Notes on Fourier Analysis

David J. Fleet and Allan D. Jepson

January 12, 2005

Fourier analysis plays a critical role in applied mathematics. In computer vision it plays a central role in the design and analysis of early operators. In the study of biological visual systems Fourier analysis is central to understanding of visual stimuli, to measuring input/output properties of neurons, and to the development of computational models.

In essence, the Fourier transform of an image is a decomposition of a signal into a weighted sum of sinusoidal signals. That is, the Fourier transform specifies, for each frequency, how much of a sinusoidal signal at that frequency exists in the signal. In discrete terms, it is simply an orthogonal matrix transform, i.e., a change of basis.

In vision, many of the image operations we employ are linear and shift-invariant. Sinusoidal signals, or Fourier basis functions, are eigen-functions of this class of operators which makes them a convenient basis set for design and analysis of linear filters. Fourier representations are also convenient for specifying various filter design constraints related to the scale and orientation of image information that we wish to enhance or attenuate. Moreover, Fourier theory provides a very nice starting point for the study of other image transforms such as the discrete cosine transform (DCT) that is used in JPEG compression, or wavelet transforms which have become popular in many contexts for the analysis and synthesis of signals at multiple scales.

In what follows we will first introduce the basic concepts of the Fourier transform with discrete signals. We'll come back to the filters and eigenfunctions later.

1 Discrete Fourier Transform (DFT)

Let $I[n]$ be a discrete signal of length N. For convenience, let $I[n]$ be a periodic signal with a period length of N, or equivalently, we can consider $I[n]$ to be cyclic, so that shifts are circular shifts. The central idea in Fourier analysis is to change the basis in which we represent the signal from a sequence of shifted delta functions (impulses) to a set of global sinusoidal signals, i.e., $s[n] = \sin(\omega n)$ where ω is the frequency of the sinusoid. Before introducing Fourier analysis, it is useful to review two important properties of discrete signals:

- First, the frequency ω is only unique between 0 and 2π. This is easy to see by noting that, because n is an integer, $\sin((\omega + 2\pi)n) = \sin(\omega n)$.
- Second, if we only consider periodic signals of length N, then we need only consider sinusoids which are periodic on the same domain. These have the form $\sin(\omega_k n)$ or $\cos(\omega_k n)$ for $\omega_k = 2\pi k/N$ with k an integer.
- Unique sinusoids exist only for N distinct frequencies $\omega_k = 2\pi k/N$, say for the integers k between 0 and $N-1$.

Finally, when working with sinusoidal signals, it's often very convenient to express then using complex exponentials. Remember Euler's formula:

$$e^{i\omega n} \;=\; \cos(\omega n) \;+\; i\;\sin(\omega n)$$

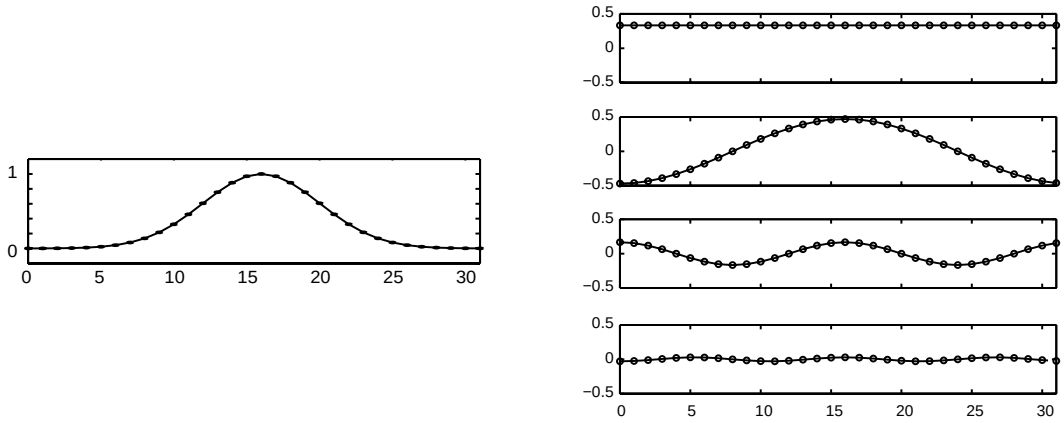

Figure 1: Here is a simple example of a Fourier decomposition. A Gaussian signal is shown on the left, and the first 4 terms of its Fourier decomposition are shown on the right.

---

where $i^2 = -1$. Conservely, with this one can write

$$\cos(\omega n) \;=\; \frac{1}{2}\left[e^{i\omega n} \,+\, e^{-i\omega n}\right], \;\; \sin(\omega n) \;=\; \frac{1}{2i}\left[e^{i\omega n} \,-\, e^{-i\omega n}\right].$$

1.1 Fourier Decomposition

The Fourier transform allows to write an arbitrary discrete signal $I[n]$ as a weighted sum of phase-shifted sinusoidal signals. Assuming that our signal $I[n]$ is really just N samples from a periodic signal (with period N), then we should only use periodic sinusoids in the sum that we use to express it. Therefore, the sum has the form

$$\begin{aligned} I[n] &= \sum_{k=0}^{N-1} \alpha_k \sin(\omega_k n + \phi_k) \\ &= \sum_{k=0}^{N-1} \alpha_k \sin(\phi_k)\cos(\omega_k n) \;+\; \alpha_k \cos(\phi_k)\sin(\omega_k n) \\ &= \sum_{k=0}^{N-1} a_k \cos(\omega_k n) \;+\; \sum_{k=0}^{N-1} b_k \sin(\omega_k n) \end{aligned} \tag{1}$$

Note that the sinusoidal signal $\sin(\omega_k n)$ is all zeros when $k = 0$, because when $k = 0$, then $\omega_0 = 0$ and therefore $\sin(\omega_0 n) = 0$ for all n. Therefore, normally one has the second summation only include terms for $k \geq 1$, but we'll keep all N terms for now.

1.2 Fourier Transform

So, now the question is: how do we get these coefficients a_k and b_k? Before answering this, let's rewrite equation (1) in a matrix form. Towards this end, imagine that our input signal $I[n]$ is written as an N-dimensional vector $\mathbf{I} = (I[0], ..., I(N-1))^T$. In addition, let $\mathbf{a} = (a_0, ..., a_{N-1})^T$ and $\mathbf{b} = (b_0, ..., b_{N-1})^T$ be N-vectors for the coefficients. Similarly, let's write the elementary sinusoidal signals as vectors, i.e. $\mathbf{S}_k = \sin(\omega_k n)$ and $\mathbf{C}_k = \cos(\omega_k n)$. That is, the j^{th} component of

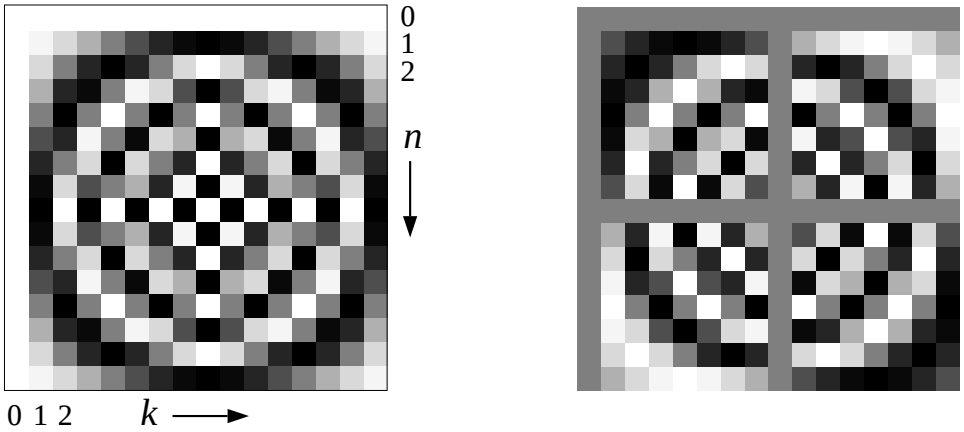

Figure 2: DFT matrices for the cosine and sine components of F for $N = 16$. (left) the cosine matrix $\mathbf{C}$ with frequencies $\omega_k = 2\pi k/16$ for k from 0 to 15 going from left to right, and n going from 0 to 15 from top to bottom. The first row and columns are all ones. (right) the sine matrix S with frequencies $\omega_k = 2\pi k/16$ for k from 0 to 15 going from left to right. In this case, the first row and column (and the 9^{th} row and column) are filled with zeros. Note the symmetry of $\mathbf{C}$ and $\mathbf{S}$, both across the main diagonal and also across the 9^{th} anti-diagonal (roughly from the bottom-left to the top-right).

---

the vector $\mathbf{C}_k$ would be $\cos(\frac{2\pi}{N}kj)$, for each j between 0 and $N-1$. Finally, collect these sinusoidal signals into $N \times N$ matrices, $\mathbf{C} = (\mathbf{C}_0\ \mathbf{C}_1 \ldots \mathbf{C}_{N-1})$ and $\mathbf{S} = (\mathbf{S}_0\ \mathbf{S}_1 \ldots \mathbf{S}_{N-1})$. Then, (1) becomes

$$\mathbf{I} \;=\; \sum_{k=0}^{N-1} a_k \mathbf{C}_k \;+\; \sum_{k=0}^{N-1} b_k \mathbf{S}_k \;=\; \mathbf{Ca} + \mathbf{Sb}. \tag{2}$$

We can rewrite this sum as a matrix equation:

$$\mathbf{I} \;=\; \begin{bmatrix} \mathbf{C} & \mathbf{S} \end{bmatrix} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} \tag{3}$$

To understand this, remember the way matrix multiplication works: The signal $\mathbf{I}$ on the left is a weighted sum of the columns of the $N \times 2N$ matrix $[C\ S]$, which is formed by concatenating C and S. The weights are coefficients in the vector on the right.

Remember that the $(n+1, k+1)$ element of C is $\cos(\frac{2\pi}{N}kn)$ where $k+1$ specifies the column index, and $n+1$ specifies the row. Similarly the corresponding element of S is $\sin(\frac{2\pi}{N}kn)$. Therefore the columns of C and S are sinusoidal basis vectors. Moreover, from these expressions it is clear that C and S are symmetric matrices (i.e. $C^T = C$ and $S^T = S$). Due to this symmetry, the rows of C and S are also formed from the same sinusoidal basis vectors, namely $\mathbf{C}_k$ and $\mathbf{S}_k$, for $k = 0, \ldots, N-1$.

Let's rewrite equation (3) as

$$\mathbf{I} \;=\; \mathbf{Fc} \tag{4}$$

where $\mathbf{F} = [C\ S]$ is the matrix above, the columns of which are the elementary sinusoidal signals, and the vector $\mathbf{c} = (\mathbf{a}^T, \mathbf{b}^T)^T$ contains the coeficients a_k and b_k as in (3). Note that $\mathbf{F}$ has $2N$

columns, each of which is of length N, so it is a $N \times 2N$ matrix. That is, the length of the signal $\mathbf{I}$ is only half of the number of coefficients $\mathbf{c}$ (i.e. N versus $2N$). Since there are more coefficients than signal sample values, the representation of $\mathbf{I}$ in terms of $\mathbf{c}$ is said to be **over-complete**. Due to this over-completeness, a basic result of linear algebra ensures us that any solution $\mathbf{c}$ of (4) is **not** unique. This non-uniqueness won't bother us here since we will pick one solution by convention.

Now, one way to determine suitable coefficients a_k and b_k is to find a $2N \times N$ matrix $\mathbf{G}$ such that when we form $\mathbf{c} = \mathbf{GI}$ it turns out that $\mathbf{c}$ satisfies (4). That is, for any $\mathbf{I}$ we have $\mathbf{c} = \mathbf{GI}$ such that

$$\mathbf{I} \;=\; \mathbf{Fc} \;=\; \mathbf{FGI}. \tag{5}$$

In other words, we need to find $\mathbf{G}$ such that $\mathbf{FG} = Id(N)$ where $Id(N)$ denotes an identity matrix of size N. Such a matrix $\mathbf{G}$ is called a pseudo-inverse of $\mathbf{F}$. If we find such a matrix $\mathbf{G}$, then computing suitable coefficients simply amounts to performing the matrix-vector product $\mathbf{c} = \mathbf{GI}$.

Our main result is that we can take $\mathbf{G} = \frac{1}{N}\mathbf{F}^T$. In particular, we show below that

$$\frac{1}{N}\mathbf{F}\mathbf{F}^T = Id(N). \tag{6}$$

In general, a linear transformation of a signal $\mathbf{I}$ of the form $\mathbf{c} = \mathbf{F}^T\mathbf{I}$ is said to be **self-inverting** if the signal can be reconstructed simply as $\mathbf{I} = \alpha\mathbf{Fc}$ for some constant α. The key property here is simply that, for a self-inverting transformation, a constant times the transpose of the transformation matrix serves as a pseudo-inverse of the transformation. Our main result can therefore be restated as, the discrete Fourier transformation matrix F^T is self-inverting.

We created $\mathbf{F}$ in (4) above so that its columns were the elementary sinusoidal signals $\mathbf{C}_k$ and $\mathbf{S}_k$. Therefore the rows of $\mathbf{F}^T$ are also these same sinusoidal signals. Furthermore, to find the coefficients a_k and b_k one simply multiplies the matrix $\mathbf{F}^T$ with the discrete signal $\mathbf{I}$, and divide the result by N. In effect, this amounts to taking the inner product of the signal $\mathbf{I}$ with each elementary sinusoidal signal (i.e. each column of $\mathbf{F}$). In practice, matrix-vector multiplication is relatively slow for full $2N \times N$ matrices such as $\mathbf{F}^T$. Matrix-vector multiplication requires $O(N^2)$ multiplications and additions, where N is the number of samples in the signal. By comparison, the fast Fourier transform (FFT) algorithm that is widely used requires only $O(N \log N)$ multiplications and additions. Note that for images, where the number of pixels can be as large as 10^6 or higher, the difference between $O(N^2)$ and $O(N \log N)$ is significant.

The set of coefficients, a_k, b_k, tells us "how much" of each frequency ω_k exists in our signal, and at what phase. The coefficients are called the Fourier transform of $I[n]$, and are often written as complex numbers $\hat{I}[k] = a_k - i\, b_k$ for convenience. Thus, the Fourier transform can be viewed as a function of frequency, again, specifying how much, and at what phase, of each frequency exists in our signal $I[n]$. In the continuous case, the Fourier transform is an explicit function of frequency, written $\hat{f}(\omega)$, while here, we write it as a function of the frequency index k, because k is an integer and ω_k is not.

With this notation, the usual way a discrete Fourier transform (DFT) is written is as follows:

$$\hat{I}[k] \;=\; \sum_n I[n]\, e^{-i\omega_k n} \tag{7}$$

$$I[n] \;=\; \frac{1}{N} \sum_k \hat{I}[k]\, e^{i\omega_k n} \tag{8}$$

To see the relationship between this formulation and the matrix equation above, remember that $e^{i\omega_k n} = \cos(\omega_k n) + i \sin(\omega_k n)$. The real part of the right hand sides in (7) and (8) therefore provide the equations above (except we have chosen to move the normalization term $1/N$ to the reconstruction equation, i.e. we use $\mathbf{c} = \mathbf{F}^T\mathbf{I}$ and $\mathbf{I} = (1/N)\mathbf{F}\mathbf{c}$).

1.3 Proof of the Self-Inverting Property

The complex form of the DFT in (7) and (8) is convenient for proving that the matrix $\mathbf{F}$ is self-inverting. In particular, let $\mathbf{U}$ be the $N \times N$ complex-valued matrix

$$\mathbf{U} = \mathbf{C} + i\mathbf{S}, \tag{9}$$

where $\mathbf{C}$ and $\mathbf{S}$ are as above (see Figure 1.2). Then the $(n+1, k+1)$ element of $\mathbf{U}$ equals $e^{i\frac{2\pi}{N}nk}$.

Consider the matrix $\mathbf{U}^*\mathbf{U}$, where $\mathbf{U}^*$ denotes the transpose of the complex conjugate of $\mathbf{U}$. Then from the previous expression for the elements of $\mathbf{U}$, it follows that the $(k+1, j+1)$ element of $\mathbf{U}^*\mathbf{U}$ is

$$\sum_{n=0}^{N-1} e^{-i\frac{2\pi}{N}kn} e^{i\frac{2\pi}{N}nj} \;=\; \sum_{n=0}^{N-1} e^{i\frac{2\pi}{N}(j-k)n} \;=\; N\delta_{j,k}. \tag{10}$$

In the last term above $\delta_{j,k}$ is the Kronecker delta, which is equal to one when $j = k$ and zero otherwise. This last equality is explained below.

For $j = k$ each of the terms in the sum in the middle term of (10) are $e^0 = 1$, so the sum is N for this case. Otherwise, suppose $j \neq k$, with $0 \leq j, k \leq N-1$. For such a pair j and k, define the constant $\theta = \frac{2\pi}{N}(j-k)$. Then the second sum above can be rewritten as

$$\sum_{n=0}^{N-1} e^{i\frac{2\pi}{N}(j-k)n} \;=\; \sum_{n=0}^{N-1} e^{i\theta n} \;=\; z \tag{11}$$

for some complex number z. But note that $\theta N = 2\pi(j-k)$, which is an integer multiple of 2π, and therefore $e^{i\theta N} = 1 = e^0$. Therefore, upon multiplying equation (11) by $e^{i\theta}$ we find

$$\begin{aligned} ze^{i\theta} &= \sum_{n=0}^{N-1} e^{i\theta(n+1)} \\ &= e^{i\theta N} + \sum_{n=1}^{N-1} e^{i\theta n} \\ &= 1 + \sum_{n=1}^{N-1} e^{i\theta n} \;=\; z. \end{aligned}$$

Thus $ze^{i\theta} = z$, so either $z = 0$ or $e^{i\theta} = 1$. But since $0 \leq j, k \leq N-1$ and $j \neq k$ it follows that $e^{i\theta} \neq 1$. Therefore $z = 0$, which completes the justification of equation (10).

We have therefore shown that $\mathbf{U}^*\mathbf{U} = N\,Id(N)$. By the definition of $\mathbf{U}$ in (9), and the symmetry of C and S, it follows that $\mathbf{U}^*\mathbf{U} = (C - iS)(C + iS) = C^2 + S^2$. Therefore we have shown that $C^2 + S^2 = N\,Id(N)$. Finally, notice that by definition $F = [C\ S]$, so $FF^T = C^2 + S^2$. As a consequence, $FF^T = N\,Id(N)$, proving that F is self-inverting.

1.4 Discrete Fourier Transform and Unitary Matrices

From the preceeding analysis, we can write the DFT in a third (and final) form. This last version of the DFT provides us with a simple intuitive model for the transform. By including the scale factor of $1/\sqrt{(}N)$ with $\mathbf{U}$, we can define the discrete Fourier transform of a signal $\mathbf{I}$ and its inverse transform as

$$\hat{\mathbf{I}} = \frac{1}{\sqrt{N}} U^* \mathbf{I}, \tag{12}$$

$$\mathbf{I} = \frac{1}{\sqrt{N}} U \hat{\mathbf{I}} \tag{13}$$

It follows from the previous section that $\frac{1}{\sqrt{N}}U$ is a unitary $N \times N$ matrix.

In this way, one can view the Fourier transform simply as an orthonormal change of basis. In more familiar terms, it involves only rotations and reflections of the original coordinates. The sinusoids in (3), or equivalently the complex exponentials in (13), can be viewed as a complete spanning set for the N-dimensional vector space of complex-valued signals. The k^{th} coefficients a_k and b_k can be viewed as the projection of $\mathbf{I}$ onto the vectors $\mathbf{C}_k$ and $\mathbf{S}_k$ in the spanning set. With real-valued inputs, the representation is overcomplete since there are $2N$ real-valued coefficients (ie. N complex-valued coefficients), but only N real numbers in the signal I. If I were complex-valued rather than real-valued, then the transform would be a complete representation, with just as many Fourier coefficients as input values. In fact, one can show that for real-valued inputs the Fourier domain is symmetric, and this accounts for the redundancy.

When we introduced Fourier analysis above, we restricted ourselves to signals that were periodic on N samples. This allowed us to consider only N frequencies. You may now ask, do we need to consider more frequencies in order to decompose a signal into a sum of sinusoids? The answer is no. The DFT is (modulo scaling) an orthogonal transform, and therefore we can completely represent and reconstruct any signal I with only N frequencies.

1.5 Examples

Let's say that the input is just a sinusoid of the form $I[n] = A\sin(\omega_3 n + \phi)$ where A is the amplitude. Using the orthogonality of the sinusoidal signals (i.e., (10)), the inner product of $I[n]$ with each row of F^T will be zero, except those rows containing $\mathbf{C}_3$ and $\mathbf{S}_3$. In other words, only the coefficients that correspond to the frequency in the signal, i.e., a_3 and b_3, are non-zero. Moreover, one can show that the phase ϕ is given by the arctan of b_3/a_3. And the magnitude of the signal, A, is equal to $\frac{1}{N}\sqrt{a_3^2 + b_3^2}$.

How about the case where the input is simply a delta function? For example let $I[n] = 0$ for all $n > 0$, and $I[0] = 1$. In this case, when one takes the product $\mathbf{c} = \mathbf{F}^T\mathbf{I}$, one can see that the resulting coefficient vector $\mathbf{c}$ is simply equal to the first column of $\mathbf{F}^T$. This column, as shown above, contains $\mathbf{C}_0$ on top of $\mathbf{S}_0$. Furthermore, when the frequency is zero, $\cos(0) = 1$ and $\sin(0) = 0$. Therefore, $a_k = 1$ for all k and $b_k = 0$. This is the well-known result that the Fourier transform of a delta function at the origin is constant. If we move the delta function to another location, then its Fourier transform will be a complex exponential.

1.6 Fourier Domain

Remember that our vector **I** is a representation of a signal $I[n]$, which is a function of the spatial variable n. We normally plot $I[n]$ as a function of spatial position n.

Similarly, it is common to plot the Fourier transform coeficients $\hat{I}[k] = a_k - i\,b_k$ as a function of frequency ω_k. So with frequency along the $x - axis$ we can plot the magnitude $|\hat{I}[k]|$ which is called the amplitude spectrum, and we can plot the phase angle $\arg[\hat{I}[k]]$ (i.e., atan2 for you C programmers), which is called the phase spectrum. We refer to functions of frequency, as functions in the frequency domain, where the independent variable is frequency.

When the magnitude (amplitude) of a particular Fourier coeficient $|\hat{I}[k]|$ is large, we say that there is a lot of power at frequency ω_k in the signal. The distribution of power, as a function of ω_k, tells us a lot about the properties of the signal.

2 Other Fourier Transforms

2.1 Discrete-Time Fourier Transform

As the length of the signal, N, increases toward infinity, the number of Fourier coefficients that we need to compute grows similarly. In the limit, although the signal is discrete, our sampling of frequencies between 0 and 2π becomes dense, so that the Fourier transform becomes a continuous function of frequency.

$$\hat{I}(\omega) \;=\; \sum_{n=-\infty}^{\infty} I[n]\, e^{-i\omega n}$$

for $0 \leq \omega < 2\pi$. The inverse transform, with which we reconstruct the signal is then given by

$$I[n] \;=\; \frac{1}{2\pi} \int_0^{2\pi} \hat{I}(\omega) e^{i\omega n} d\omega$$

This transform is used for a number of different purposes. If one had a discrete signal of finite length, one could in principle pad it with zeros out to infinity and take its DTFT. If the signal was an impulse response, then the DTFT would tell you how the filter behaves when applied to any frequency of interest. This is often very useful. It is also easy to show that the DFT is simply a sampled version of DTFT.

Finally, note that one can compute a good approximation to the DTFT without padding the signal with zeros. Rather, one simply has to add more rows to the DFT matrix at the frequencies that one is interested in.

2.1.1 Examples

How about some examples with smoothing filters like $h[n] = \frac{1}{4}[1, 2, 1]$? Well, for convenience, assume that we take the DTFT by padding $h[n]$ with zeros, and assume that $h[n]$ has it's nonzero samples centered at the origin. Then,

$$\begin{aligned} \hat{h}(\omega) \;&=\; \sum_{n=-\infty}^{\infty} h[n]\, e^{-i\omega n} \\ &=\; \sum_{n=-1}^{1} h[n]\, e^{-i\omega n} \end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{4}\left(e^{i\omega} + 1 + e^{-i\omega}\right) \\
&= \frac{1}{2}(1 + \cos(\omega))
\end{aligned}
$$

2.2 Fourier Transforms of Continuous Periodic Signals

Imagine now that we have a signal $I(x)$ that is defined at all spatial positions x on the real-line. As above with the discrete case, we'll also assume that $I(x)$ is periodic with period T; that is, $I(x+T) = I(x)$. This may be because $I(x)$ is actually periodic, or it may be that our signal of interest is of finite length T, and $I(x)$ is a periodic version of it. In either case, we expect all of the relevant elementary sinusoidal components of $I(x)$ to also be periodic with period T. Therefore, we will express the signal $I(x)$ as a weighted sum of sinusoidal signals with frequencies $\omega_k = 2\pi k/T$ where k is an arbitrary integer, as follows

$$
I(x) = \sum_{k=-\infty}^{\infty} \alpha_k e^{i\omega_k n} \tag{14}
$$

Because the signal is not discrete, it can contain sinusoids of arbitrarily high frequencies. Therefore the sum is infinite.

The coefficients in the sum are given by the inner product between the signal $I(x)$ and basis functions. The inner product is no longer a vector dot-product as above. It is now defined by an integral for continuous functions, over an interval of length T within which the signal is unique:

$$
\alpha_k = \frac{1}{T}\int_{-T/2}^{T/2} I(x) e^{-i\omega_k x} dx \tag{15}
$$

These coefficients α_k are the Fourier coefficients. They are often referred to as a Fourier series representation of $I(x)$.

Note that while the periodic signal $I(x)$ is continuous, defined everywhere on the real line, the Fourier transform is still a discrete signal. It is defined only at a discrete set of frequencies ω_k.

2.3 Fourier Transforms of Continuous Signals

Above we have discussed the Fourier transforms of discrete signals of bounded extent and of continuous signals of bounded extent. The most general case concerns a sufficiently smooth function with arbitrary extent defined on the real-line, for which we cannot assume periodicity. In this case, for sufficiently smooth signals $I(x)$, the Fourier transform $\hat{I}(\omega)$ is

$$
\begin{aligned}
\hat{I}(\omega) &= \int I(x) e^{-i\omega x} dx \\
I(x) &= \frac{1}{2\pi}\int \hat{I}(\omega) e^{i\omega x}.
\end{aligned}
$$

Now, both the input signal $I(x)$ and the Fourier transform $\hat{I}(\omega)$ are continuous functions defined everywhere on x (spatial position) and ω (frequency domain).

3 Multi-Dimensional Fourier Transforms

The same basic ideas hold in multple dimensions. In the continuous domain, we have

$$\begin{aligned} \hat{I}(\vec{\omega}) &= \int ... \int I(\mathbf{x}) \exp[-i\vec{\omega}^T\mathbf{x}] d\mathbf{x} \\ I(\mathbf{x}) &= \frac{1}{(2\pi)^n} \int ... \int \hat{I}(\vec{\omega}) \exp[i\vec{\omega}^T\mathbf{x}] d\vec{\omega}. \end{aligned}$$

where $\mathbf{x} = (x_1, ..., x_n)$ is the n-dimensional spatial position, $\vec{\omega} = (\omega_1, ..., \omega_n)$ denotes the corresponding frequency variables, and $\vec{\omega}^T\mathbf{x}$ denotes the usual dot product.

Note that if $I(\mathbf{x})$ is separable, then the multi-dimensional Fourier transform is the product of the 1d Fourier transforms. For example, if $I(x, y) = f(x)g(y)$, then

$$\begin{aligned} \hat{I}(\omega_x, \omega_y) &= \int\int I(x, y) e^{-i(x\omega_x + y\omega_y)} dx dy \\ &= \int f(x) e^{-ix\omega_x} dx \int g(y) e^{-iy\omega_y} dy \\ &= \hat{f}(\omega)\ \hat{g}(\omega) \end{aligned}$$

4 Properties of the Fourier Transform

In what follows we will list several important properties of the Fourier transform that we will use occasionally. For notational convenience, we will write the Fourier transform of $f(x)$ as $\hat{f}(\vec{\omega}) = \mathcal{F}[f(\vec{x})]$ where $\vec{x} = (x_1, ...x_n)$ and $\vec{\omega} = (\omega_1, ...\omega_n)$. Moreover, note that many of these properties are most straightforward to define in the continuous case.

- Shifting Property:

$$\mathcal{F}\left[f(\mathbf{x} - \mathbf{x}_0)\right] = exp(-i\,\vec{\omega}^T\mathbf{x}_0)\,\hat{f}(\vec{\omega}) \tag{16}$$

  In particular, note that $\mathcal{F}[\delta(\mathbf{x} - \mathbf{x}_0)] = exp(-i\,\vec{\omega}^T\mathbf{x}_0)$.
  You can prove this with substitution and change of variables.

- Modulation Property:

$$\mathcal{F}\left[exp(i\,\vec{\omega}_0^T\mathbf{x})\,f(\mathbf{x})\right] = \hat{f}(\vec{\omega} - \vec{\omega}_0) \tag{17}$$

  This is really identical to the shifting property, and can be proven in the same way, but with the Fourier domain and the spatial domain switched.

- Differentiation:

$$\mathcal{F}\left[\frac{\partial^n f(\mathbf{x})}{\partial {x_j}^n}\right] = (i\,\omega_j)^n\,\hat{f}(\vec{\omega}) \tag{18}$$

  This is a little tougher to prove, but give it a try. For intuition, note that $\frac{\partial \sin(\omega x)}{\partial x} = \omega\cos(\omega x)$.

  One can also use this fact, along with the convolution theorem below to show that the Fourier transform of the impulse response of a perfect n-th order differentiator is simply $(i\omega)^n$.

- Parseval's Theorem:

$$2\pi \; <f(\mathbf{x}),\, g(\mathbf{x})> \quad = \quad <\hat{f}(\vec{\omega}),\, \hat{g}(\vec{\omega})> \;, \tag{19}$$

where the inner product $<\cdot,\,\cdot>$ is defined by

$$<f(\mathbf{x}),\, g(\mathbf{x})> \quad = \quad \int_{-\infty}^{\infty} f(\mathbf{x})^*\, g(\mathbf{x})\, d\mathbf{x} \;. \tag{20}$$

Accordingly, $\| f(\mathbf{x}) \|^2 = <f(\mathbf{x}),\, f(\mathbf{x})>$. The proof relies on the fact that orthogonal transformations (rotations) do not change the lengths of vectors, and the Fourier transform is basically an orthogonal (unitary) transform.

- Convolution Theorem:

$$\mathcal{F}[f * g] \quad = \quad \mathcal{F}[f]\; \mathcal{F}[g] \tag{21}$$

Let's prove the Convolution Theorem in the discrete 1d case:

$$\begin{aligned} \mathcal{F}[f * g] \quad = \quad \sum_n f * g \; e^{-i\omega n} \quad &= \quad \sum_n \sum_m f[m] g[n-m] e^{-i\omega n} \\ &= \quad \sum_m f[m] \sum_n g[n-m] e^{-i\omega n} \\ &= \quad \sum_m f[m] \mathcal{F}[g] e^{-i\omega m} \qquad \text{(shift property)} \\ &= \quad \mathcal{F}[g]\mathcal{F}[f] \end{aligned} \tag{22}$$

This theorem is very important in practice. It means that one can apply filters very efficiently in the Fourier domain where convolution becomes multiplication. It is very common for filtering to be done in the Fourier domain!

This also helps show us what filters do. Given that we can decompose any signal into a sum of sinusoids, we can characterize what a filter does to any signal by characterizing what it does to sinusoidal signals. The amplitude spectrum of the filter's DFT tells us how each frequency in the signal is attentuated by the filter, and the filter's phase spectrum tells us how each sinusoidal component of the input will be phase shifted in the response. This is clear from viewing convolution as multiplication in the Fourier domain.

Also helps prove properties. For example, we can prove that

$$\frac{\partial}{\partial x}\,(h * g) \quad = \quad \frac{\partial h}{\partial x} * g \quad = \quad h * \frac{\partial g}{\partial x}$$

- Symmetries:
  - Real-valued signals have even-symmetric Fourier transforms: $\hat{f}(\omega) = \hat{f}^*(-\omega)$.
  - Even-symmetric signals have real-valued Fourier coefficients.
  - Odd-symmetric signals have purely imaginary Fourier coefficients.

Remember that the transform is just an orthogonal matrix, and therefore properties that relate one domain (space or Fourier) to the other will usually have a reciprocal property. Anyway, these symmetry properties are not tough to prove and we suggest doing it as an exercise.

- Reflections: The Fourier transform of $I[-n]$ is

$$\mathcal{F}[I[-n]] \;=\; \hat{I}[-\omega]$$

5 Another Perspective on Fourier Analysis

One of the reasons that sinusoids so important to linear, shift-invariant systems is that, when the input to such a filter is a sinusoid, then the output is also a sinusoid of the same frequency. Let's show that this is true. For now, let's consider discrete sinusoids, $I[n]$, on N samples with frequencies $\omega_k = 2\pi k/N$, and let the filter's impulse response be $h[n]$. The convolution equation is given by

$$\begin{aligned} R[n] &= \sum_{m=0}^{N-1} e^{i\,\omega_k (n-m)} h[m] \\ &= e^{i\,\omega_k\, n} \sum_{m=0}^{N-1} e^{-i\,\omega_k\, m} h[m] \end{aligned}$$

This shows that the output is equal to the input, multiplied by a complex-valued number that is equal to the inner product of $h[n]$ and $f_k[n] = e^{-i\,\omega_k\, n}$. In vector form we can write the inner product as $H_k = \vec{f}_k^{\,T}\vec{h}$. Then, we have $R[n] = I[n]H_k$. (You might also recognise H_k as the k^{th} Fourier coefficient of the DFT of h.)

Anyway, if we want to know what the filter does to all sinusoids of interest, then we need to know H_k for all $0 \le k < N$. We can also collect these values into a vector: $H[k] = \vec{f}_k^{\,T}\vec{h}$. In vector form this becomes a matrix equation,

$$\vec{H} = \mathbf{F}\,\vec{h}\;, \qquad \text{where} \quad \mathbf{F} = \begin{bmatrix} \vec{f}_0^{\,T} \\ \vdots \\ \vec{f}_{N-1}^{\,T} \end{bmatrix} \tag{23}$$

Finally, with some algebra one can show that $\mathbf{F}$ is a scaled unitary matrix; its inverse is given by

$$\mathbf{F}^{-1} \;=\; \frac{1}{N}\,\mathbf{F}^{*T} \;=\; \frac{1}{N}\,\left[\vec{f}_0^{\,*}, ..., \vec{f}_{N-1}^{\,*}\right]$$

where $\mathbf{F}^{*T}$ is often called the conjugate transpose of $\mathbf{F}$. (In fact, one can also show that $\mathbf{P}$ is symmetric as well, but we don't need this fact here). Anyway, if we multiply (23) on both sides by F^{-1}, we obtain

$$\vec{h} \;=\; \frac{1}{N}\,\mathbf{F}^{*T}\,\vec{H} \quad , \quad \text{or equivalently,} \quad h[n] \;=\; \frac{1}{N}\sum_{k=0}^{N-1} e^{i\,\omega_k\, n}\, H[k] \tag{24}$$

We've just derived the discrete Fourier transform and it's inverse; i.e. $\mathbf{F}$ is the DFT matrix, and $\vec{H}$ is the DFT of $\vec{h}$. Although we started with only periodic sinusoids on N samples, clearly we don't need to consider any more since the transform we have is invertible. Equation (24) shows that any signal $h[n]$ with N samples can be expressed as a sum of N complex-valued sinusoidal signals.

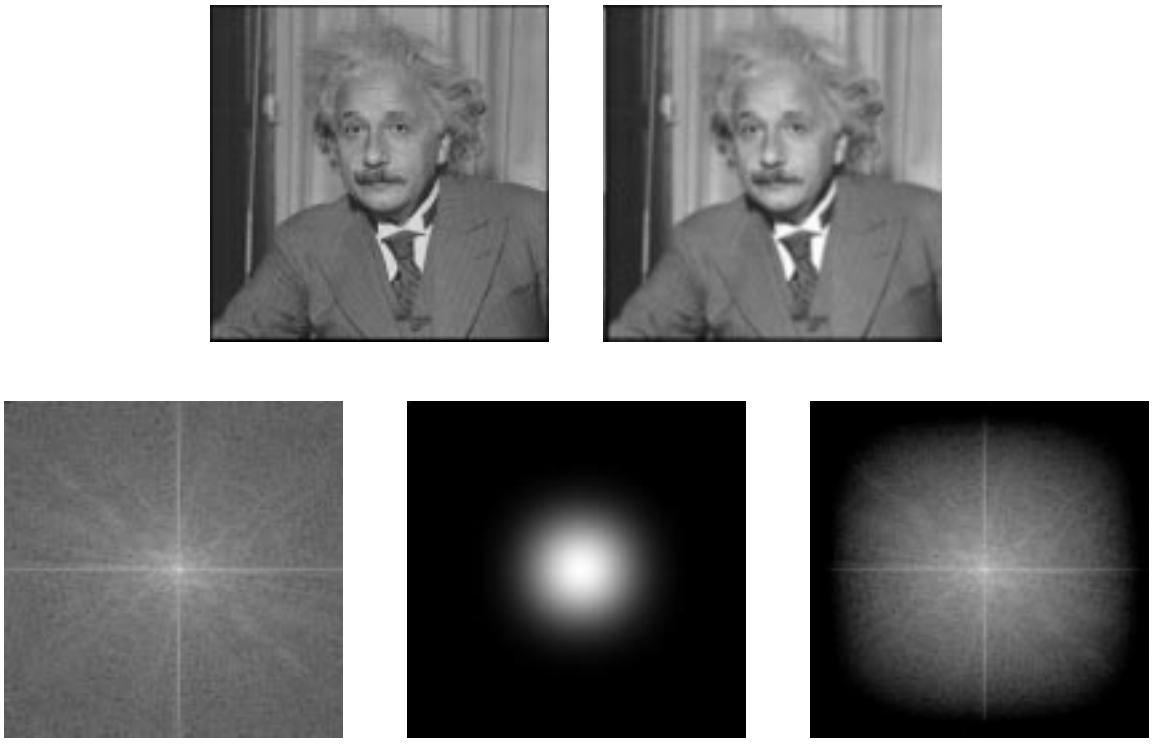

Figure 3: Blurring of Al. (top) the original image of Al and a blurred (low-pass) version. The blurring kernel was simple a separable kernel composed of the outer product of the 5-tap 1d impulse response $\frac{1}{16}(1, 4, 6, 4, 1)$. (bottom) From left to right are the log amplitude spectrum of Al, the amplitude spectrum of the impulse response, and the product of the two amplitude spectra, which is the inverse Fourier transform of the blurred version of Al.

Figure 4: From left to right is the original Al, a high-pass filtered version of Al, and the amplitude spectrum of the filter. This impulse response is defined by $\delta[n] - h[n, m]$ where $h[n, m]$ is the separable blurring kernel used in the previous figure.

Figure 5: From left to right is the original Al, a band-pass filtered version of Al, and the amplitude spectrum of the filter. This impulse response is defined by the difference of two low-pass filters.

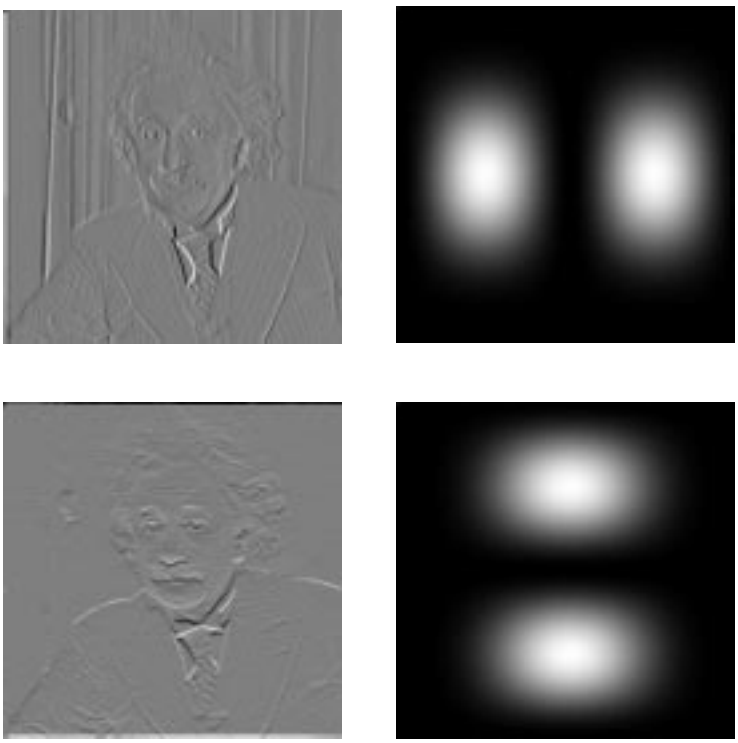

Figure 6: Derivative filters are common in image processing and computer vision. Here are crude approximations to a horizontal derivative and a vertical derivative, each of which is separable and composed of an outer product of a smoothing filter in one direction (i.e., $\frac{1}{4}(1,2,1)$) and a first-order central difference (i.e., $\frac{1}{2}(-1,0,1)$) in the other direction.

6 Design and Analysis of Linear Filters

6.1 Fourier Domain

There are several reasons that filters are often designed and analyzed in the Fourier domain. In general, they concern the ease with which one can constrain the *tuning* of the filters to scale, to orientation, and to velocity in the case of spatiotemporal filters.

We need to talk about a few things here, but we haven't done it yet.

- scale-specificity
- orientation/motion-specificity
- feature definitions (phase coincidence)

6.2 Classes of Filters

Broadly speaking there are three main types of filters, namely, low-pass filters, band-pass filters, and high-pass filters, all defined with respect to a frequency spectrum centered at the origin. In the discrete case, frequencies are defined between $-\pi$ and π in this case:

- Low-pass filters attentuate all frequencies above a cut-off frequency, thereby leaving significant power only at low frequencies. With the loss of high-frequency energy, the filter outputs have relatively poor spatial resolution, and look somewhat blurred. They are used to remove noise and to remove what might be considered irrelevant image detail depending on the task at hand.
- High-pass filters attentuate power at all frequencies below a certain cut-off frequency. One of the most important types of high-pass filter is the ideal differentiation filter. As explained above, one can infer from the differentiation property of Fourier transforms that the ideal differentiator is a high-pass filter.
- A band-pass filter is one that attentuates power at all frequencies below a certain cut-off frequency, and all frequencies above another cut-off frequency. This leaves a band of frequencies that the filter passes. The outputs of band-pass filters are generally restricted to a given range of scales. If the passband region is sufficiently narrow then the output of a band-pass filter will be expected to modulate at frequencies close to those at the center of the pass-band range.

Figures 3 - 6 show the application of filters that were discussed in the Linear Systems set of notes, along with their Fourier transforms.

6.3 Least-Squares Filter Design

(Modified from a handout written by D. Heeger, Stanford University.)

There are many ways to design discrete, linear filters (e.g., see Ch. 7 of Oppenheim and Schafer). Here, we derive a weighted least-squares design method. It is a very simple method that works well most of the time.

We want to design a real-valued filter $h[n]$ with a finite (hopefully very small) number of taps (nonzero samples) that has a desired Fourier spectrum, $H[k]$. For example, assume that $h[n]$ will have 5 taps, in which case its frequency response can be expressed as

$$H[k] = \sum_{n=-2}^{2} h[n]\, e^{-i\omega_k n} \ , \tag{25}$$

for $-2 \leq n \leq 2$, $0 \leq k \leq M-1$, and $\omega_k = 2\pi k/M$. Here, $H[k]$ is the frequency response of $h[n]$ assuming that $h[n]$ is the impulse response obtained with an impulse sequence of length M. This is equivalent to a sampled version of the DTFT with M samples.

Let's say that $\tilde{H}[k]$ is the desired frequency response. Our goal is to choose the filter taps, $h[n]$, to minimize:

$$\sum_{k=0}^{M-1} |H[k] - \tilde{H}[k]|^2.$$

In vector form, we want to minimize $||\vec{H} - \tilde{\vec{H}}||^2$.

Even-Symmetric Filters. First, let's consider a 5-tap, even symmetric filter; i.e.,

$$\begin{aligned} h_0 &= h[0] \\ h_1 &= h[1] = h[-1] \\ h_2 &= h[2] = h[-2], \end{aligned}$$

where $\vec{h} = (h_0, h_1, h_2)^T$ are the three distinct filter taps. There are only three distinct taps because we are enforcing even symmetry. The frequency response of this filter is obtained by writing out all the terms in Eq (25):

$$\begin{aligned} H[k] &= h[-1]\, \exp\left[i\,\omega_k\right] + h[1]\, \exp\left[-i\,\omega_k\right] \\ &\quad + h[-2]\, \exp\left[i\,2\omega_k\right] + h[2]\, \exp\left[-i\,2\omega_k\right] \\ &\quad + h[0]. \end{aligned}$$

Using the fact that $2\cos(x) = \exp(i\,x) + \exp(-i\,x)$,

$$H[k] = h_0 + 2h_1 \cos[\omega_k] + 2h_2 \cos[2\omega_k].$$

In vector form, we can therefore express $\vec{H}$ as

$$\vec{H} \ = \ \mathbf{C}\vec{h} \ , \tag{26}$$

where the columns of $\mathbf{C}$ are cosine basis vectors. The zeroth column corresponds to the signal $C_0[k] = 1$, the first column corresponds to $C_1[k] = 2\cos[\omega_k]$, and the second column corresponds to $C_2[k] = 2\cos[2\omega_k]$.

We can now rewrite the problem above as the minimization of

$$\|\mathbf{C}\vec{h} - \tilde{\vec{H}}\|^2 \ . \tag{27}$$

The least-squares (regression) solution is given by the usual formula:

$$\hat{\vec{h}} = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\tilde{\vec{H}}, \tag{28}$$

where $\mathbf{C}^T$ denotes the transpose of $\mathbf{C}$.

Odd-Symmetric Filters. We can use the same approach to design an odd symmetric filter. For a 5-tap odd symmetric filter the vector $\vec{h}$ is given by

$$\begin{aligned} h_0 &= 0 \\ h_1 &= -h[1] = h[-1] \\ h_2 &= -h[2] = h[-2]. \end{aligned}$$

The derivation is essentially the same except that you end up with sinusoids instead of cosinusoids in the columns of $\mathbf{C}$ because the frequency response of the filter is now given by:

$$H[k] = 2\,i\,h_1 \sin[\omega_k] \;+\; 2\,i\,h_2 \sin[2\omega_k].$$

Weighted Least-Squares. Often, we care more about some frequency components than others. For example, we might want to enforce that the filter have zero dc response. Or we might want to enforce that the frequency response be very small (or zero) for some other set of frequency components. In these cases, it is helpful to use a weighted least squares method. Use large weights for frequency components that you care a lot about and use small (or zero) weights for the other frequency components. Using weighted least squares, we want to choose $\vec{h}$ to minimize:

$$\sum_{k=0}^{M-1} (w[k])^2 (H[k] - \tilde{H}[k])^2,$$

where $w[k]$ are the weights. This can be written in matrix notation as follows:

$$\|\mathbf{A}\vec{h} - \vec{b}\|^2,$$

where $b[k] = w[k]\tilde{H}[k]$ is a weighted version of the desired frequency response. The columns of $\mathbf{A}$ are weighted versions of the (co)sine basis vectors (columns of $\mathbf{C}$). In particular, the jth column of $\mathbf{A}$ is given by: $A_j[k] = w[k]C_j[k]$. The solution (as above) is given by:

$$\vec{\hat{h}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\vec{b}.$$

Derivative Filters Many image processing algorithms depend on computing derivatives of a digital image: edge detectors (Laplacian zero crossings, gradient magnitude), steerable filters, motion estimation, depth from stereo, anisotropic diffusion. But derivatives are only defined for continuous functions of continuous variables, not for discretely-sampled and quantized signals. Often, people use simple differences between an adjacent pair of pixels to approximate the derivative. But one can do much better by designing a set of matched pairs of derivative filters and lowpass prefilters.

We conceive of the derivative operation (on a discrete signal) as performing three steps:

1. Reconstruct (interpolate) a continuous function from the discrete signal: $p(x) * a[n]$. Here $a[n]$ is a discrete signal, $p(x)$ is an interpolation filter (e.g., a sinc or some other low pass filter), and $*$ means convolution.

2. Take the derivative of the interpolated continuous signal: $\frac{\partial}{\partial x}(p(x) * a[n])$.

3. Sample the continuous derivative: $\mathcal{S}\left[\frac{\partial}{\partial x}(p(x) * a[n])\right]$, where $\mathcal{S}$ is the sampling operation.

Altogether, these three steps are the same as convolving with a discrete filter:

$$\mathcal{S}\left[\frac{\partial}{\partial x}(p(x) * a[n])\right] \;=\; \left\{\mathcal{S}\left[\frac{\partial}{\partial x}p(x)\right]\right\} * a[n],$$

where $d[n] = \mathcal{S}\left[\frac{\partial}{\partial x}p(x)\right]$ is a discrete filter kernel (the sampled derivative of a lowpass prefilter).

One could use an ideal lowpass (sinc) function for the prefilter, or a gentler function such as a Gaussian. But for many practical applications, we would like a relatively small filter kernel so we cannot use an ideal lowpass filter (which would have an infinite size kernel). On the other hand, the important thing for many applications is that we end up with a pair of signals, one which is the derivative of the other. A non-ideal interpolator will introduce some distortions, making it inappropriate to compare the original signal with its "derivative." This suggests that we should compute two convolution results: (1) the prefiltered original computed by convolving with the discrete prefilter $p[n]$, and (2) the derivative of the prefiltered original computed by convolving with the discrete derivative filter $d[n]$.

Now we wish to design a discrete prefilter $p[n] = \mathcal{S}[p(x)]$ and a discrete derivative filter $d[n] = \mathcal{S}[\frac{\partial}{\partial x}d(x)]$ so that the latter is the derivative of the former. In the frequency domain, we want:

$$D[k] \;=\; i\,\omega_k\,P[k]\;,$$

where $P[k]$ is the frequency response of $p[n]$, $D[k]$ is the frequency response of $d[n]$, and $0 \leq k \leq (M-1)$. This equation is based on the derivative property of the Fourier transform given above.

Using weighted least-squares, we want to minimize:

$$\sum_{k=0}^{M-1} (w[k])^2 (D[k] - i\,\omega_k P[k])^2.$$

This can be rewritten as:

$$\|\mathbf{A}\vec{d} - \mathbf{A}'\vec{p}\|^2,$$

where $\vec{p}$ is a vector containing the prefilter kernel, $\vec{d}$ is a vector containing the derivative kernel, $\mathbf{A}$ contains weighted versions of the Fourier basis functions as above, and $\mathbf{A}'$ is a similar matrix containing the Fourier basis functions multiplied by $i\omega_k$. After consolidating terms, we want to minimize

$$\|\mathbf{M}\vec{u}\|^2,$$

where

$$\mathbf{M} = (-\mathbf{A}' \mid \mathbf{A}) \quad \text{and} \quad \vec{u} = \begin{pmatrix} \vec{p} \\ \vec{d} \end{pmatrix}.$$

The solution $\hat{\vec{u}}$ is given by the eigenvector corresponding to the smallest eigenvalue of $\mathbf{M}^T\mathbf{M}$. Then the filters are both renormalized (by the same scale factor) so that $p[n]$ has unit dc response (i.e., the samples of $p[n]$ sum to one).

An example of a pair of 5-tap filters are:

$$p[n] = [0.035698,\ 0.24887,\ 0.43086,\ 0.24887,\ 0.035698]$$

$$d[n] = [0.10766,\ 0.28267,\ 0,\ -0.28267,\ -0.10766]$$

The frequency responses of these two filters are compared in Figure 7 .

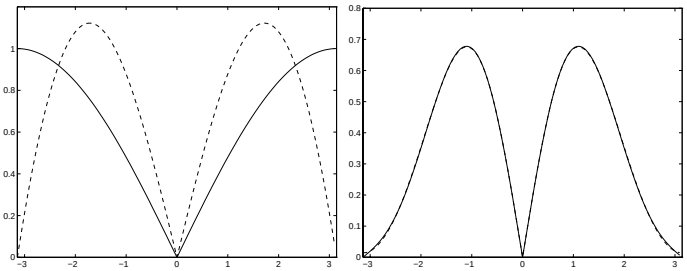

Figure 7: Frequency responses of 2-tap, finite difference (left) and 5-tap (right) derivative/prefilter pairs. Shown are the magnitude of the Fourier transforms of: a) the derivative kernel (dashed line), and b) the frequency-domain derivative of the prefilter (that is, its Fourier magnitude multiplied by $\omega = 2\pi(k/M)$.