

The use of signal transforms

A transform is simply an alternative way to view a signal. We've already encountered the Fourier and Z transforms, and there are many others. A transform is valuable only if it reveals properties of a signal or a system (such as a filter) that we could not observe directly from the signal itself. In these notes, we examine several transforms and their uses in DSP.

Multiresolution transform

One very simple, but useful, transform is the “sum-difference” transform. This works by taking sums and differences of every pair of signal values. Suppose we have the signal

$$x = [0, 1, 2, 3, 4, 5, 6, 7]$$

Then the sum-difference transform can be written

$$X = [(1, 1), (5, 1), (9, 1), (13, 1)]$$

Note that the first number in every pair in the transform is the corresponding sum of a pair of samples in the signal, and the second number is the difference. If we group the pairwise sums and differences, we obtain

$$X = [(1, 5, 9, 13), (1, 1, 1, 1)]$$

The pairwise sums tell us that the signal is increasing, while the pairwise differences tell us that the increments are by 1 at each sample.

Part of the appeal of the sum-difference transform is that it is repeatable. We start with the transform above, and repeat the process of sums and differences on the pairwise sums, producing

$$X = [((6, 22), (4, 4)), (1, 1, 1, 1)]$$

This shows that even the pair wise sums of the previous transform were themselves made up of constant increments.

The repeated sum-difference transform gives us a “multiresolution” description of the signal, which is a way of observing the signal at different scales or levels of detail. The image compression standard JPEG2000 is based on the wavelet

transform, which is a multiresolution transform similar to the sum-difference transform.

Fourier transform

The most useful transform in DSP is the Fourier transform. We've already seen a version of this, the Discrete Fourier Transform (DFT).

Suppose we have a signal $x[n]$, and compute its DFT

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi(k/N)n} \quad k=0,1,\dots,N-1 \quad \text{DFT}$$

The DFT allows us to observe the spectral content of signal at frequencies

$$F = 0, \frac{1}{N}, \frac{2}{N}, \dots, \frac{N-1}{N} \text{ cycles/sample}$$

As the number of samples N increase, the frequencies evaluated become closer together. In the limit as $N \rightarrow \infty$, we can write

$$X(F) = \sum_{n=-\infty}^{\infty} x[n] e^{-j2\pi F n} \quad \text{DTFT}$$

This is called the Discrete-time Fourier Transform (DTFT). Its properties are similar to the DFT, since it is essentially a limiting case of the DFT.

We know that the DFT has an inverse, the IDFT

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j2\pi(k/N)n} \quad \text{IDFT}$$

Similarly, the DTFT has an inverse. Since there is factor of N dividing the sum above, the result as $N \rightarrow \infty$ becomes an integral

$$x[n] = \int_{-0.5}^{0.5} X(F) e^{j2\pi F n} dF \quad \text{IDTFT}$$

Example

We've already seen how the inverse DTFT of the ideal lowpass filter is the sinc function. If F_c is the cutoff frequency, then

$$x[n] = \int_{-F_c}^{F_c} X(F) e^{j2\pi F n} dF = \frac{\sin(2\pi F_c n)}{(\pi n)} = 2F_c \text{sinc}(2F_c n)$$

One very useful property of the DTFT is the phase relationship to delay. Suppose that a signal $x[n]$ is delayed by d samples, i.e., $y[n] = x[n-d]$. Then

$$\begin{aligned} Y(F) &= \sum_{-\infty}^{\infty} y[n] e^{-j2\pi F n} = \sum_{-\infty}^{\infty} x[n-d] e^{-j2\pi F n} \\ &= \sum_{-\infty}^{\infty} x[n] e^{-j2\pi F(n+d)} = e^{-j2\pi F d} X(F) \end{aligned}$$

Therefore, a delay changes the transform only in the phase spectrum: the amplitude of the spectral distribution does not change, only the phase. The phase change is $-2\pi F d$, which is a linear function of frequency F . This is called a linear phase shift.

Input-output properties of a system

The DTFT is useful for examining the input-output properties of a linear system, such as a filter. We've already seen how the output $y[n]$ of a FIR filter is obtained by the rule

$$y[n] = \sum_{k=0}^M b_k x[n-k]$$

The filter coefficients b_k are also the impulse response of this filter. As the number of coefficients increases, we can write the limiting case as

$$y[n] = \sum_{k=-\infty}^{\infty} h[k] x[n-k]$$

This relationship covers not only the FIR case, but also the case of infinite-duration impulse response (IIR) filters.

The operation of multiplying a function $h[k]$ by the delayed signal $x[n-k]$ and summing is called **convolution**. We say that the output is the convolution of the input with the impulse response of the filter.

By applying the phase shift property we can prove this fundamental relationship between the input spectrum $X(F)$, the frequency response $H(F)$ of the filter, and the output spectral distribution $Y(F)$:

$$Y(F) = H(F) X(F)$$

We can use this relationship to our advantage in calculating the response of long FIR filter. If we have a signal $x[n]$ with N samples, then the output $y[n]$ of the FIR filter with $M+1$ coefficients (b_0, b_1, \dots, b_M) has $M+N$ values. By padding sufficiently many zeros, we can compute the $M+N$ point DFT of the input and the impulse response, multiply these together, and compute the inverse DFT of the result. Specifically, the operation is

$$y[n] = \text{IDFT}_{M+N} \left\{ \text{DFT}_{M+N} \{x[n]\} \times \text{DFT}_{M+N} \{h[n]\} \right\}$$

The DFT can be implemented using a FFT, in which the computation requires on the order of $(M+N) \log_2(M+N)$ operations. Since there are 3 FFT involved above, the total is $3(M+N) \log_2(M+N)$ operations, which compares to roughly $M(M+N)$ operations in the straightforward implementation. We see that the FFT method uses fewer operations when $M > 3 \log_2(M+N)$.

The frequency domain method can be even more attractive when the input signal is long. In this case, it can be broken up into segments of a convenient size N , and the results of each segment are added with overlap. Chapter 18 of Smith's book describes this approach called "overlap add".

One natural question to ask is whether the DFT can be used to implement *arbitrary* filters, by simply specifying $H[k]$ in the DFT domain. It would even seem possible to obtain filters with *zero* transition width! However, the DFT only specifies the response at a grid of frequencies spaced $1/N$ apart. In between those grid points, the response of the filter can oscillate substantially. This can be checked by taking the inverse DFT of the specified $H[k]$, and next taking the DFT with zero padding. See Chapter 17 of Smith for examples.

Continuous-time transforms

Fourier analysis started before there were sampled signals. For any continuous-time signal $x(t)$, the Fourier transform is defined by the integral

$$X_c(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi f t} dt \quad \mathbf{FT}$$

and the inverse Fourier transform by the integral

$$x(t) = \int_{-\infty}^{\infty} X_c(f) e^{j2\pi f t} df \quad \mathbf{IFT}$$

When the signal is sampled at a rate of f_s samples/sec, it can be shown that resulting samples $x[n]$ have a DTFT that is a result of aliasing and folding:

$$X(F) = \sum_{-\infty}^{\infty} x[n] e^{-j2\pi F n} = f_s \sum_{k=-\infty}^{\infty} X_c(Ff_s - kf_s)$$

If there is no folding or aliasing (the sampling rate is high enough), the DTFT is simply a frequency-scaled version of the FT: $X(F) = f_s X_c(Ff_s)$.

We can also connect the DFT to the FT. Since the DFT consist of samples of the DTFT (with the effects of windowing), we can write

$$X[k] \approx X(k/N) = f_s \sum_{n=-\infty}^{\infty} X_c((k/N)f_s - n f_s)$$

Or, if the sampling rate meets the Nyquist condition, $X[k] \approx f_s X_c((k/N)f_s)$. The approximation improves with the number of samples N . We've discussed the effects of windowing in our study of spectrum estimation.

Gibbs phenomenon

One important property of Fourier analysis, whether for continuous-time or for discrete-time, is that it ultimately relies on a sum of sinusoidal functions. Since each sinusoidal function is continuous, a sum of sinusoids can never exactly reproduce a discontinuity, no matter how many functions are used. This has important implications for FIR filter design, among many other aspects of DSP.

Consider an ideal lowpass filter, with cutoff frequency F_c . We know that its inverse DTFT (the impulse response that creates this filter) is the infinite-duration sinc() function. Suppose that we create a finite-duration approximation of this response, by defining

$$h_N[n] = 2F_c \text{sinc}(2F_c n), \text{ for } n = -N, \dots, N$$

$$h_N[n] = 0 \text{ otherwise.}$$

It would be reasonable to expect that as $N \rightarrow \infty$, the frequency response of h_N converges to the ideal lowpass filter.

However, that is not what happens! Figure 1 shows the frequency response for $N=500$, which is a large value. Notice that the overshoot and undershoot of the

response coming from the truncated filter is 9%, even when we are using a large number of coefficients. The MATLAB script “gibbsmovie.m” in the Appendix shows that the peak overshoot and undershoot do not diminish with N , though their location and the breadth of ripples diminish. This phenomenon, that the errors do not diminish in size with increasing N , is known as “Gibb's phenomenon”, after Josiah Willard Gibbs, a Yale physicist who explained it in 1889.

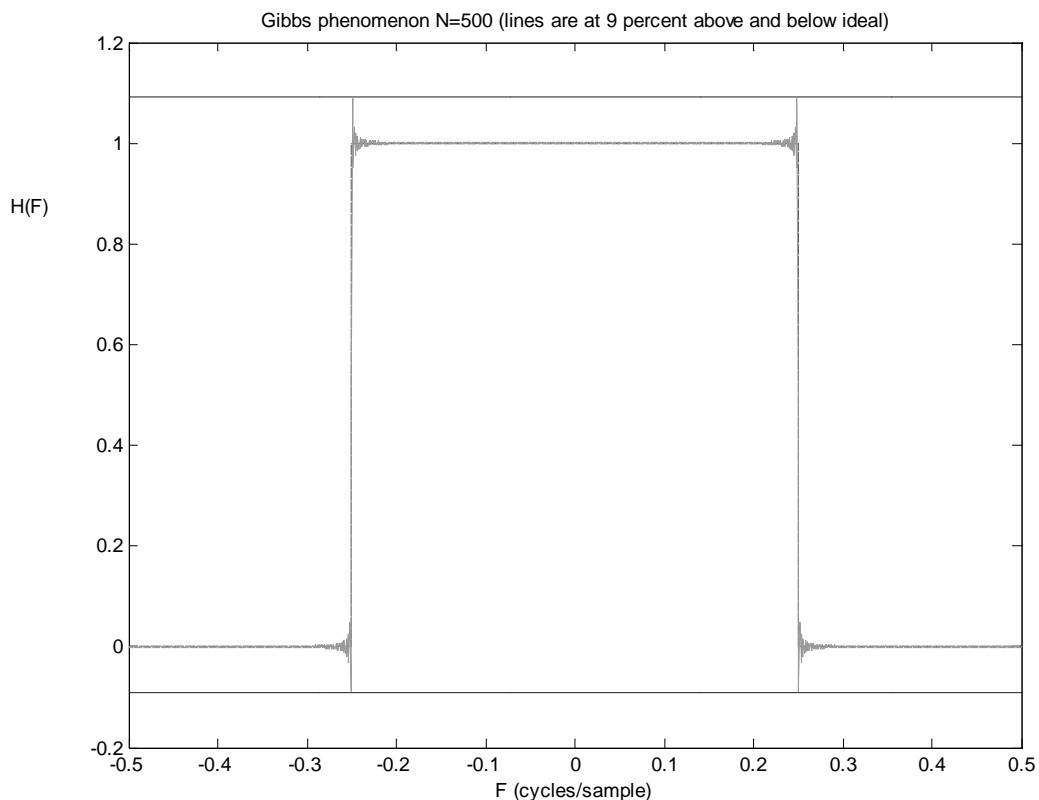


Figure 1: Illustration of Gibbs phenomenon, as occurs when approximating an ideal filter with a truncated sinc impulse response.

Transforms in abstraction

Finally, we should examine signal transforms from a mathematical point of view. If we represent any set of N samples as a column vector

$$x = [x(0), x(1), \dots, x(N-1)]^t,$$

then a transform can be viewed as a matrix operation:

$$X = \mathbf{A} x$$

The DFT can be realized this way, with a suitable \mathbf{A} matrix. Note that the inverse transform is simply

$$x = \mathbf{A}^{-1} X$$

Therefore, any invertible matrix is a transform!

A particularly useful matrix transform is the Discrete Cosine Transform (DCT). This transform uses only cosine functions, and produces real-valued outputs for any real signal, unlike the Fourier transform. However, it does not have the convolution property of the Fourier Transform, so therefore cannot be used for filtering. Despite this, the DCT is very useful for compression, due to their ability to represent typical image content well with small number of transform coefficients.

Appendix: gibbsmovie.m

```
%
%gibbsmovie; R. Kakarala, UCBX
%
Fc=0.25;           % take representative cutoff frequency.
F=-0.5:0.001:0.5; % take 1000 samples of frequency response
idealresp=cos(2*pi*F)>0; % define ideal response at every F
ninepercentplus=1.09*ones(size(F)); % nine percent above discontinuity
ninepercentminus=-0.09*ones(size(F)); % nine percent above discontinuity

for N=25:25:500 % partial sums
    H=zeros(size(F)); % initialize the DTFT at those points (to be
summed below)
    for n=-N:N
        H=H+2*Fc*sinc(2*Fc*n)*exp(-j*2*pi*F*n);
    end;
    plot(F,ninepercentplus,'b',F,ninepercentminus,
        'b',F,idealresp,'r--',F,real(H),'g');
        % plot four things on the same axis.
        % note that the imaginary part of H is
        % tiny, due to rounding error,
        % and can be neglected.

    title(sprintf('Gibbs phenomenon N=%d
(blue lines are at 9 percent above and below ideal)',N));
    pause(1.5);
end;
```