Dirac Delta Function, dot product, Euler's formula, etc.*

A Complex Numbers

Complex numbers are just mathematical objects, like integers, reals, and matrices, that have their own algebraic rules. A complex number is of the form $a + bi$, where i is $\sqrt{-1}$.

Especially in engineering, i is often called j. Complex numbers were controversial when introduced by Cardano back in the 1500's, and they still resist 'understanding' about what they 'really' are. Maybe there's a 'Metaphysics of Mathematics' course that worries about that, but it's easiest to think of them as Useful Gadgets That Do The Right Thing.

For instance, there isn't anything wrong with the equation $x^2 = 1$: the answer's $x = \pm 1$. By analogy, what's wrong with the equation $x^2 = -1$? Nothing, except you won't find an integer, fraction, real number, matrix, etc. that solves it. But it *is* solved by $x = \pm i$ if you define i such that $i^2 = -1$. And making such a definition turns out to be perfectly consistent mathematically, as well as useful. So think of it as a formal shorthand solution for a (mind-bending but) simple mathematical problem. You may recall that the solution to a simple quadratic equation $ax^2 + bx + c = 0$ is complex if $\sqrt{b^2 - 4ac} < 0$: in fact it is $((-b/2a) \pm (i/2a)\sqrt{b^2 - 4ac})$.

Or, if you're still queasy with $\sqrt{-1}$, you can just define a complex number as an (a, b) pair of real numbers, using the particular set of rules below to define operations on such pairs.

A complex number $a + bi$ can be considered to live in a plane with a real-part axis (giving value of a) and an imaginary-part axis (giving b's value). So it is a 2-vector in that 2-dimensional (real-imaginary) space called the *complex plane* (Fig. 29).

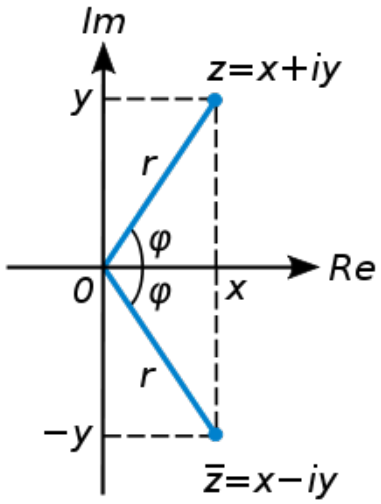


Figure 29: **An Argand diagram displays a complex number z and its conjugate $\bar{z}$ as points in the complex plane (the real-imaginary plane.)**

There's a new operator called the *conjugate* of a complex that simply changes the sign of its imaginary part. Confusingly for us, conjugation is often denoted by *, so we'll use $conj()$ as a conjugate function.

Thus $conj(a + bi) = (a - bi)$ and *vice-versa*. You can work out that $(a + bi) \cdot conj(a + bi)$ is $a^2 + b^2$, which is the squared magnitude (length) of the (a, b) vector in the complex plane.

Complex numbers are added, subtracted, multiplied, and divided by formally applying the associative, commutative and distributive laws of algebra, together with the equation $i^2 = 1$:

1. Addition: $(a + bi) + (c + di) = (a + c) + (b + d)i$
2. Subtraction: $(a + bi) - (c + di) = (a - c) + (b - d)i$
3. Multiplication: $(a + bi)(c + di) = ac + bci + adi + bdi^2 = (ac - bd) + (bc + ad)i$
4. Division:
$$\frac{a + bi}{c + di} = \frac{(a + bi)(c - di)}{c^2 + d^2} = \frac{(ac + bd) + (bc - ad)i}{c^2 + d^2}$$
where c and d are not both zero. Derive this by multiplying both the numerator and the denominator by the conjugate of the denominator $(c + di)$, which is $(c - di)$.

   It turns out that addition of complex numbers corresponds graphically to addition of the corresponding vectors in the complex plane. Somewhat more surprisingly, but also very useful, is the fact that if these vectors are represented by polar coordinates (r, θ), then multiplication of complex numbers corresponds to multiplying the lengths and adding the angles. That is, if $z_1 = (r_1, \theta_1)$ and $z_2 = (r_2, \theta_2)$ represent complex numbers in polar coordinates, then $z_3 = z_1 z_2$ is given by $(r_1 r_2, \theta_1 + theta_2)$. This leads to interesting graphical visualizations for operations such as extracting square roots, raising to powers, etc, as well as interesting facts such as the n nth roots of any complex number lying on a regular n-gon centered on the origin.

   Complex numbers are ubiquitous, so you've got to get used to them. As always, consult your favorite text or Wikipedia ('Complex Numbers') for more.

B Power Spectrum Examples

These are meant to highlight frequency-domain features and how they are related to their image (spatial domain) counterparts. The power spectrum has a wide range of values and must be manipulated in order to be shown. Here I zeroed out the central peak (always huge, representing the sum of all image values), added 1 to all the values, took their logarithm, then scaled the values to [0,1].

The vertical line in the Limbaugh power spectrum and the horizontal one in the beach sand are doubtless artifacts having to do with image cropping.

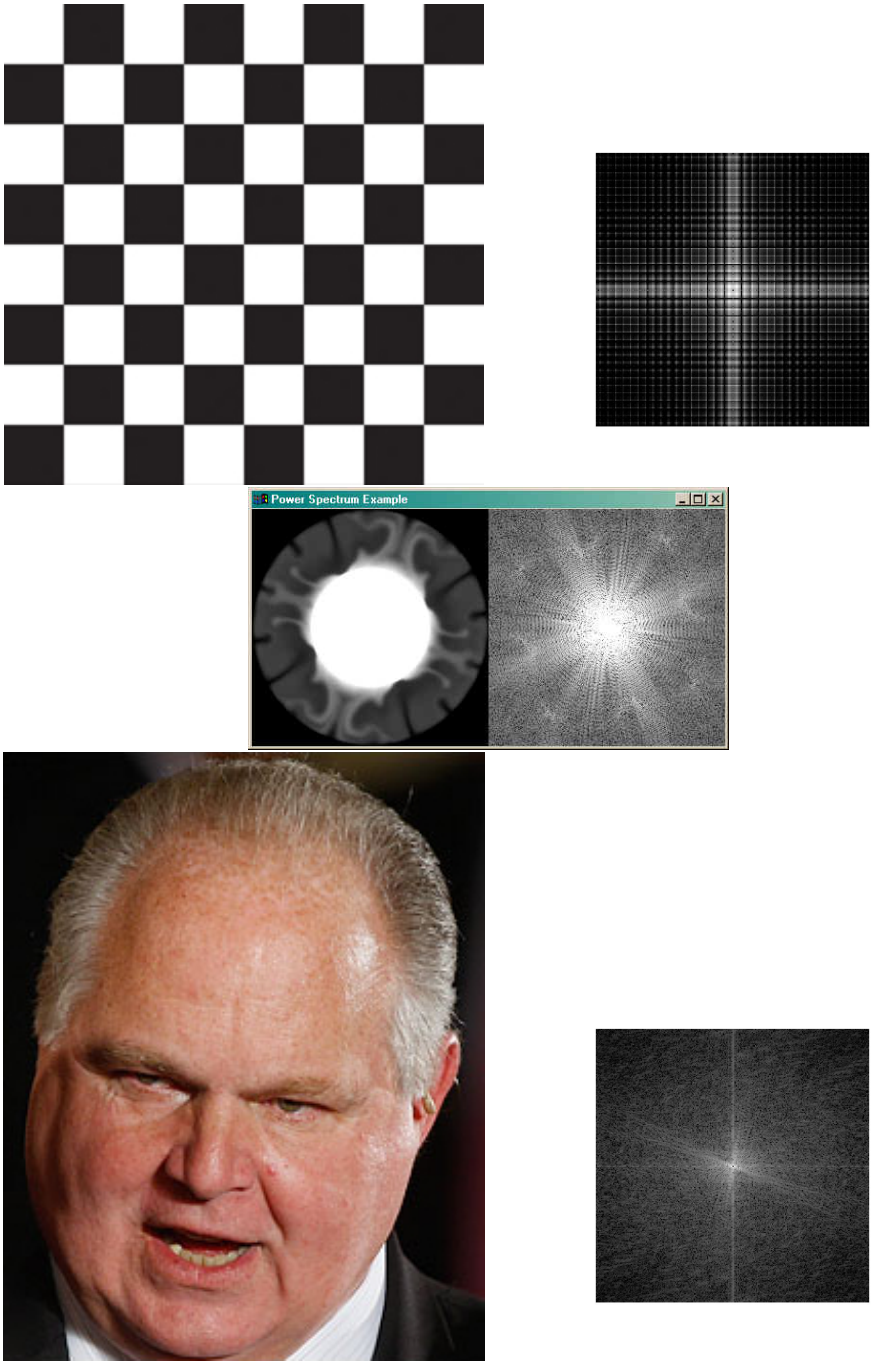


Figure 30: **Images and their log power spectra. The logarithm vastly reduces the range of the numbers, effectively emphasizing higher frequencies relative to lower. The checkerboard has s similarly-checkered spectrum, reminding us a bit of the 2-D dirac comb whose power spectrum is also a comb. The middle figure exhibits a few symmetries and some high-frequency structure. Rush's image has few straight lines but some high-frequency effects like hair.**

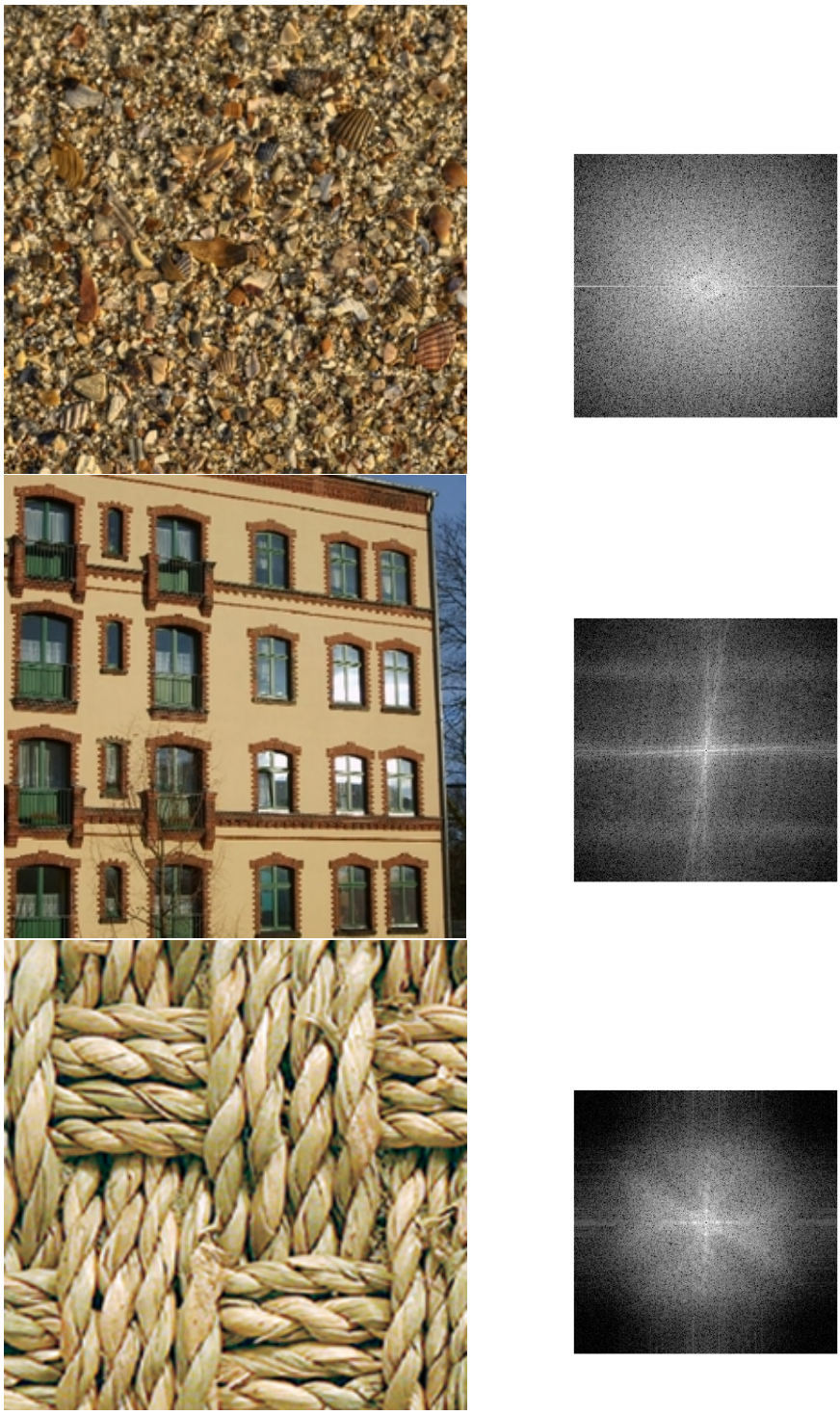


Figure 31: **Images and their log power spectra. We expect the sand image to have relatively much high-frequency power in no specific direction, and that the building and basket exhibit some straight line power spikes and 4-fold periodicity.**