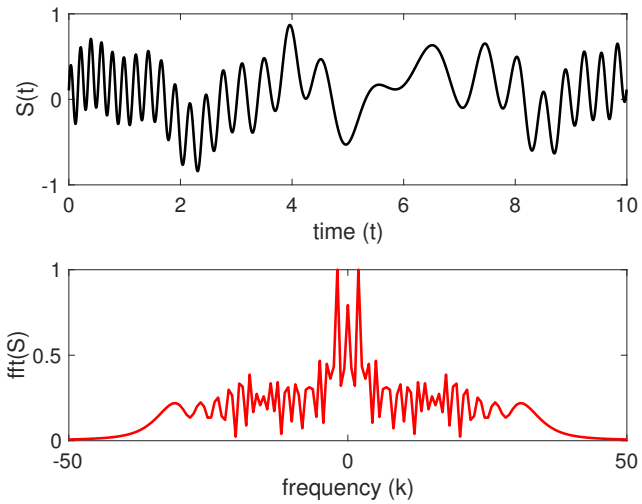
Lecture 4
Time-Frequency Analysis: Windowed Fourier Transforms

Jason J. Bramburger

In the previous lecture we saw that the Fourier transform can be used to analyze the frequency of a signal. The only drawback we saw was that the shift invariance of the absolute value of the Fourier transform loses information about what is happening in the time domain. Particularly, we lost *when* certain frequencies occur or how the frequencies change over time. Let's consider a signal that has higher frequencies at the beginning and end, but lower frequencies in the middle. The Fourier transform can tell us about what frequencies are present, but it can't tell use when there are higher frequencies and when there are lower frequencies.

Note: The Fourier transform eliminates time-domain information because you integrate out time to calculate it.

```
L = 10; n = 2048;
t2 = linspace(0,L,n+1); t = t2(1:n);
k = (2*pi/L)*[0:n/2-1 -n/2:-1];
ks = fftshift(k);

% Create signal
S = (3*sin(2*t) + 0.5*tanh(0.5*(t-3)) + 0.2*exp(-(t-4).^2) ...
    + 1.5*sin(5*t) + 4*cos(3*(t-6).^2))/10 + (t/20).^3;
St = fft(S);

% Plot signal in both time and frequency domain
figure(1)
subplot(2,1,1) % time domain
plot(t,S,'k','Linewidth',2)
set(gca,'Fontsize',16); xlabel('time (t)'); ylabel('S(t)')

subplot(2,1,2) % frequency domain
plot(ks,abs(fftshift(St))/max(abs(St)),'r','Linewidth',2); axis([-50 50 0 1])
set(gca,'Fontsize',16)
xlabel('frequency (k)'), ylabel('fft(S)')
```



Don't be misled here - the Fourier transform is still very useful for a wide range of applications. For example, in the radar detection example we were looking for some signal with a fixed frequency. Therefore, we could

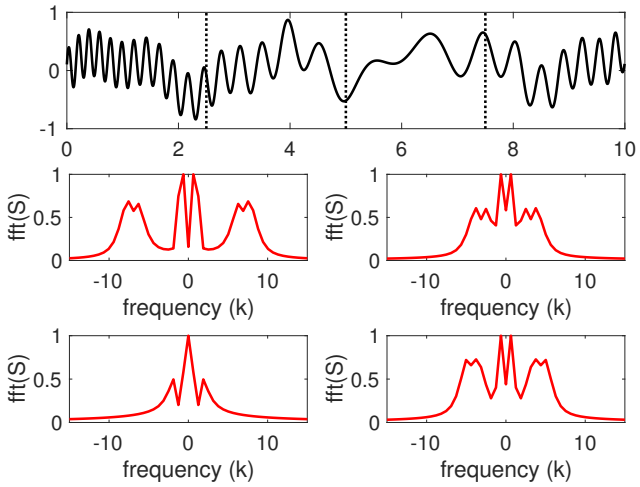
track the signal using different measurements in time. The Fourier transform really excels when characterizing **stationary** or periodic signals. A stationary signal is such that repeated measurements of the signal in time yield an average value that does not change in time. One example is a sine wave which produces the same frequency no matter which part you sample. Another is white noise, which does not give the same Fourier transform at different times, but will be the same on average. In practice, many signals (such as the one above) are non-stationary, thus necessitating another method.

Question: How can we get both time and frequency information from a signal?

To answer this question, let's start with a simple approach that will be a good motivator of the true approach that we will eventually get to. Let's split up the time domain into subdomains, and then take the Fourier transform on each subdomain. The hope here is that we can isolate the frequencies that are present in each subdomain to better understand when each frequency is present in the full signal.

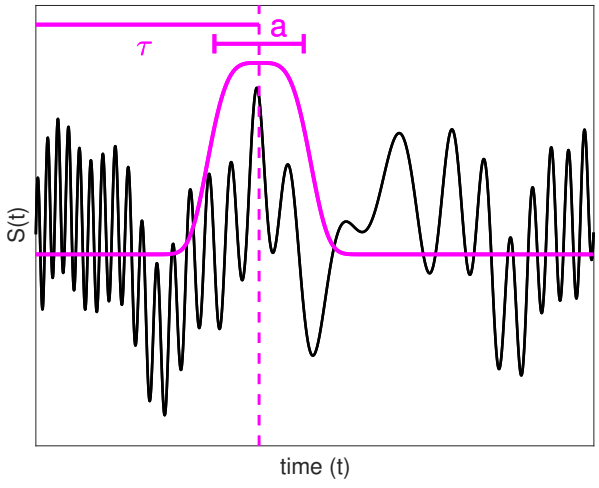
```
figure(2)
subplot(3,2,[1 2])
plot(t,S,'k','Linewidth',2)
hold on
for t_loc = [2.5,5,7.5]
    plot([t_loc t_loc],[-1,1],'k:','Linewidth',2)
end
set(gca,'Fontsize',16)

subplot(3,2,3)
k1 = (2*pi/L)*[0:n/8-1 -n/8:-1];
k1s = fftshift(k1);
S1 = S(1:n/4);
S1t = fft(S1);
plot(k1s,abs(fftshift(S1t))/max(abs(S1t)),'r','Linewidth',2); axis([-15 15 0 1])
set(gca,'Fontsize',16)
xlabel('frequency (k)'), ylabel('fft(S)')

subplot(3,2,4)
S2 = S(n/4+1:n/2);
S2t = fft(S2);
plot(k1s,abs(fftshift(S2t))/max(abs(S2t)),'r','Linewidth',2); axis([-15 15 0 1])
set(gca,'Fontsize',16)
xlabel('frequency (k)'), ylabel('fft(S)')

subplot(3,2,5)
S3 = S(n/2+1:3*n/4);
S3t = fft(S3);
plot(k1s,abs(fftshift(S3t))/max(abs(S3t)),'r','Linewidth',2); axis([-15 15 0 1])
set(gca,'Fontsize',16)
xlabel('frequency (k)'), ylabel('fft(S)')

subplot(3,2,6)
S4 = S(3*n/4+1:end);
S4t = fft(S4);
plot(k1s,abs(fftshift(S4t))/max(abs(S4t)),'r','Linewidth',2); axis([-15 15 0 1])
set(gca,'Fontsize',16)
xlabel('frequency (k)'), ylabel('fft(S)')
```

Notice that this method of breaking the signal into subdomains now captures the fact that there are higher frequencies in the beginning and lower frequencies in the middle. However, what we did here is fairly crude and doesn't work well in practice for a few reasons. First, the windows are fixed so we don't have any information about portions of the signal that span multiple windows. You could imagine (and should expect) the subdomains being chosen so that their boundaries don't properly delineate the transition from a high frequency to a low frequency. Hence, we would like to be able to consider all time windows of a given length. Second, there are sharp transitions between regions, which potentially can cause problems. This is because we are essentially assuming the signal



is zero outside of each subdomain. To circumvent these sharp transitions we can use a time-filter to pick out a window. This is just like the filtering we did in Lecture 2, but now applied to the time domain. Putting these two solutions to our problems together, we will slide the time filter across the domain and consider all possible locations for the filter.

```
figure(3)
plot(t,S,'k','Linewidth',2)
hold on
filter = exp(-(t-4).^4);
plot(t,filter,'m','Linewidth',3);
plot([0 4],[1.2 1.2],'m','Linewidth',3)
plot([3.2 4.8],[1.1 1.1],'m','Linewidth',3)
plot([3.2 3.2],[1.05 1.15],'m','Linewidth',3)
plot([4.8 4.8],[1.05 1.15],'m','Linewidth',3)
plot([4 4],[-1 2],'--m','Linewidth',2)
text(1.8,1.1,'\tau','Fontsize',24,'Color','m')
text(4.2,1.2,'a','Fontsize',24,'Color','m')
set(gca,'Fontsize',16), xlabel('time (t)'), ylabel('S(t)')
xticks([]), yticks([])
axis([0 10 -1 1.3])
```



The value τ describes where the filter is centred, so we can apply the Fourier transform by sweeping τ over the domain. The value a describes (approximately) the width of the window, or subdomain, that we are considering.

This forms the basis of the Gabor transform, or the short-time Fourier transform (STFT). Let's start with the mathematical background.

The Gabor Transform

Let us consider a filter function $g(t)$. Then, shifting by τ and multiplying by a function $f(t)$ represents the filtered function $f(t)g(t-\tau)$, with the filter centred at τ. The *Gabor transform*, or STFT, is then given precisely by

$$\tilde{f}_g(\tau, k) = \int_{-\infty}^{\infty} f(t)g(t-\tau)\mathrm{e}^{-\mathrm{i}kt}\mathrm{d}t.$$

That is, for a fixed τ the function $\tilde{f}_g(\tau, k)$ gives you information about the frequency components near time τ. We should note that your results are dependent on the choice of filter $g(t)$, hence the subscript g on $\tilde{f}$. There are many choices for g and some commonly used assumptions are:

1. The function g is real and symmetric.
2. $\|g\|_2 := \left(\int_{-\infty}^{\infty} |g(t)|^2 \mathrm{d}t \right)^{\frac{1}{2}} = 1$. That is, the L_2-norm of g is set to unity.

Note: the L_2-norm $\|\cdot\|_2$ is commonly understood to represent the total energy of a function.

The inverse of the Gabor transform is given by

$$f(t) = \frac{1}{2\pi\|g\|_2} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \tilde{f}_g(\tau, k)g(t-\tau)\mathrm{e}^{\mathrm{i}kt}\mathrm{d}k\mathrm{d}\tau.$$

You can see that assumption (2) above on g makes computing the inverse simpler since it's one less number to consider.

Since we saw above that the Gabor transform is dependent on your choice of the function g, a natural question arises: what should we take g to be? In theory, you can choose anything (assuming it satisfies the above assumptions), but when Gabor was developing this method he used a Gaussian. Therefore, a Gaussian will be our default choice as well. It is worth noting that some people refer to the transform with general $g(t)$ the STFT, while the Gabor transform is specific to having $g(t)$ being a Gaussian. I will not be this strict in my definition and use the terms Gabor transform and STFT interchangeably.

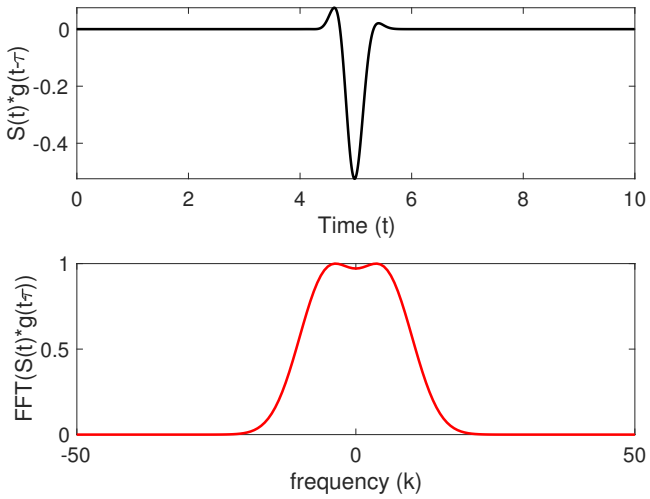
Width of the Gabor Window

Even with $g(t)$ specified to be a Gaussian function, we still have another parameter: the width of the window (denoted a in the figure above)[1]. If the window is huge, then we just recover the Fourier transform over the whole signal, which has all the frequency information, but no information about time. If the window is extremely small, then we are almost look at individual times along the signal, and so there would be no frequency information. These two limits represent obtaining either information about frequency only or time only, respectively. Recall that this lecture is on *time-frequency analysis*, meaning that we are trying to strike a balance. With a window with that is not too big and not too small, the Gabor transform returns some information about time and some information about frequency. Note that this is the Heisenberg uncertainty principle in action! There is a tradeoff here - you can't have all the information about both time and frequency. We compromise by getting a bit of information about both.

```
tau = 5; % centre of window
a = 10; % window size
g = exp(-a*(t-tau).^2); % Gaussian
Sf = g.*S;
Sft = fft(Sf);

figure(4)
```

[1]This issue arises for choices of $g(t)$ that are not Gaussian too.

```
subplot(2,1,1) % Time domain
plot(t,Sf,'k','Linewidth',2)
set(gca,'Fontsize',16), xlabel('Time (t)'), ylabel('S(t)*g(t-\tau)')

subplot(2,1,2) % Fourier domain
plot(ks,abs(fftshift(Sft))/max(abs(Sft)),'r','Linewidth',2); axis([-50 50 0 1])
set(gca,'Fontsize',16)
xlabel('frequency (k)'), ylabel('FFT(S(t)*g(t-\tau))')
```



Discrete Gabor Transform

As pointed out with Fourier transforms, if we actually want to use the Gabor transform on data we need a discrete version. The most important aspect of this is that we cannot shift the window by any arbitrary τ - it can only be taken in some discrete set of values. We also have to consider a discrete set of frequencies:

$$k = m\omega_0$$
$$\tau = nt_0$$

where m and n are integers and ω_0 and t_0 are positive constants (the frequency resolutions). The discrete Gabor transform is:

$$\tilde{f}_g(m,n) = \int_{-\infty}^{\infty} f(t)g(t-nt_0)\mathrm{e}^{2\pi i m\omega_0 t}\mathrm{d}t.$$

Note that if we want to reproduce the signal, we need a large enough window to have some overlap (or a small enough step t_0). Similarly, you need a small enough ω_0 to get good frequency resolution.