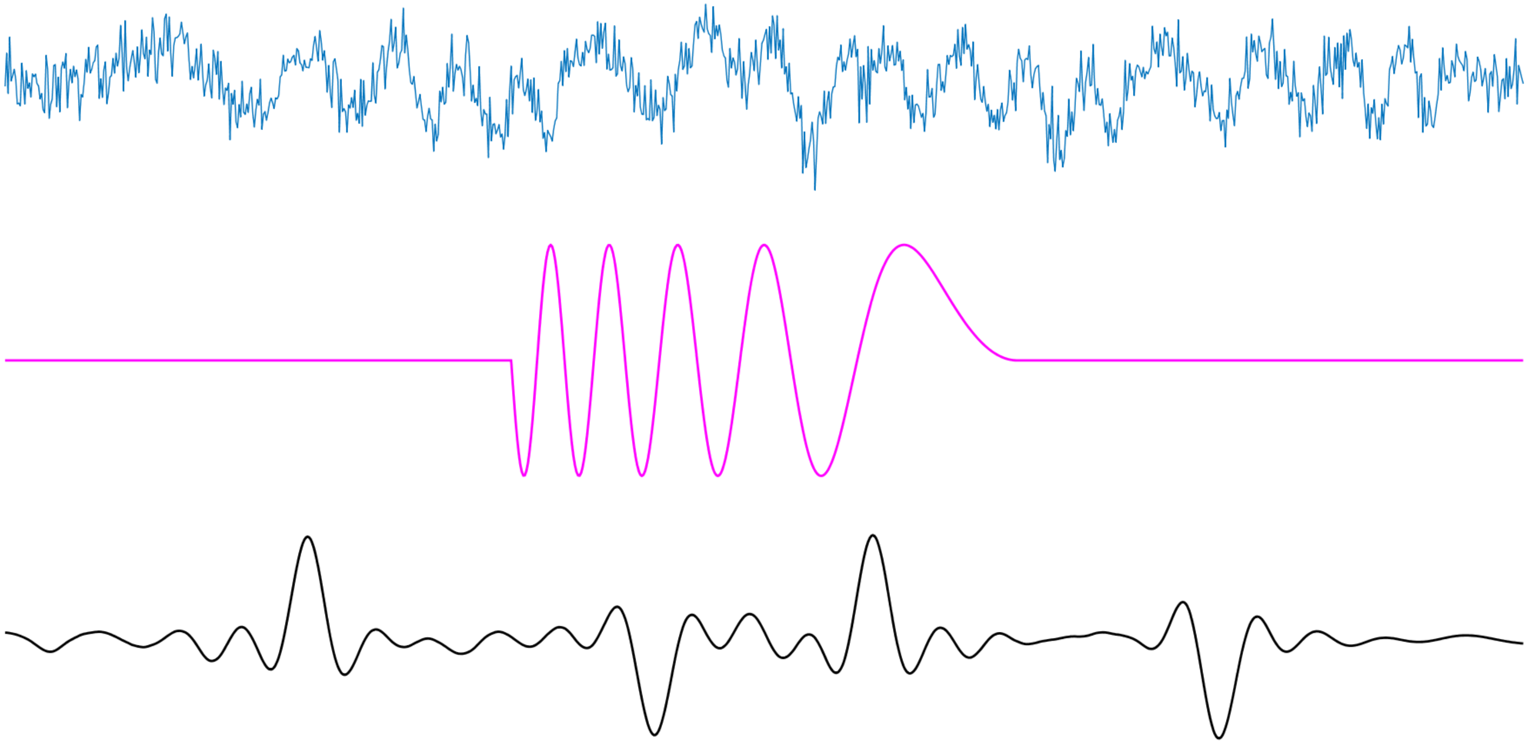
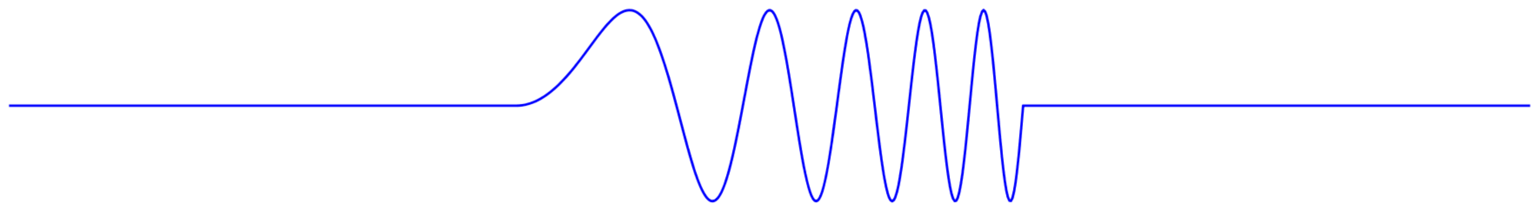
Matched Filtering



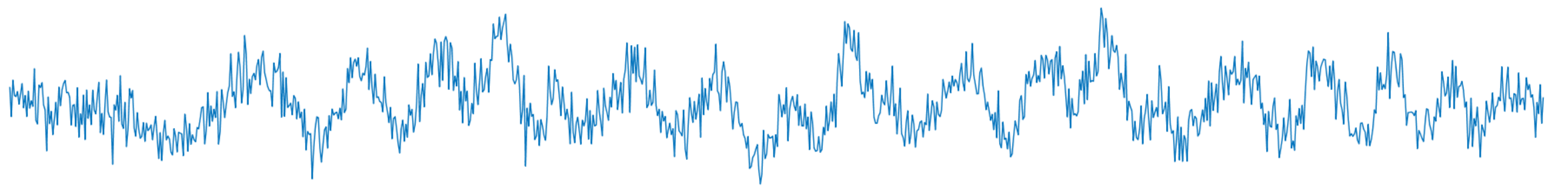
Dan Jacobellis | EE 445S | November 8, 2021

$g(t)$ is a signal or pulse generated by the transmitter



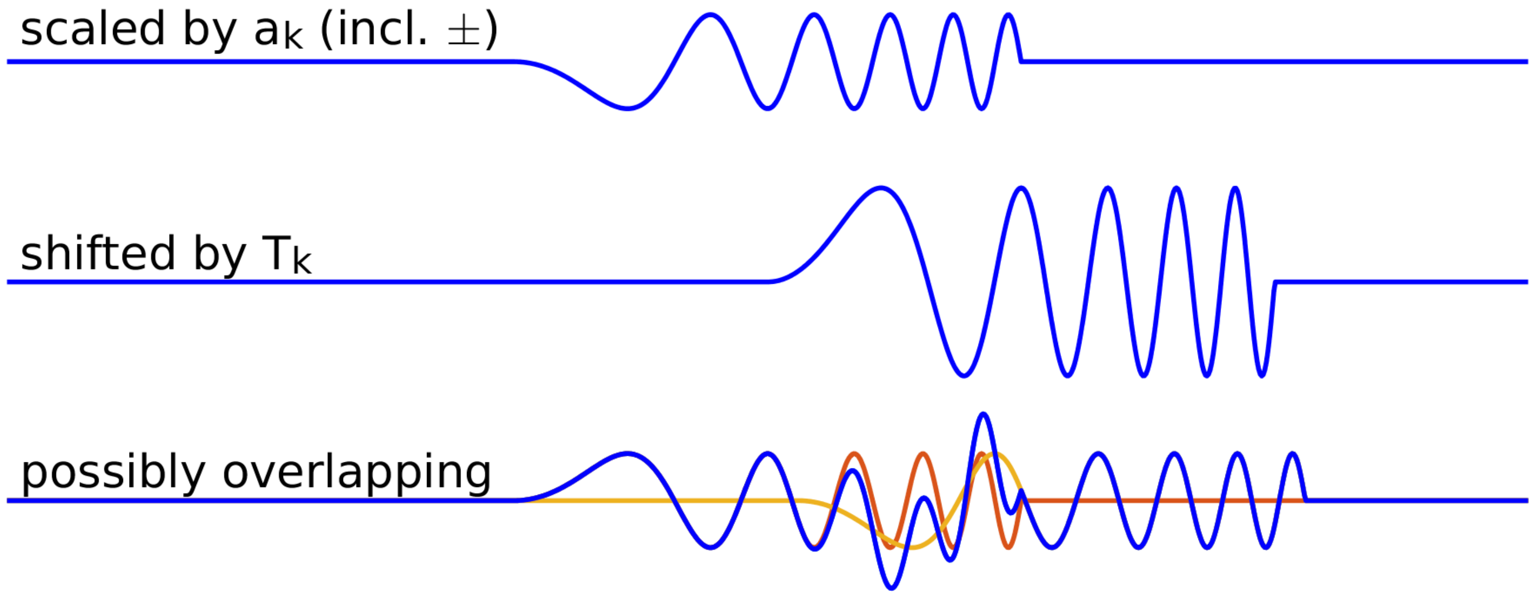
$w(t)$ is additive noise

$x(t) = w(t) + \sum_{k=1}^{K} a_k g(t - T_k)$ is the received signal



$$x(t) = w(t) + \sum_{k=1}^{K} a_k g(t - T_k)$$

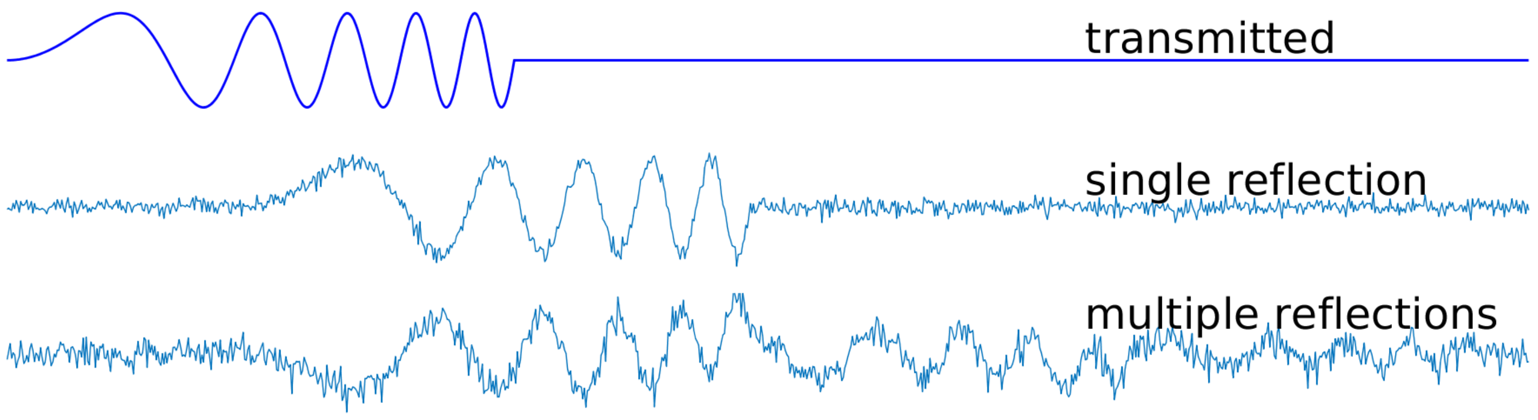
Contains copies of $g(t)$ which are



Goal: create a filter $h(t)$ which will help us to **detect** and **characterize** any copies of $g(t)$ that are present in $x(t)$.

Application: Radar/Sonar

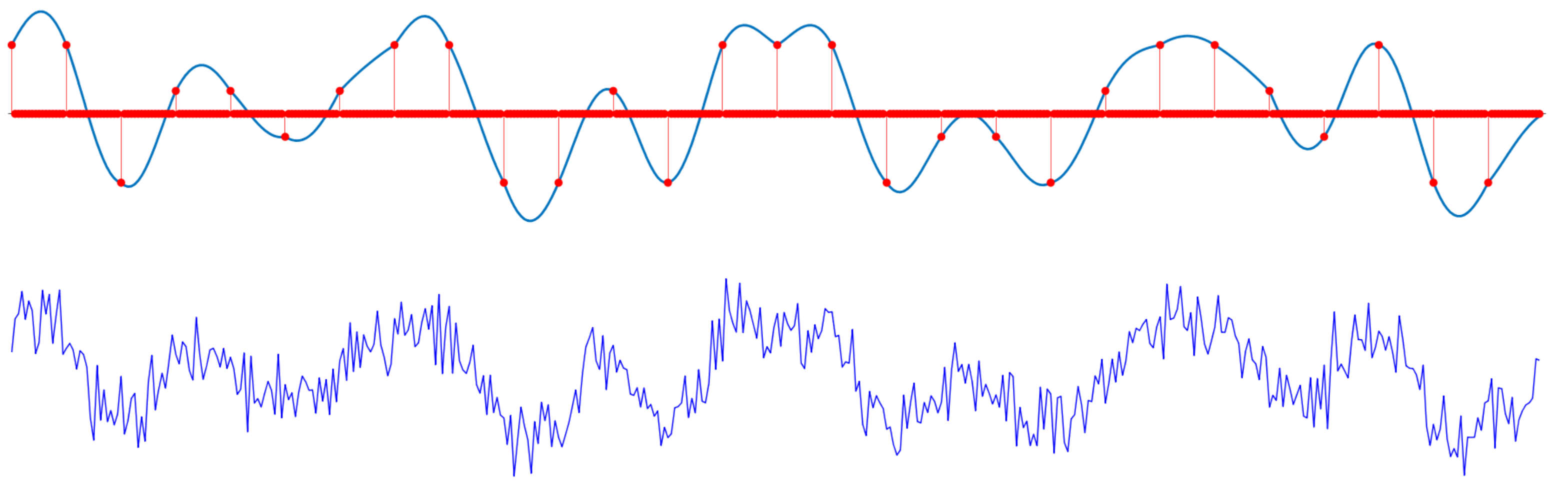
- Transmitter and receiver are at known locations (possibly the same location).
- The transmitter sends a pulse $g(t)$ (often a chirp signal).
- The signal propagates through space, and may reflect off of objects
- The receiver listens for the reflections and records them into $x(t)$



Goal: create a filter $h(t)$ which will help us to **detect** and **characterize** any copies of $g(t)$ that are present in $x(t)$.

Application: Digital Communication

- Transmitter and receiver are at different (possibly unknown) locations
- The transmitter sends a sequence of pulses $g(t)$ (often a raised cosine).
- Pulses are known to be separated by T_{sym}



Goal: create a filter $h(t)$ which will help us to **detect** and **characterize** any copies of $g(t)$ that are present in $x(t)$.

The received signal is

$$x(t) = w(t) + \sum_{k=1}^{K} a_k g(t - T_k)$$

Application of a filter $h(t)$ can be analyzed in terms of its effects on $w(t)$ and $a_k g(t - T_k)$ separately

$$\begin{aligned} y(t) &= h(t) * x(t) \\ &= \underbrace{h(t) * w(t)}_{\text{filtered noise}} + \sum_{k=1}^{K} \underbrace{h(t) * a_k g(t - T_k)}_{\text{filtered pulses}} \end{aligned}$$

To make predictions about a_k or T_k, we want to minimize the power of the filtered noise while maximizing the remaining signal.

The optimal filter can be derived formally by maximizing the signal to noise ratio.

The optimal filter can be derived informally by considering the definition of convolution.

Informal Derivation

Question: what filter $h(t)$ should I apply to $x(t)$ to help me **detect** and **characterize** any copies of $g(t)$?

Ideally, the answer should be useful to both of the applications.

Simplification: assume that $x(t)$, $g(t)$, and $h(t)$ must all be real. Then, convolution and cross-correlation have a simple relationship

$$x(t) * g(t) = x(t) \star g(-t)$$

$$x(t) \star g(t) = x(t) * g(-t)$$

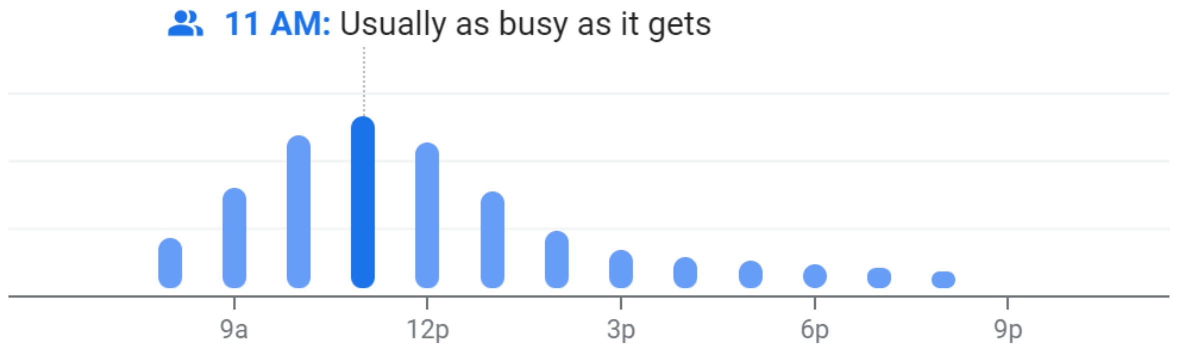
New question: what function $h(t)$ should I cross-correlate with $x(t)$ to help me **detect** and **characterize** any copies of $g(t)$?

Convolution and Cross-correlation

Why is it called cross-**correlation**? What is the relationship to correlation in probability/statistics?

In statistics, correlation measures similarity between two collections of samples.

Example: Want tacos. Don't want to wait in line. Length of line is a random variable.



What is correlation?

Let t_c be the current time. Imagine you collect some data.

k	**1**	**2**	**3**	**4**	**5**	$\cdots$
$x[k]$ = abs(t_c - 11:00am)	9	51	26	122	250	$\cdots$
$y[k]$ = Number of people in line	5	3	2	1	2	$\cdots$

The data are drawn from random variables X and Y. The correlation between these random variables is

$$\rho(X,Y) = \frac{\mathrm{cov}(X,Y)}{\sqrt{\mathrm{var}(X)\mathrm{var}(Y)}}$$

The **sample correlation** of the collected data is

$$r(x,y) = \frac{1}{\sigma_x \sigma_y} \sum_{k=1}^{K} (x[k] - \mu_x)(y[k] - \mu_y)$$

What is correlation?

$$r(x,y) = \frac{1}{\sigma_x \sigma_y} \sum_{k=1}^{K} (x[k]-\mu_x)(y[k]-\mu_y)$$

Correlation is an *inner product*. An inner product measures similarity of two signals.

$$\text{Inner product of two discrete signals } x_1[n], x_2[n] = \sum_{k=-\infty}^{\infty} x_1[k]x_2[k]$$

$$\text{Inner product of two continuous signals } x_1(t), x_2(t) = \int_{\tau=-\infty}^{\infty} x_1(\tau)x_2(\tau)d\tau$$

Positive correlation indicates similarity between x and y. Negative correlation indicates similarity between $-x$ and y. Noise will lower the correlation.

If $y[k] = \underbrace{a}_{>0} x[k] + \underbrace{w}_{\text{noise}}$ then $0 < r < 1$.

Convolution and cross-correlation as an inner product

The convolution operation is defined as:

$$x_1[n] * x_2[n] = \sum_{k=-\infty}^{\infty} x_1[k] x_2[n-k]$$

$$x_1(t) * x_2(t) = \int_{\tau=-\infty}^{\infty} x_1(\tau) x_2(t-\tau)$$

The cross-correlation operation is nearly identical to convolution, but without the 'flip'

$$x_1[n] \star x_2[n] = \sum_{k=-\infty}^{\infty} \overline{x_1[k]} x_2[n+k]$$

$$x_1(t) \star x_2(t) = \int_{\tau=-\infty}^{\infty} \overline{x_1(\tau)} x_2(t+\tau)$$

Convolution and cross-correlation as an inner product

An inner product takes two signals and returns a number.

$$\text{Inner product of two continuous signals } x_1(t), x_2(t) = \int_{\tau=-\infty}^{\infty} x_1(\tau)x_2(\tau)d\tau$$

Convolution and cross-correlation take two signals and return **another signal**

$$x_1(t) * x_2(t) = \int_{\tau=-\infty}^{\infty} x_1(\tau)x_2(t-\tau)$$

$$x_1(t) \star x_2(t) = \int_{\tau=-\infty}^{\infty} \overline{x_1(\tau)}x_2(t+\tau)$$

The new signal is no longer a function of the original time variable. Instead it's a function of the relative **time shift** between the two signals.

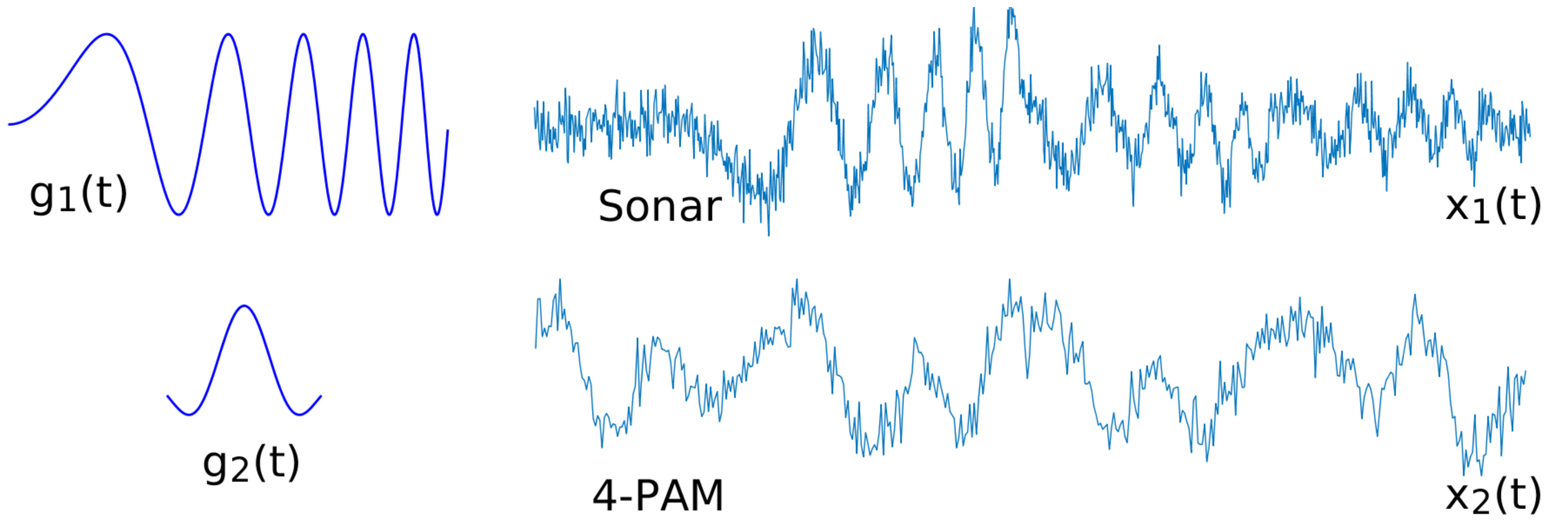
For any particular time shift, cross-correlation tells us about about the similarity of $\overline{x_1(t)}$ and $x_2(t)$.

For any particular time shift, convolution tells us about the similarity of $x_1(t)$ and $x_2(-t)$.

Informal Derivation

Back to the question:

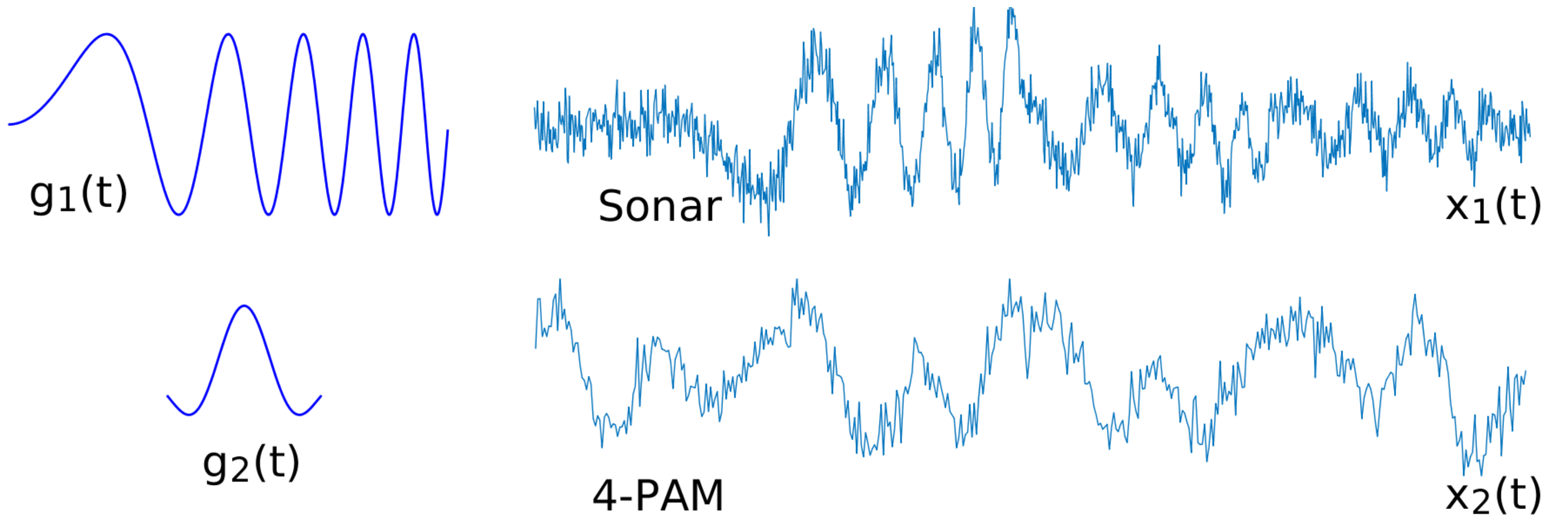
Assume that $x(t)$ and $g(t)$ are both real. What function should we cross-correlate with $x(t)$ to help **detect** and **characterize** any copies of $g(t)$?



Informal Derivation

Back to the question:

Assume that $x(t)$ and $g(t)$ are both real. What function should we cross-correlate with $x(t)$ to help **detect** and **characterize** any copies of $g(t)$?

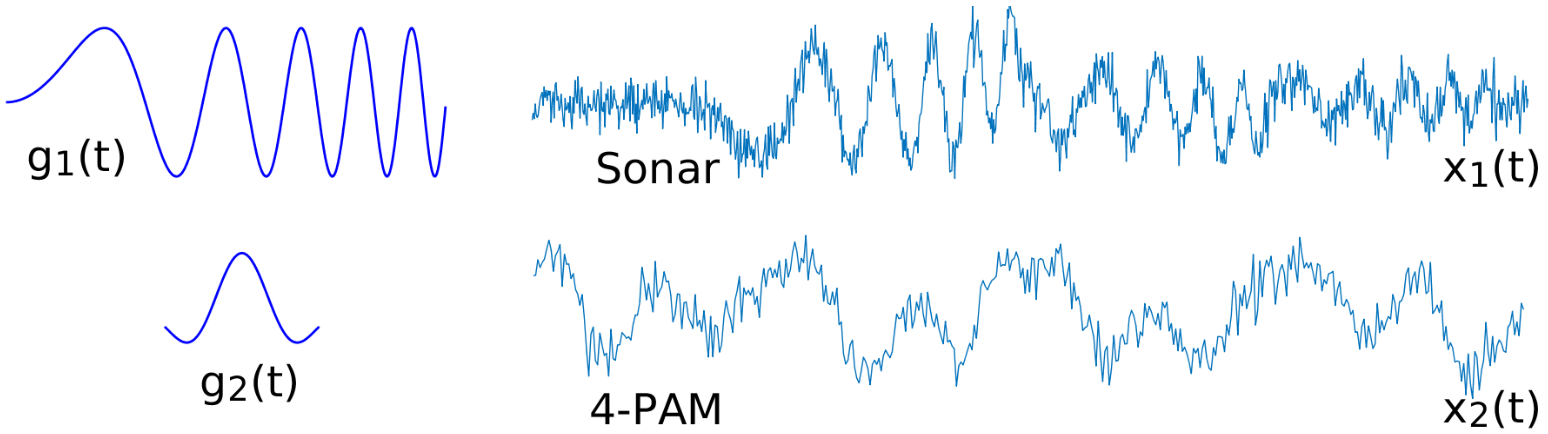


Answer: The function that will produce a measure of similarity between $x(t)$ and $g(t - \tau)$ for every possible time shift τ.

Answer: We should cross-correlate x(t) with the signal g(t) that we want to detect.

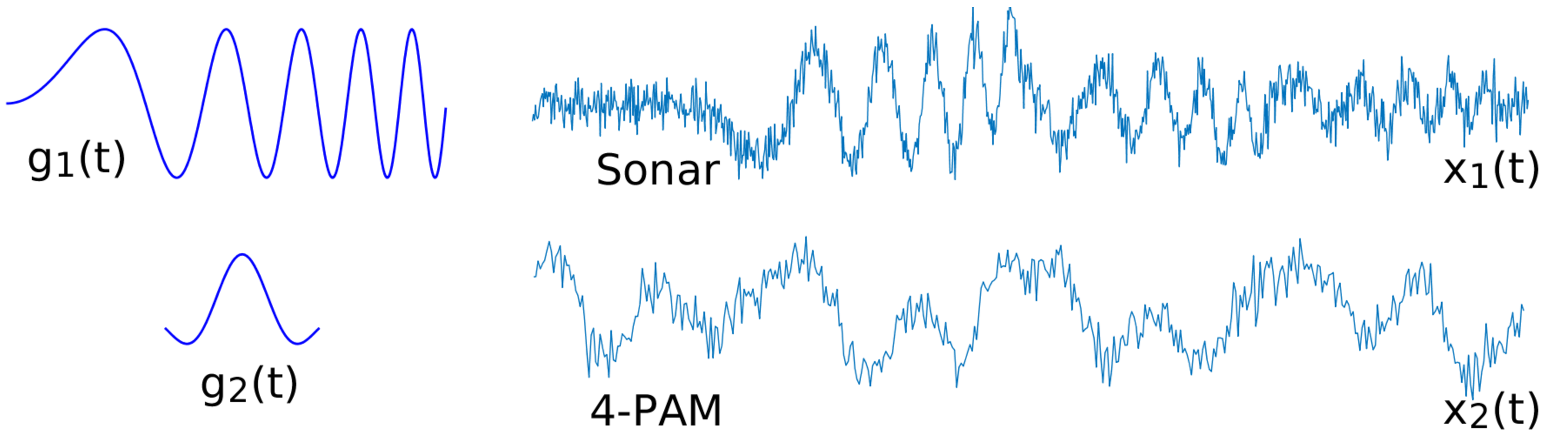
Informal Derivation

What filter $h(t)$ should be applied to $x(t)$ to help **detect** and **characterize** any copies of $g(t)$?



Informal Derivation

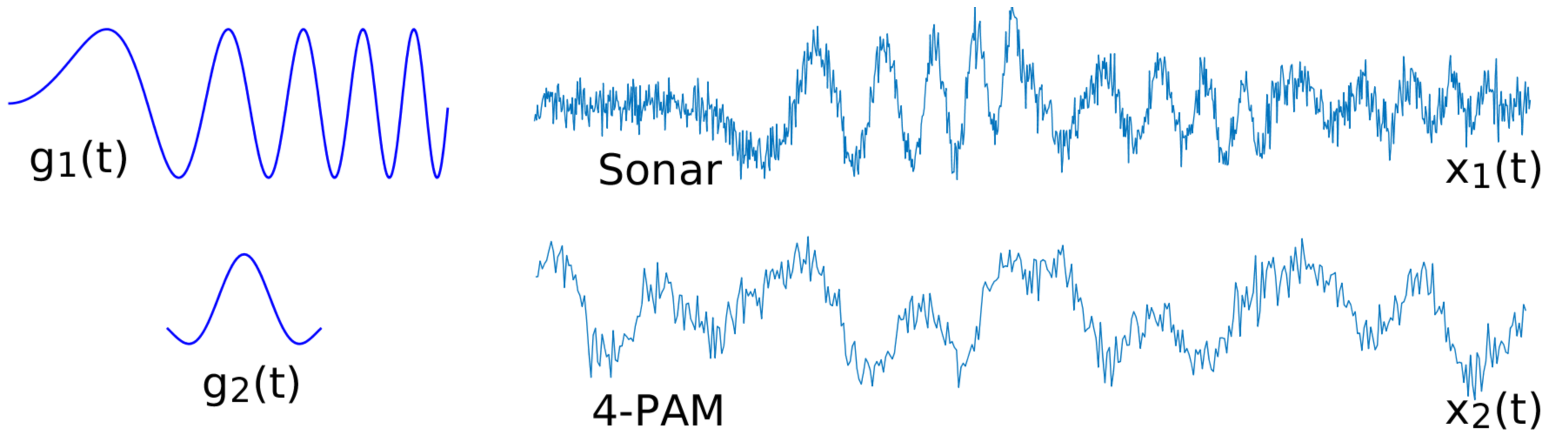
What filter $h(t)$ should be applied to $x(t)$ to help **detect** and **characterize** any copies of $g(t)$?



Answer: The time-reversed filter $h(-t)$ should **match** the known signal $g(t)$.

Informal Derivation

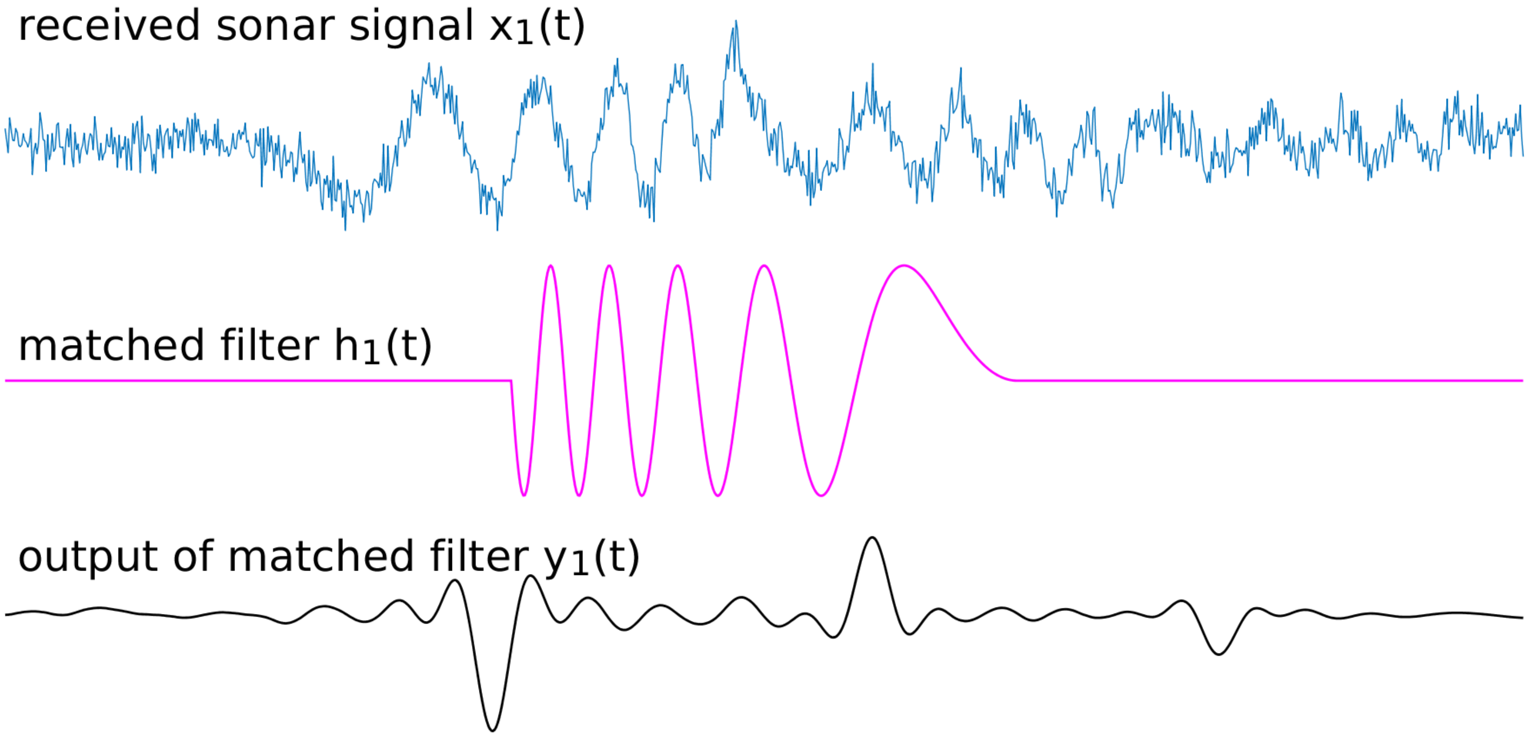
What filter $h(t)$ should be applied to $x(t)$ to help **detect** and **characterize** any copies of $g(t)$?

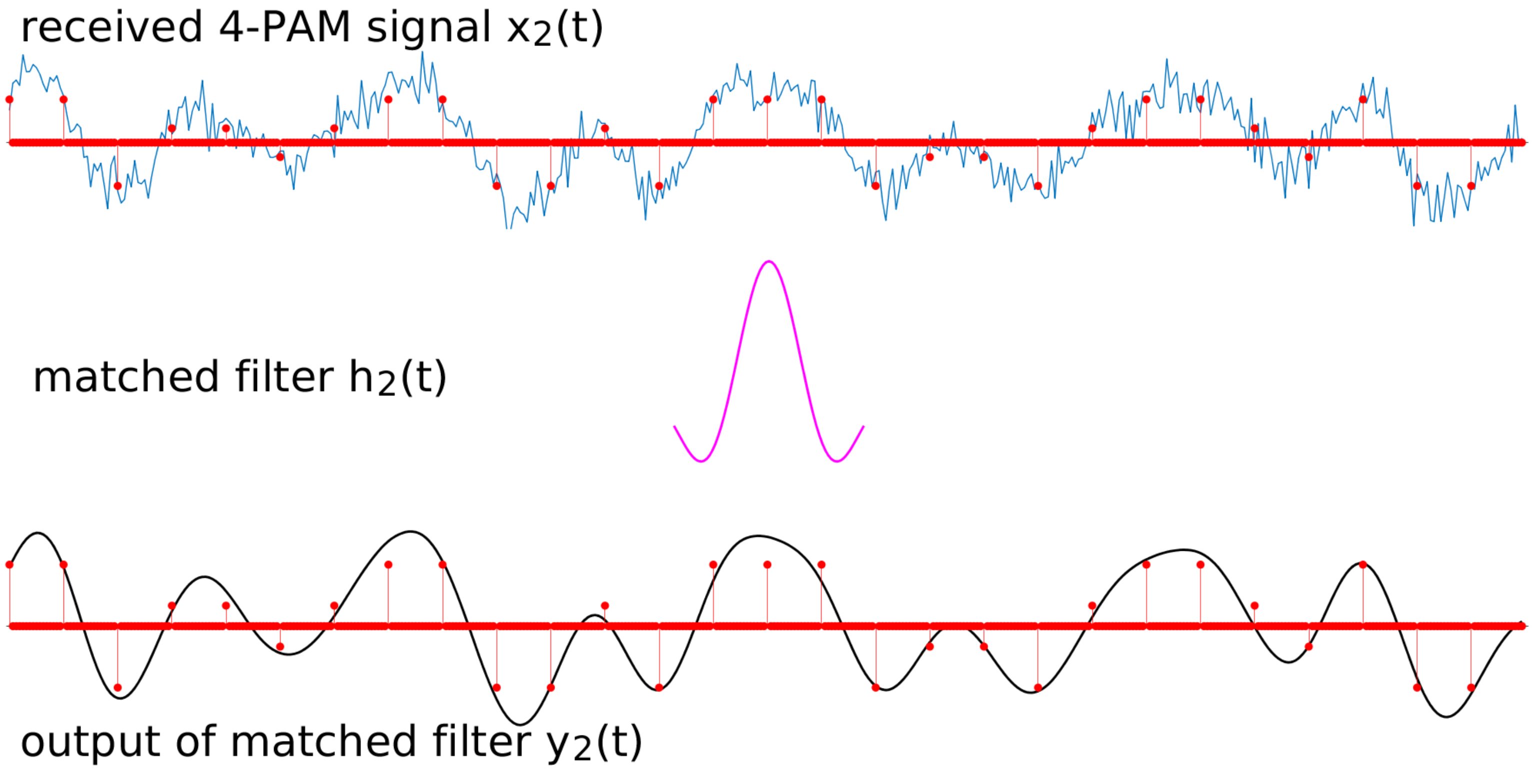


Answer: The time-reversed filter $h(-t)$ should **match** the known signal $g(t)$.

Conclusion: The optimal filter is the matched filter: $h(t) = g(-t)$

Disclaimer: if we allow complex signals then $h(t) = \overline{g(-t)}$


received sonar signal $x_1(t)$
matched filter $h_1(t)$
output of matched filter $y_1(t)$

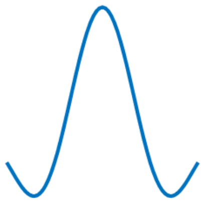

received 4-PAM signal $x_2(t)$
matched filter $h_2(t)$
output of matched filter $y_2(t)$

Simulation

In [81]:

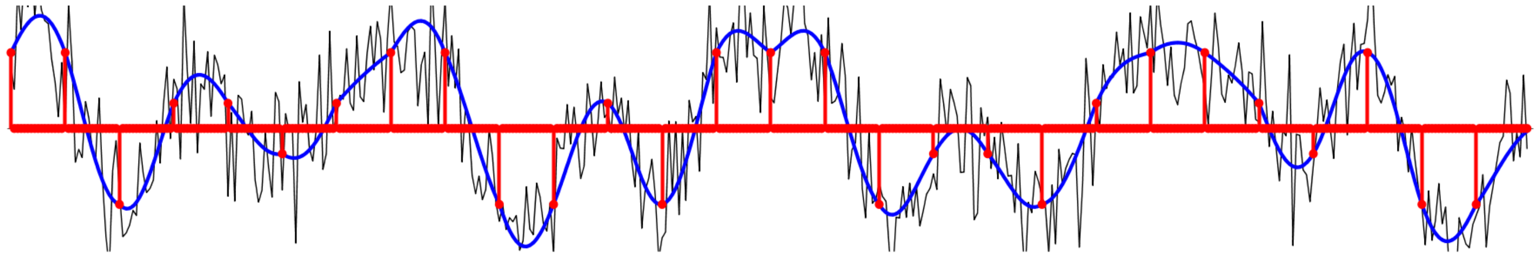
```
g = sinc(-2:(1/16):2);
E = sum(g.^2);
```

matched filter = pulse shaping filter =



In [85]:

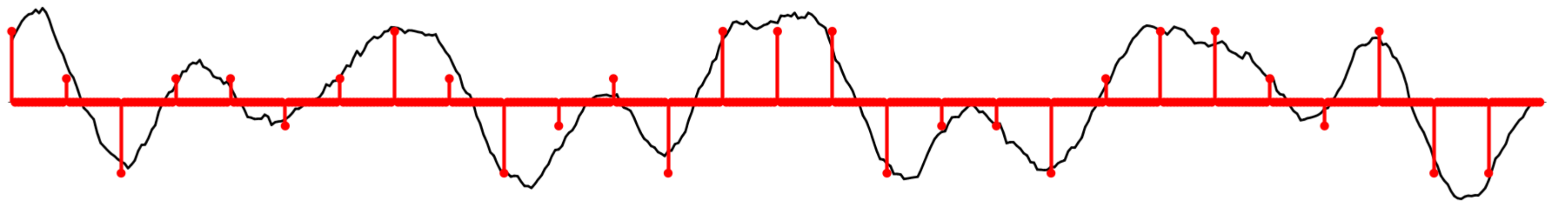
```
symbols = reshape((dec2bin(uint8('meatball'))'),[28,2]);
amplitudes = upsample(2*bin2dec(serialized_data)-3,16);
pulses = conv(amplitudes,g/E,'same');
x = pulses + 0.1*randn(size(pulses));
```



Simulation

In [98]:

```
quantize = @(x) 2*(round(0.5*x + 1.5)-1.5);
h = E*ones(16,1)/16;
y = conv(x,h,'same');
yT = quantize(downsample(y,16));
```



In [101]:

```
number_errors = sum(abs((yT - downsample(amplitudes,16)))>0)
```

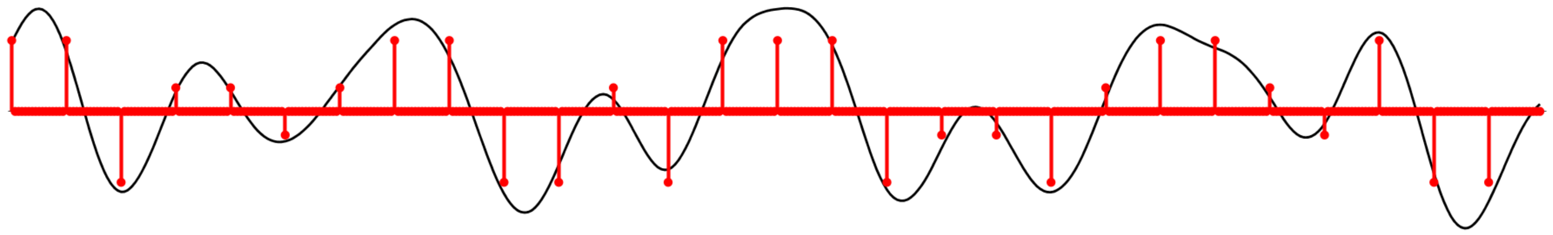
```
number_errors =

     3
```

Simulation

In [102]:

```
h = g;
y = conv(x,h,'same');
yT = quantize(downsample(y,16));
```



In [105]:

```
number_errors = sum(abs((yT - downsample(amplitudes,16)))>0)
```

```
number_errors =

     0
```

Formal Derivation

Application of a filter $h(t)$ can be analyzed in terms of its effects on $w(t)$ and $a_k g(t - T_k)$ separately

$$\begin{aligned} y(t) &= h(t) * x(t) \\ &= \underbrace{h(t) * w(t)}_{\text{filtered noise}} + \sum_{k=1}^{K} \underbrace{h(t) * a_k g(t - T_k)}_{\text{filtered pulses}} \end{aligned}$$

To make predictions about a_k or T_k, we want to minimize the power of the filtered noise while maximizing the remaining signal.

We want to maximize the signal to noise ratio at the times T_k Where a copy of $g(t)$ is present.

Formal Derivation

$$\begin{aligned} y(t) &= h(t) * x(t) \\ &= \underbrace{h(t) * w(t)}_{\text{filtered noise}} + \sum_{k=1}^{K} \underbrace{h(t) * a_k g(t - T_k)}_{\text{filtered pulses}} \end{aligned}$$

$$|g_0(t)|^2 = |g(t) * h(t)|^2 = \text{output signal power}$$

$$E\left[n^2(t)\right] = E\left[(w(t) * h(t))^2\right] = \text{output noise power}$$

Goal: maximize the peak pulse SNR

$$\eta = \frac{\text{Peak signal power}}{\text{Average noise power}} = \frac{|g_0(t)|^2}{E\left[n^2(t)\right]}$$

Formal Derivation

We don't know the amplitude values of $n(t)$ but we can find its **power spectrum**.

$$\begin{aligned} P_N(f) &= |N(f)|^2 \\ &= N(f)\overline{N(f)} \\ &= \mathcal{F}\left\{n(t) * \overline{n(t)}\right\} \\ &= \mathcal{F}\left\{R_n(t)\right\} \end{aligned}$$

For the two-sided random noise signal $n(t)$, the Fourier transform may not exist, but its power spectrum does.

Formal Derivation

For a zero-mean Gaussian random process $n(t)$ with variance σ^2, the autocorrelation is $R_n(t) = \sigma^2\delta(t)$

Therefore, its power spectrum is $P_n(f) = \mathcal{F}\left\{R_n(t)\right\} = \sigma^2$

The noise at the output of the matched filter is $n(t) = w(t) * h(t)$

Its power spectrum is

$$S_N(f) = S_W(f)S_H(f) = \frac{N_0}{2}\left|H(f)\right|^2$$

$$\text{Average noise power} = E\left[n^2(t)\right] = \int_{-\infty}^{\infty} S_N(f)df = \int_{-\infty}^{\infty} \frac{N_0}{2}\left|H(f)\right|^2 df$$

Formal Derivation

$$g_0(t) = g(t) * h(t)$$
$$G_0(t) = H(f)G(f)$$

$$\begin{aligned} |g_0(t)|^2 &= |\mathcal{F}^{-1}\{G_0(f)\}|^2 \\ &= |\mathcal{F}^{-1}\{H(f)G(f)\}|^2 \\ &= \left|\int_{-\infty}^{\infty} H(f)G(f)e^{j2\pi fT}df\right|^2 \end{aligned}$$

$$\eta = \frac{\text{Peak signal power}}{\text{Average noise power}} = \frac{|g_0(t)|^2}{E\left[n^2(t)\right]} = \frac{\left|\int_{-\infty}^{\infty} H(f)G(f)e^{j2\pi fT}df\right|^2}{\frac{N_0}{2}\int_{-\infty}^{\infty}|H(f)|^2 df}$$

Formal Derivation

$$\eta = \frac{\text{Peak signal power}}{\text{Average noise power}} = \frac{\left|\int_{-\infty}^{\infty} H(f)G(f)e^{j2\pi fT}df\right|^2}{\frac{N_0}{2}\int_{-\infty}^{\infty}|H(f)|^2 df}$$

Question: How do we maximize η?

Formal Derivation

$$\eta = \frac{\text{Peak signal power}}{\text{Average noise power}} = \frac{\left|\int_{-\infty}^{\infty} H(f)G(f)e^{j2\pi fT}df\right|^2}{\frac{N_0}{2}\int_{-\infty}^{\infty}|H(f)|^2 df}$$

Question: How do we maximize η?

Answer: Schwarz inequality can be used to determine an **upper bound** for the **inner product.**

$$|\langle u, v\rangle|^2 \leq |u|^2|v|^2$$

$$\left|\int \phi_1(x)\phi_2(x)dx\right|^2 \leq \int |\phi_1(x)|^2 dx \int |\phi_2(x)|^2 dx$$

Formal Derivation

$$\eta = \frac{\text{Peak signal power}}{\text{Average noise power}} = \frac{\left|\int_{-\infty}^{\infty} H(f)G(f)e^{j2\pi fT}df\right|^2}{\frac{N_0}{2}\int_{-\infty}^{\infty}|H(f)|^2 df}$$

Question: How do we maximize η?

Answer: Schwarz inequality can be used to determine an **upper bound** for the **inner product.**

$$|\langle u, v\rangle|^2 \leq |u|^2|v|^2$$

$$\left|\int \phi_1(x)\phi_2(x)dx\right|^2 \leq \int |\phi_1(x)|^2 dx \int |\phi_2(x)|^2 dx$$

After some simplification, the maximum SNR becomes

$$\eta_{\max} = \frac{2}{N_0}\int_{-\infty}^{\infty}|G(f)|^2 df$$

In lecture assignment 4

In-Lecture Assignment Related to Homework #6 Consider performing an iterative maximization of

$$J(x) = 8 - x^2 + 6\cos(6x)$$

via the steepest descent algorithm (JSK equation (6.5) on page 116) with the sign on the update reversed from negative to positive so that the algorithm will maximize rather than minimize; i.e.

$$x[k+1] = x[k] + \mu \frac{dJ(x)}{dx}$$

where $x = x[k]$

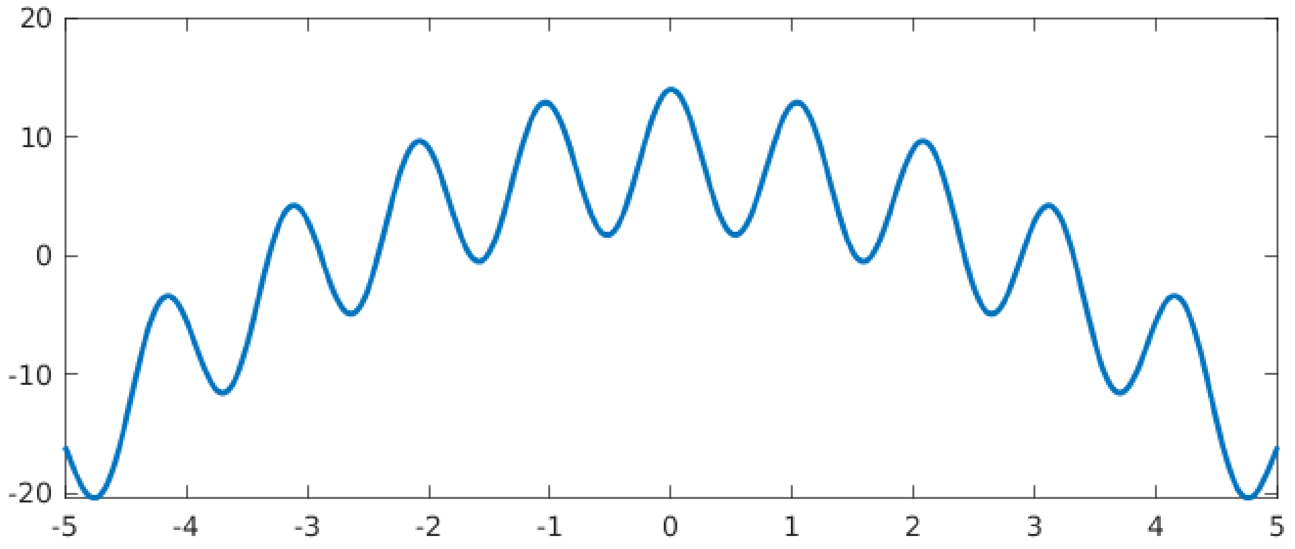
In lecture assignment 4

Visualize and analyze the shape of the objective function $J(x)$.

1. Plot $J(x)$ for $-5 < x < 5$. Give the Matlab code for your answer.
2. Describe the plot.
3. How many local maxima do you see?

In [123]:

```
J = @(x) 8 - x.^2 + 6*cos(6*x);
x = linspace(-5,5,1000);
plot(x,J(x),'linewidth',2');
```



In lecture assignment 4

Implement the steepest descent algorithm in Matlab with x_0 = 0.7.

1. To what value does the steepest descent algorithm converge?

2. Is the convergent value of x in the global maximum of J(x)? Why or why not?

In [125]:

```
dJdx = @(x) -2*x - 36*sin(6*x);
N=50;          % number of iterations
mu=0.001;      % algorithm stepsize
x=zeros(1,N); % initialize sequence of x values to zero
x(1)=0.7;      % starting point x(1)
for k=1:N-1
    x(k+1)= x(k) + (dJdx(x(k)))*mu;   % update equation
end
x(end)
```

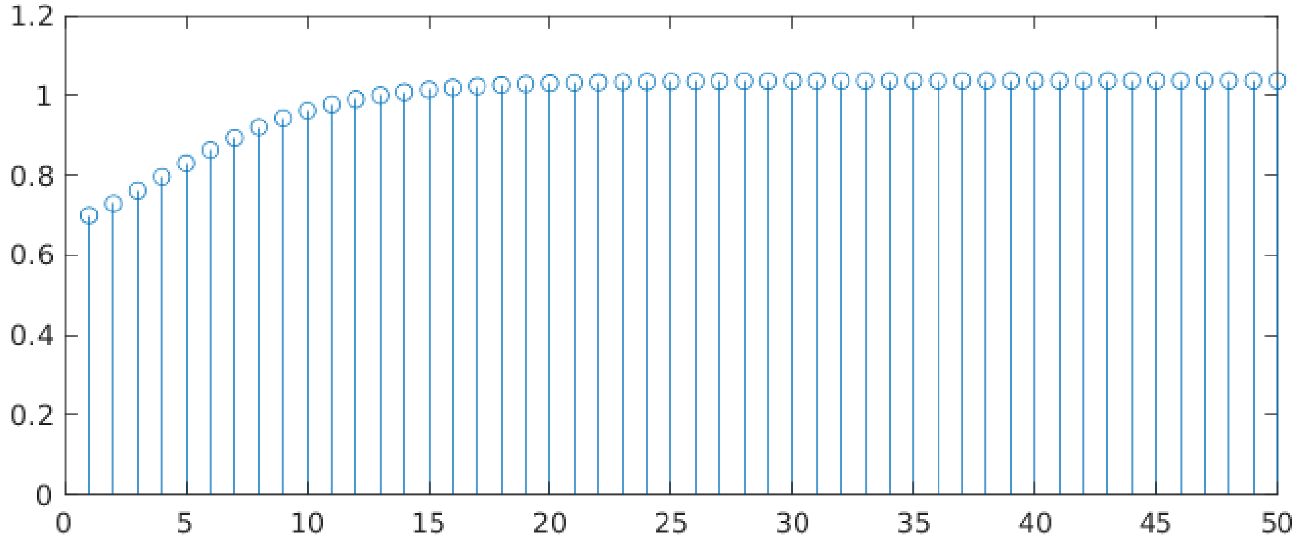
```
ans =

    1.0376
```

In lecture assignment 4

In [126]:

```
stem(x)
```



1.2
1
0.8
0.6
0.4
0.2
0
0
5
10
15
20
25
30
35
40
45
50