**Lecture 2**
**Radar Detection and Filtering**

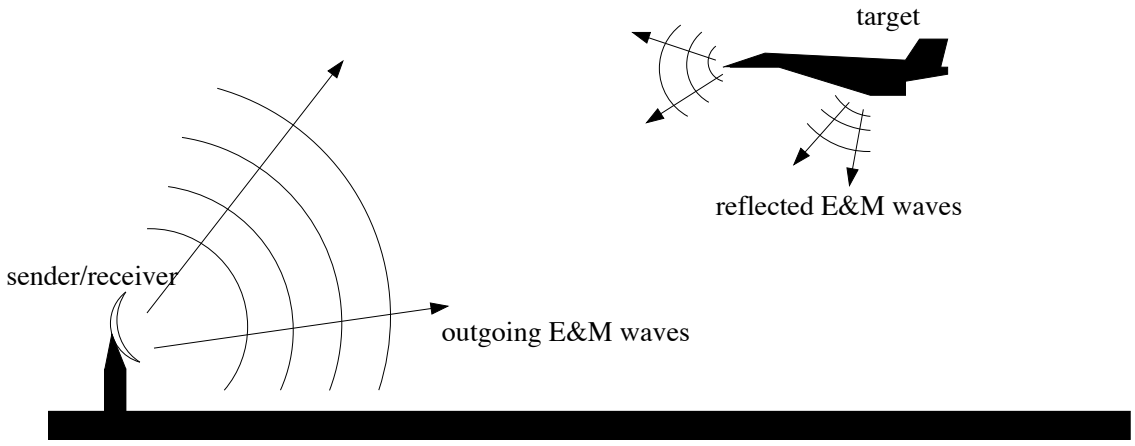
Jason J. Bramburger

---

**Important Fact:** The wider a function is, the thinner its Fourier transform is. Conversely, the thinner a function is the wider its Fourier transform is. You might have heard of this fact, because in quantum mechanics (QM) it is known as the *Heisenberg uncertainty principle*. That is, in QM a particle doesn't have a definite position. Instead, the position is represented by a wave function which gives the probability that the particle is in a given region. Therefore, if the function is thinner, there is a high probability that the particle is in a confined area near the peak of the function. Something similar happens when you try to describe the momentum of a particle, and it turns out that the position and momentum are Fourier transforms of each other. So, that means that the more you can localize the position of a particle, the less you know about the momentum (and vice versa).

In the next few lectures we will focus on signal processing - or more important time series data. So the function will be a function of time and the Fourier transform will be a function of frequency. What the above fact means for this case is that a short time pulse will not be very localized in frequency space and a signal that is localized to a specific frequency will be spread out across time. This leads to a bit of a tension between time information and frequency information about a signal.

Imagine a radar tower that wants to detect an aircraft:



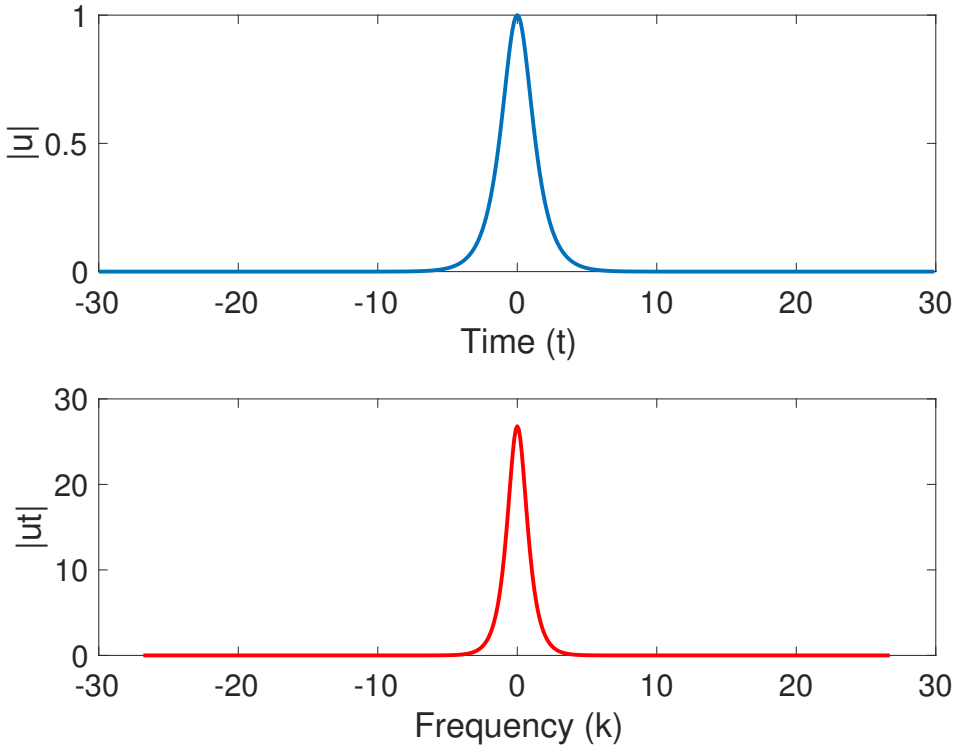

The radar tower sends out an electromagnetic signal at a certain frequency. If there is an aircraft, the signal reflects off of the aircraft and the reflected signal is received by the radar tower. We could imagine getting an electromagnetic pulse received back from the aircraft in the form of a hyperbolic secant. Here's the code to plot the function and its Fourier transform:

```
L = 30; % time slot to transform
n = 512;    % number of Fourier modes 2^9
t2 = linspace(-L,L,n+1);    % time discretization
t = t2(1:n);        % only use the first n points (periodicity)
k = (2*pi/(2*L))*[0:(n/2-1) -n/2:-1];   %frequency components
u = sech(t);    %ideal signal in the time domain

% Plot the time domain
```

```
subplot(2,1,1)
plot(t,u, 'Linewidth', 2)
set(gca, 'Fontsize', 16)
xlabel('Time (t)')
ylabel('|u|')

% Plot the frequency domain
ut = fft(u);
subplot(2,1,2)
plot(fftshift(k), fftshift(abs(ut)), 'r', 'Linewidth', 2)
set(gca, 'Fontsize', 16)
xlabel('Frequency (k)')
ylabel('|ut|')
```



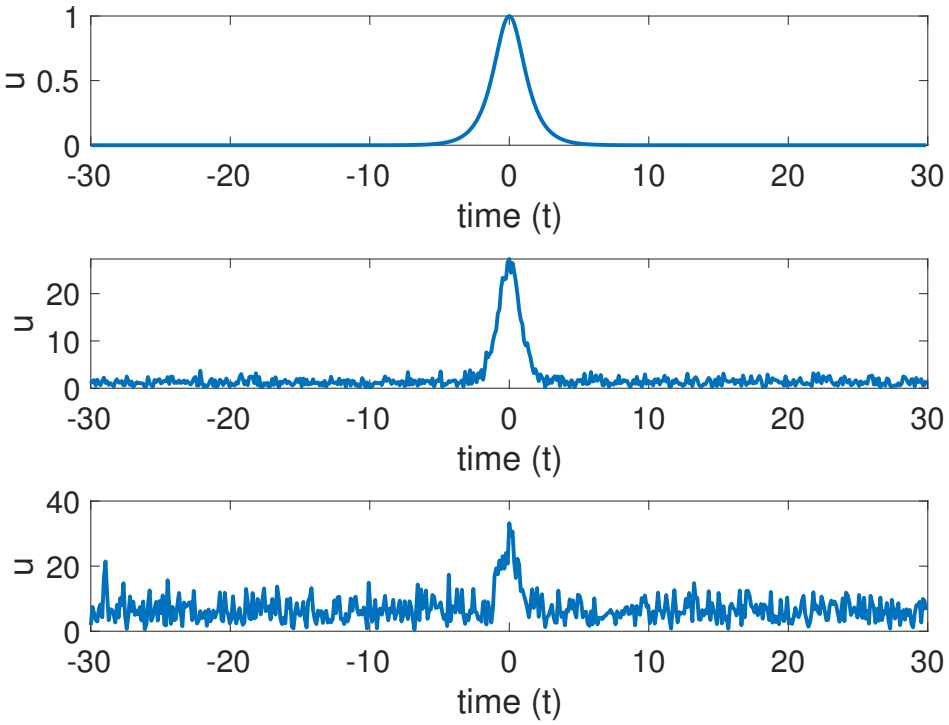

Signals are never this clean in the real world. For example, in addition to reflections coming from the desired target (the aircraft), there will also be reflections from geographical objects (mountains, trees, etc.) and atmospheric phenomena (clouds, precipitation, etc.). There might even be other sources of electromagnetic energy at the desired frequency. This results in a *noisy signal*. Typically this noise is **white noise**, noise that affects all frequencies the same. In order to add white noise to our idealized MATLAB signal, we can add a (complex) random number to each Fourier coefficient, drawing the random values from the normal distribution with mean 0 and standard deviation 1.

```
% Plot ideal signal
subplot(3,1,1)
plot(t,u,'Linewidth',2)
xlabel('time (t)')
ylabel('u')
set(gca,'Fontsize',16)

% Plot noisy signal (noise amplitude = 1)
noise = 1;
% Add white noise to the signal in frequency domain
utn = ut + noise*(normrnd(0,1,1,n) + 1i*normrnd(0,1,1,n));
```

```
un = ifftshift(utn); % Noisy signal in time domain
subplot(3,1,2)
plot(t,abs(un),'Linewidth',2)
xlabel('time (t)')
ylabel('u')
set(gca,'Fontsize',16)

% Plot noisy signal (noise amplitude = 5)
noise = 5;
% Add white noise to the signal in frequency domain
utn = ut + noise*(rand(1,n) + 1i*rand(1,n));
un = ifftshift(utn); % Noisy signal in time domain
subplot(3,1,3)
plot(t,abs(un),'Linewidth',2)
xlabel('time (t)')
ylabel('u')
set(gca,'Fontsize',16)
```



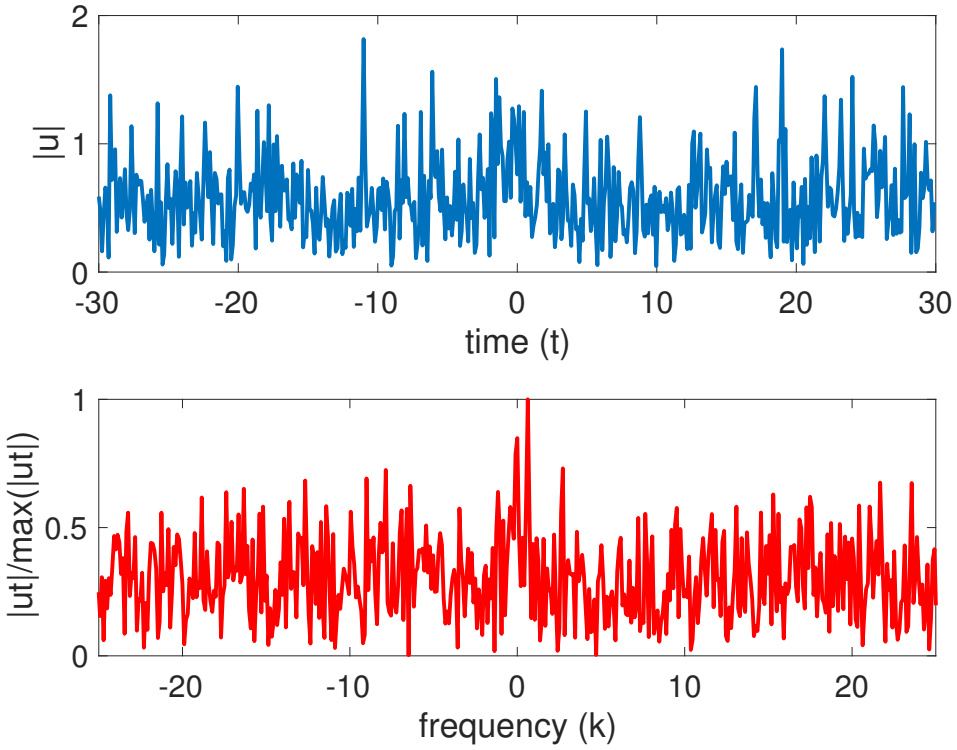

When the noise level is relatively low (i.e. high signal-to-noise ratio), you can still clearly detect the presence of the signal. However, when the noise is large, the signal is difficult to detect.

```
noise = 10;
utn = ut + noise*(normrnd(0,1,1,n) + 1i*normrnd(0,1,1,n));
un = ifft(utn);

% Plot the noisy signal
subplot(2,1,1)
plot(t,abs(un),'Linewidth',2)
axis([-30 30 0 2])
xlabel('time (t)')
ylabel('|u|')
set(gca,'Fontsize',16)

% Plot the Fourier transform
```

```
subplot(2,1,2)
plot(fftshift(k), abs(fftshift(utn))/max(abs(fftshift(utn))),'r','Linewidth',2)
axis([-25 25 0 1])
xlabel('frequency (k)')
ylabel('|ut|/max(|ut|)')
set(gca,'Fontsize',16)
```




Imagine receiving this signal: could you detect whether the pulse from the aircraft was present or whether it was all noise? Personally, I would have a hard time with this. Therefore, we want to try to **filter** out the noise in order to better detect the signal. In our radar example, the frequency of the signal is known. We can use a spectral filter around this frequency to try to remove undesired frequencies and much of the white noise. Mathematically, a filter is just a function that we will multiply the signal by (in the frequency domain). There are tons of options for filters, but let's use a simple one:

$$F(k) = \mathrm{e}^{-\tau(k-k_0)^2}.$$

This is a Gaussian function with $\tau$ determining the width of the filter and the constant $k_0$ determining the centre of the filter. Let's try a filter with $\tau = 0.2$ and $k_0 = 0$ since the pulse is centred at 0 in the frequency domain.

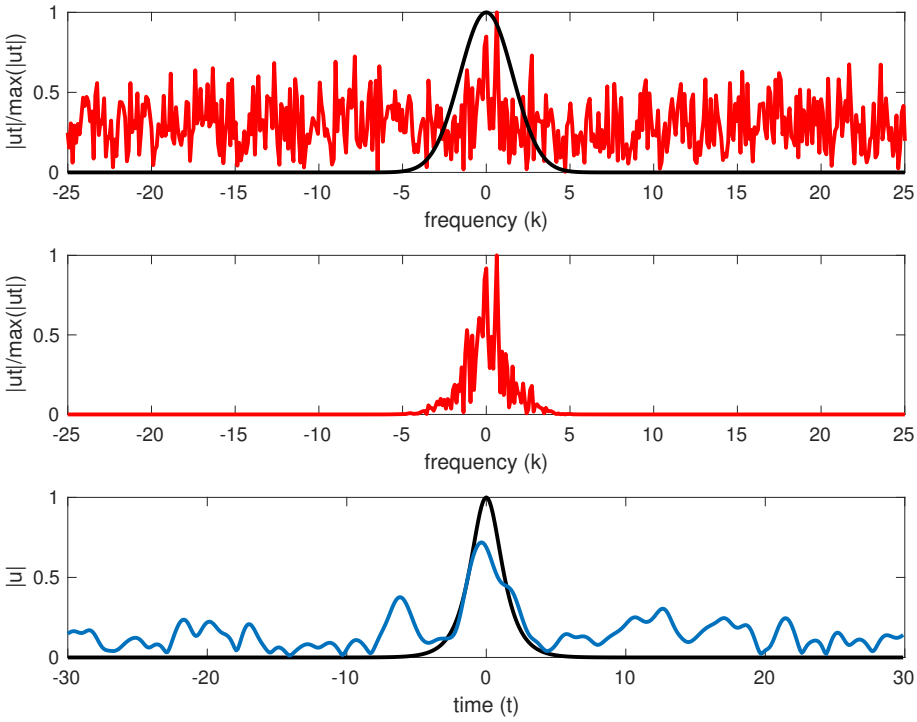
```
tau = 0.2;
k0 = 0;
filter = exp(-tau*(k - k0).^2); % Define the filter
unft = filter.*utn; % Apply the filter to the signal in frequency space
unf = ifft(unft);

% Plot the unfiltered signal in the frequency domain and the Gaussian filter
subplot(3,1,1)
plot(fftshift(k),abs(fftshift(utn))/max(abs(fftshift(utn))),'r','Linewidth',2)
hold on
plot(fftshift(k),fftshift(filter),'k','Linewidth',2)
axis([-25 25 0 1])
xlabel('frequency (k)')
ylabel('|ut|/max(|ut|)')

```

```
% Plot the filtered signal in the frequency domain
subplot(3,1,2)
plot(fftshift(k),abs(fftshift(unft))/max(abs(fftshift(unft))),'r','Linewidth',2)
axis([-25 25 0 1])
xlabel('frequency (k)')
ylabel('|ut|/max(|ut|)')

% Plot the filtered signal in the time domain and the ideal signal
subplot(3,1,3)
plot(t,u,'k','Linewidth',2)
hold on
plot(t,abs(unf),'Linewidth',2)
xlabel('time (t)')
ylabel('|u|')
```

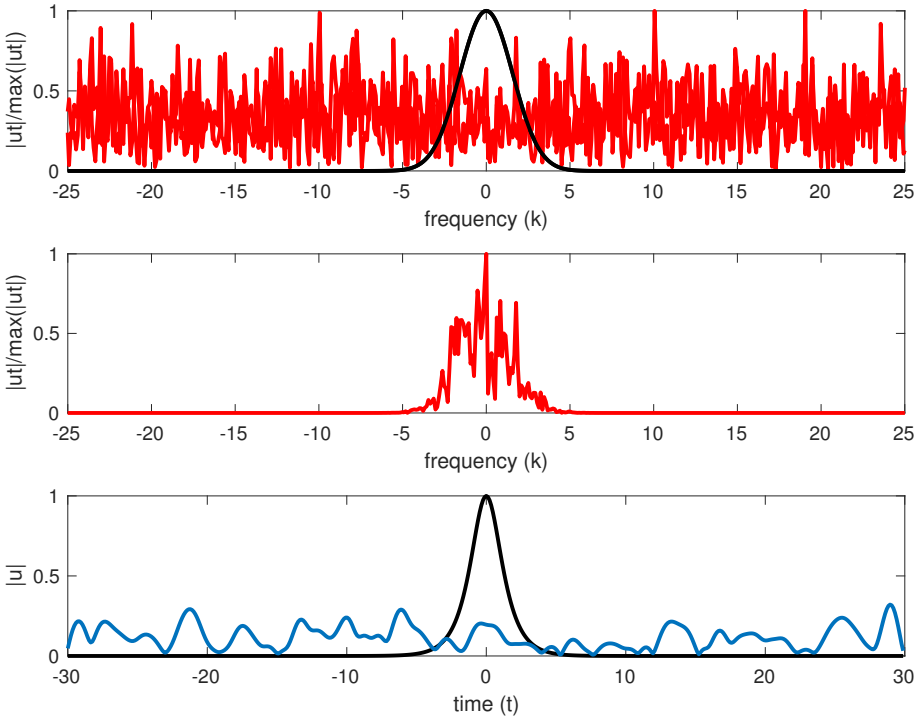



It's not perfect, but it's far better than the noisy signal. Just out of curiosity, what if there was no signal and you tried to do this filtering?

```
utn = noise*(normrnd(0,1,1,n) + 1i*normrnd(0,1,1,n));
tau = 0.2;
k0 = 0;
filter = exp(-tau*(k - k0).^2); % Define the filter
unft = filter.*utn; % Apply the filter to the signal in frequency space
unf = ifft(unft);

% Plot the unfiltered signal in the frequency domain and the Gaussian
% filter
subplot(3,1,1)
plot(fftshift(k),abs(fftshift(utn))/max(abs(fftshift(utn))),'r','Linewidth',2)
hold on
plot(fftshift(k),fftshift(filter),'k','Linewidth',2)
axis([-25 25 0 1])
xlabel('frequency (k)')
ylabel('|ut|/max(|ut|)')
```

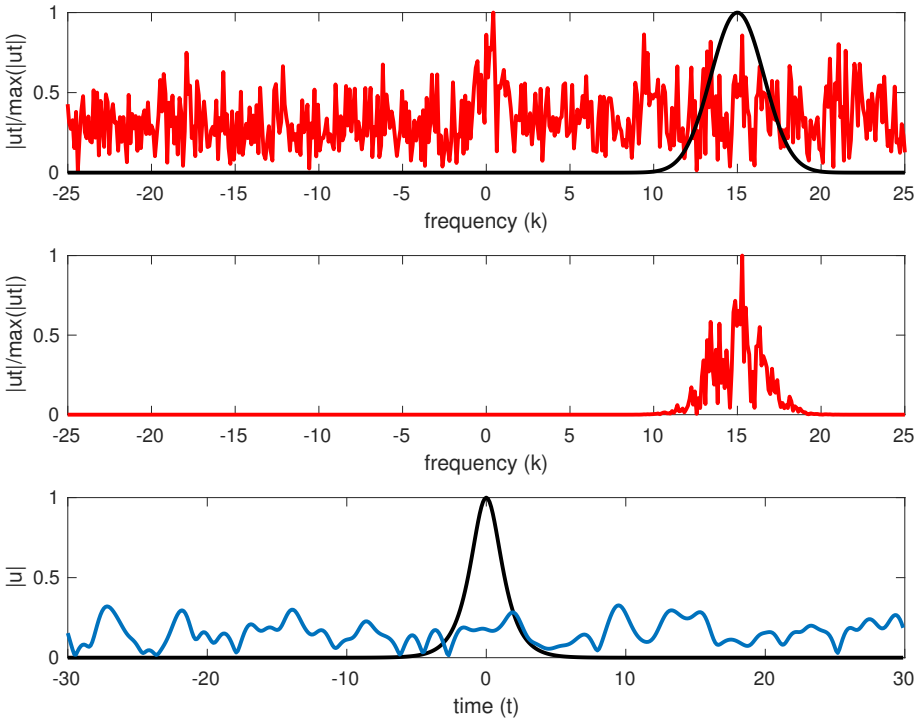
```
17
18 % Plot the filtered signal in the frequency domain
19 subplot(3,1,2)
20 plot(fftshift(k),abs(fftshift(unft))/max(abs(fftshift(unft))),'r','Linewidth',2)
21 axis([-25 25 0 1])
22 xlabel('frequency (k)')
23 ylabel('|ut|/max(|ut|)')
24
25 % Plot the filtered signal in the time domain and the ideal signal
26 subplot(3,1,3)
27 plot(t,u,'k','Linewidth',2)
28 hold on
29 plot(t,abs(unf),'Linewidth',2)
30 xlabel('time (t)')
31 ylabel('|u|')
```




**Important Note:** We knew where to filter around! Let's see what happens if we filter around the wrong frequency.

```
1  tau = 0.2;
2  k0 = 15;
3  utn = ut + noise*(normrnd(0,1,1,n) + 1i*normrnd(0,1,1,n));
4  filter = exp(-tau*(k - k0).^2); % Define the filter
5  unft = filter.*utn; % Apply the filter to the signal in frequency space
6  unf = ifft(unft);
7
8  % Plot the unfiltered signal in the frequency domain and the Gaussian
9  % filter
10 figure(6)
11 subplot(3,1,1)
12 plot(fftshift(k),abs(fftshift(utn))/max(abs(fftshift(utn))),'r','Linewidth',2)
13 hold on
14 plot(fftshift(k),fftshift(filter),'k','Linewidth',2)
15 axis([-25 25 0 1])
16 xlabel('frequency (k)')
17 ylabel('|ut|/max(|ut|)')
```

```
18
19 % Plot the filtered signal in the frequency domain
20 subplot(3,1,2)
21 plot(fftshift(k),abs(fftshift(unft))/max(abs(fftshift(unft))),'r','Linewidth',2)
22 axis([-25 25 0 1])
23 xlabel('frequency (k)')
24 ylabel('|ut|/max(|ut|)')
25
26 % Plot the filtered signal in the time domain and the ideal signal
27 subplot(3,1,3)
28 plot(t,u,'k','Linewidth',2)
29 hold on
30 plot(t,abs(unf),'Linewidth',2)
31 xlabel('time (t)')
32 ylabel('|u|')
```




You can see that this is tricky. Ultimately, the goal is to use repeated measurements from the radar of the airspace in order to try to avoid a detection error or failure. It should be noted that filter design, shape, and parametrization play significant roles in designed optimal filters. Thus filter design is an object of strong interest in signal processing applications.