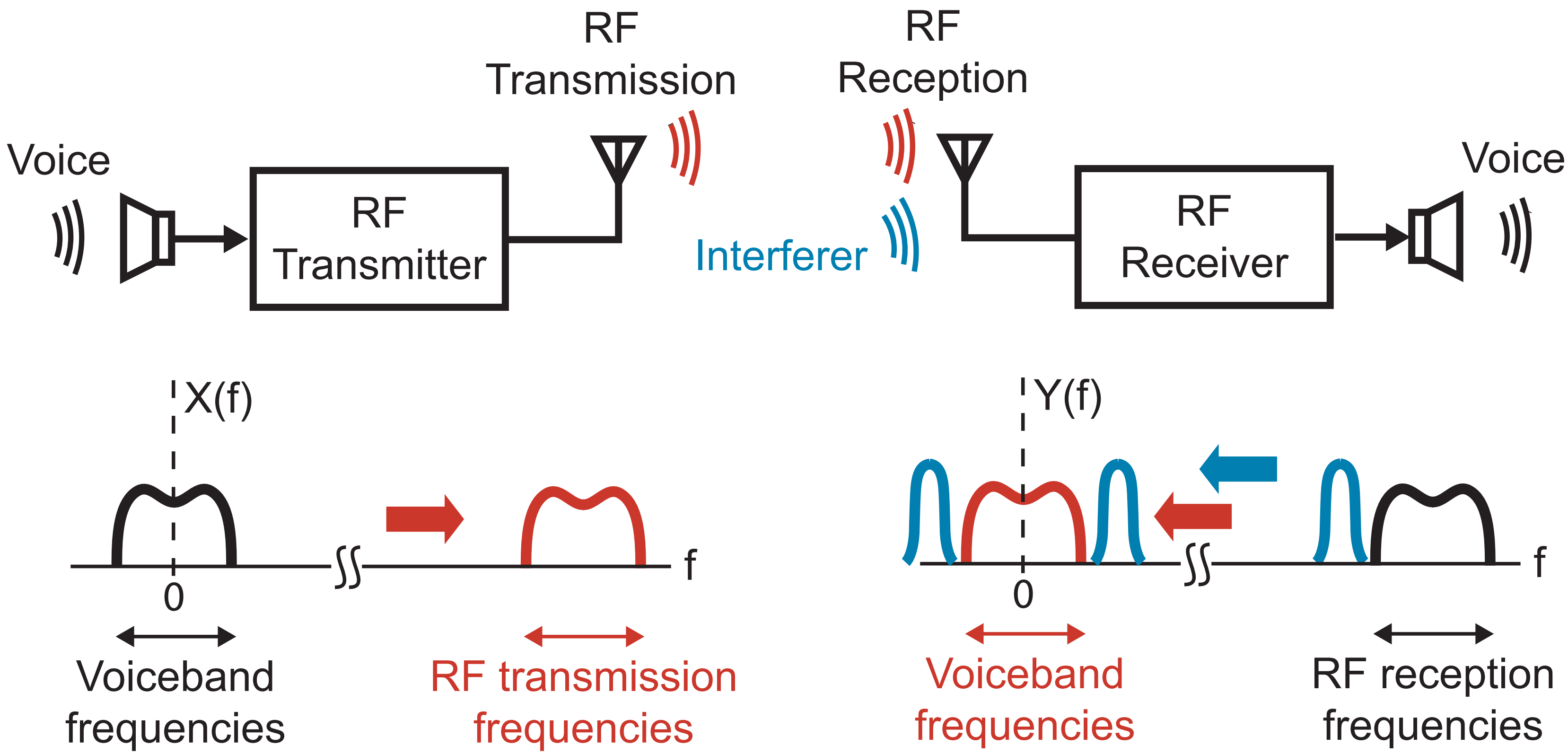
Filtering in Continuous and Discrete Time

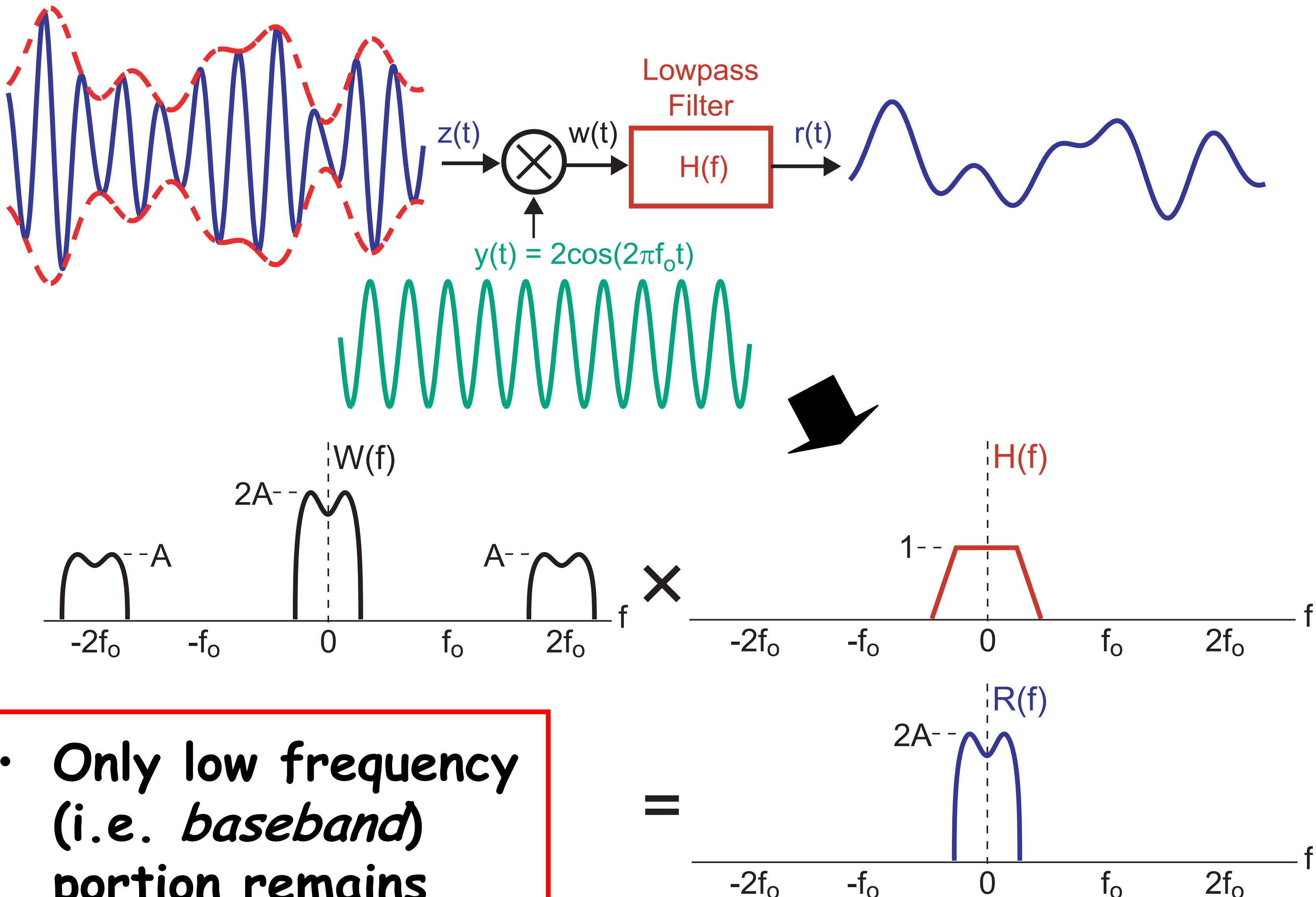
- Lowpass, highpass, bandpass filtering
- Filter response to cosine wave inputs
- Discrete-Time Fourier Transform
- Filtering based on difference equations

Motivation for Filtering



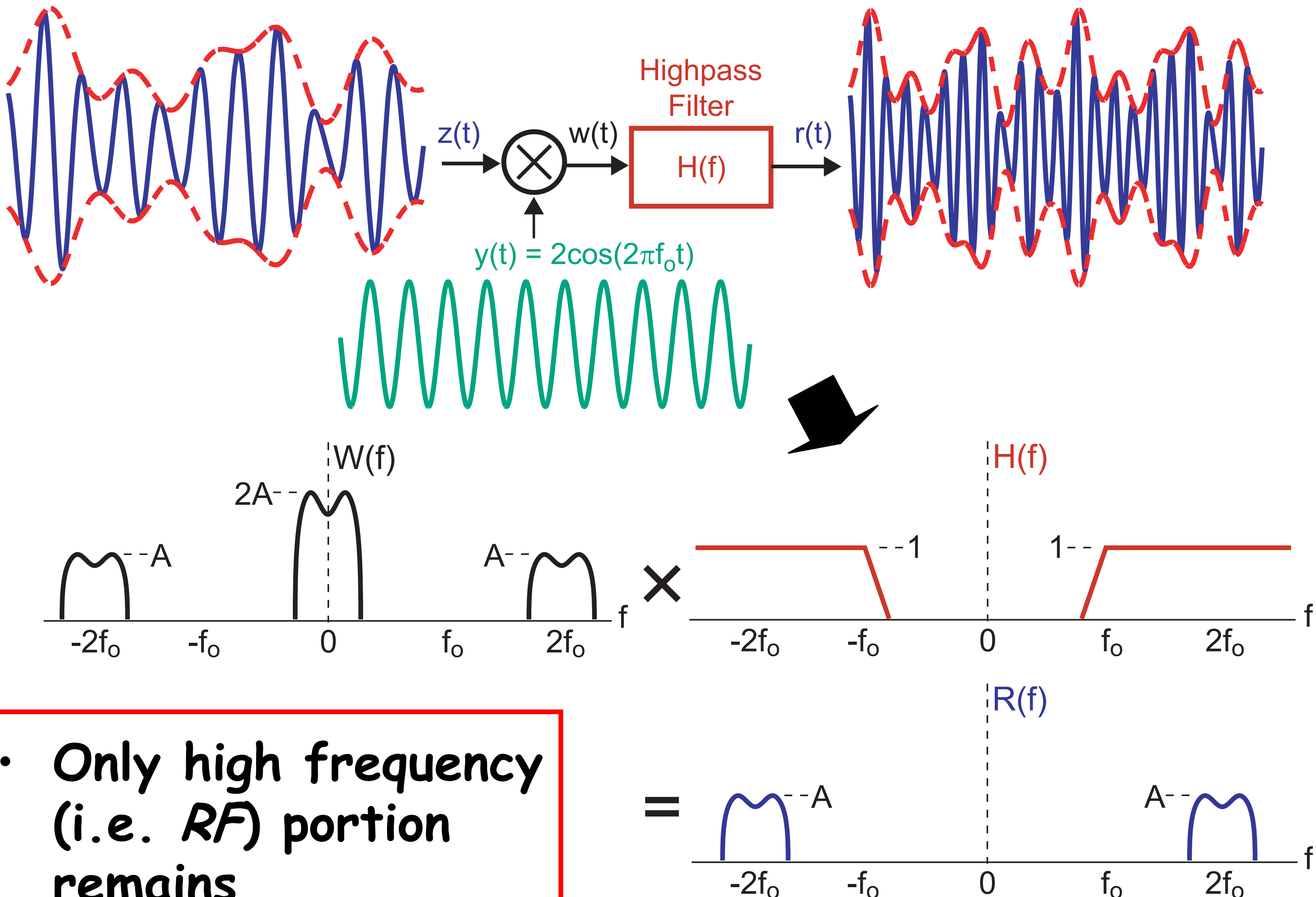
- **Filtering is used to remove undesired signals outside of the frequency band of interest**
  - Enables selection of a specific radio, TV, WLAN, cell phone, cable TV *channel* ...
  - Undesired channels are often called interferers

Lowpass Filter



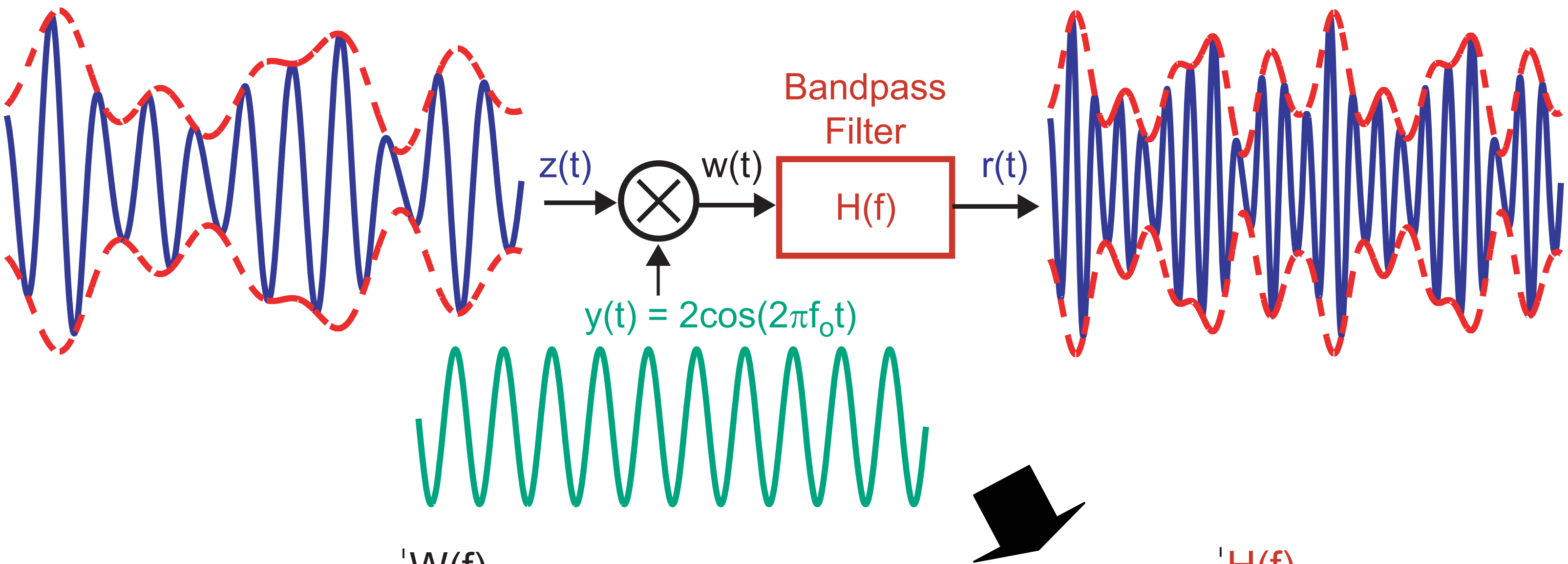
- Only low frequency (i.e. *baseband*) portion remains

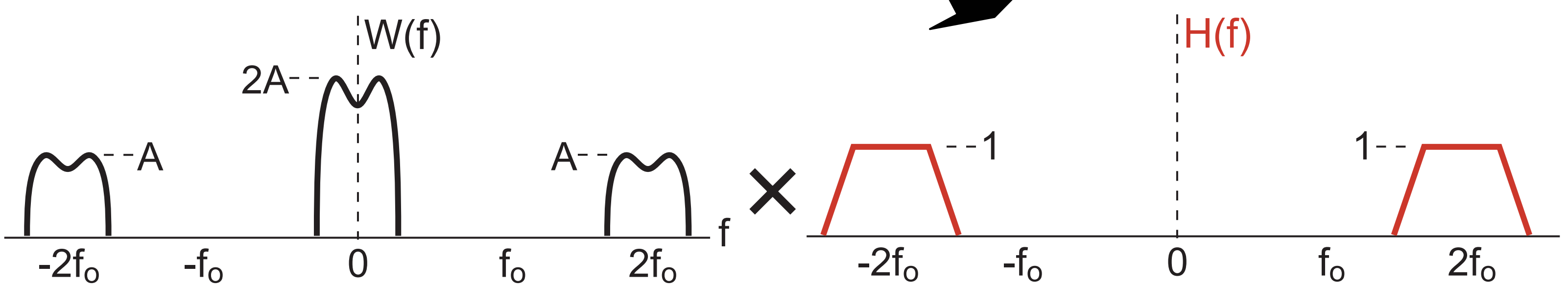
Highpass Filter



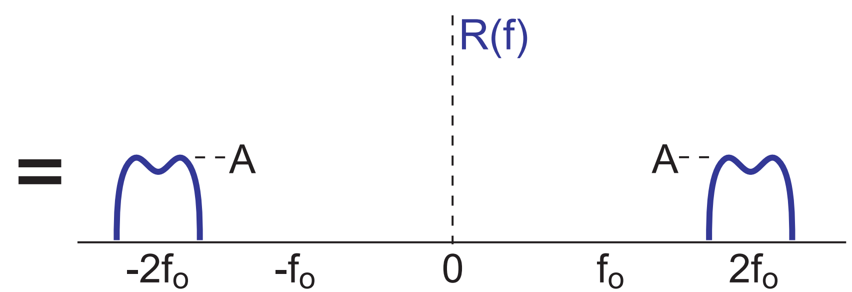
- **Only high frequency (i.e. *RF*) portion remains**

Bandpass Filter



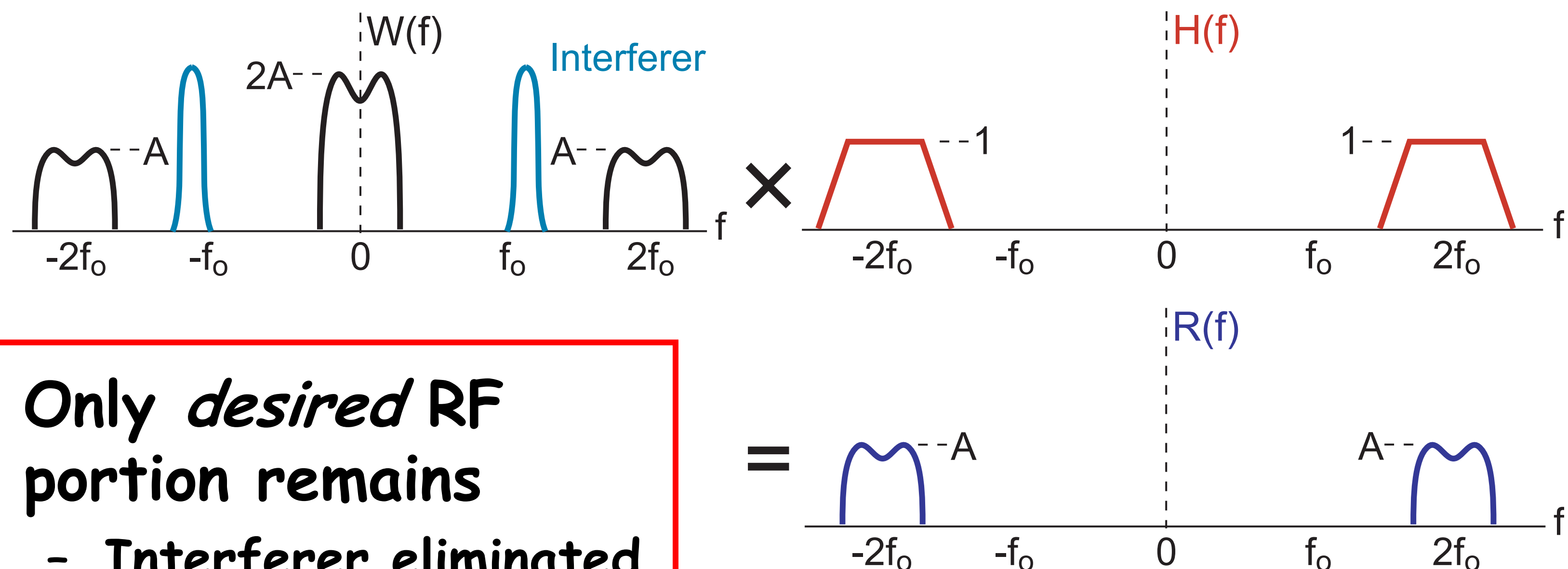


- **Only high frequency (i.e. *RF*) portion remains**



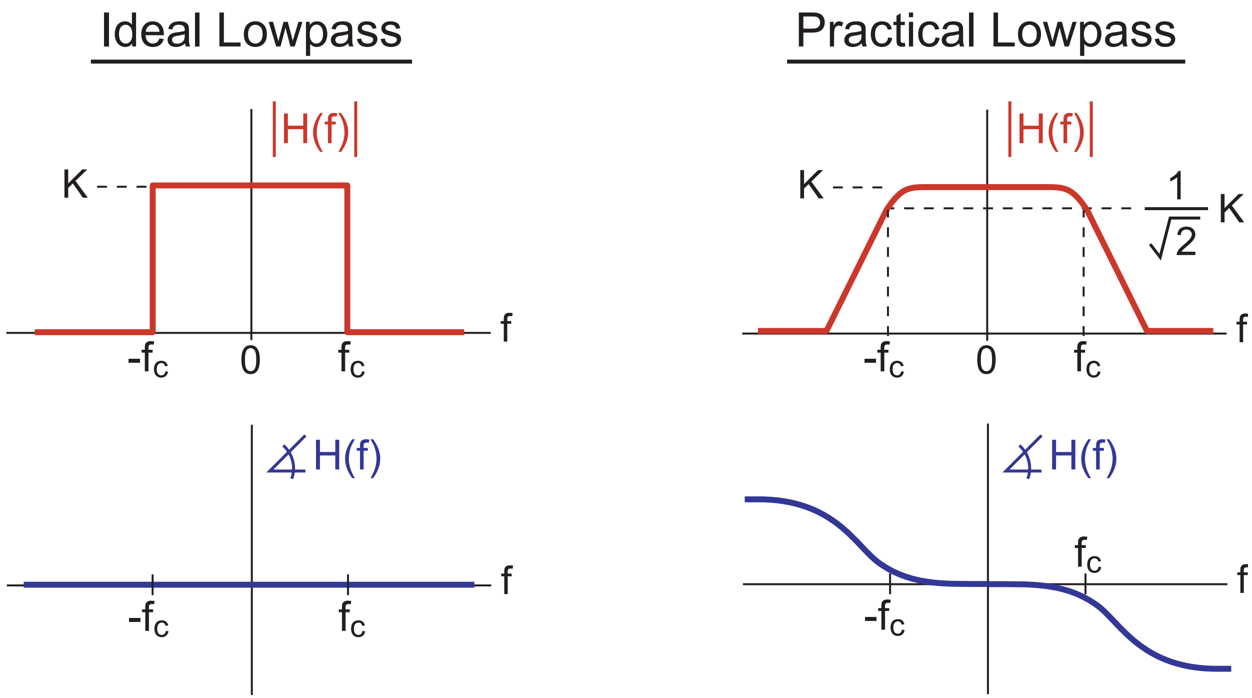
Why is Bandpass Filtering Useful?

- Allows removal of interfering signals
  - Highpass filtering would be of limited use here
- Typically higher complexity implementation than with lowpass or highpass filters
  - Many RF systems such as cell phones use specialized components called *SAW filters* to achieve bandpass filtering



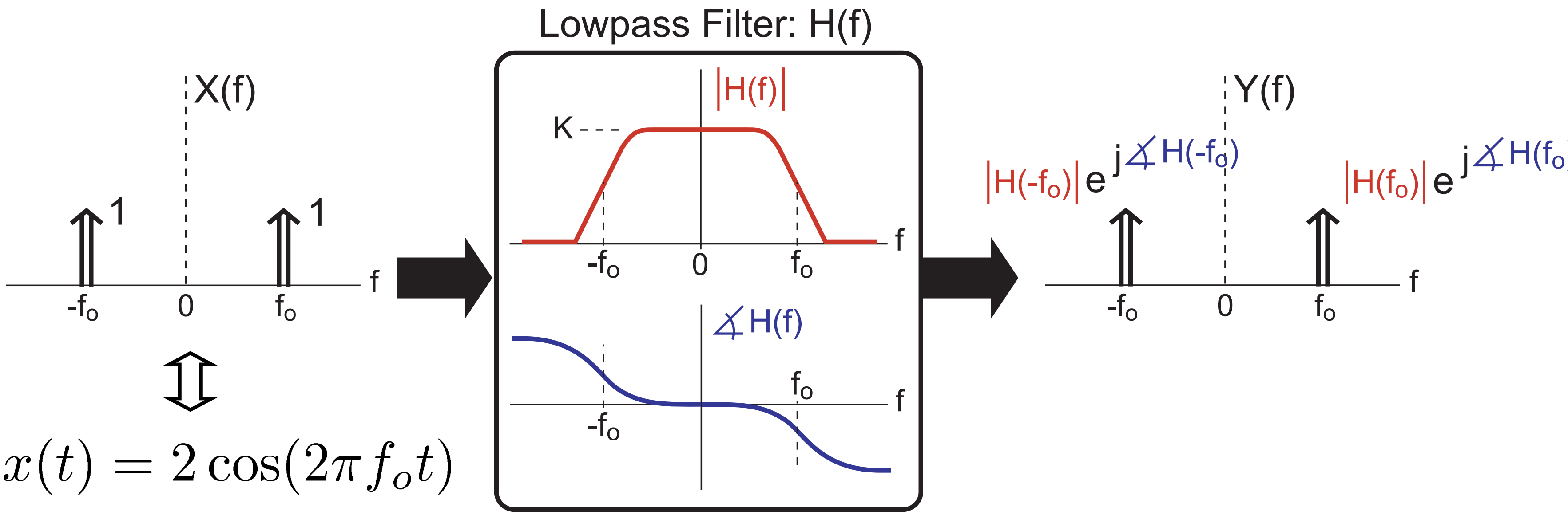
- Only *desired* RF portion remains
  - Interferer eliminated

A More Formal Treatment of Filters



- **An ideal filter would have a "brickwall" magnitude response and zero phase response**
  - Practical filters have a more gradual magnitude *rolloff* and a non-zero phase response
- **Design of the filter usually focuses on getting a reasonable magnitude rolloff with a specified cutoff frequency f_c (i.e., filter *bandwidth*)**

Response of Filter to Input Cosine



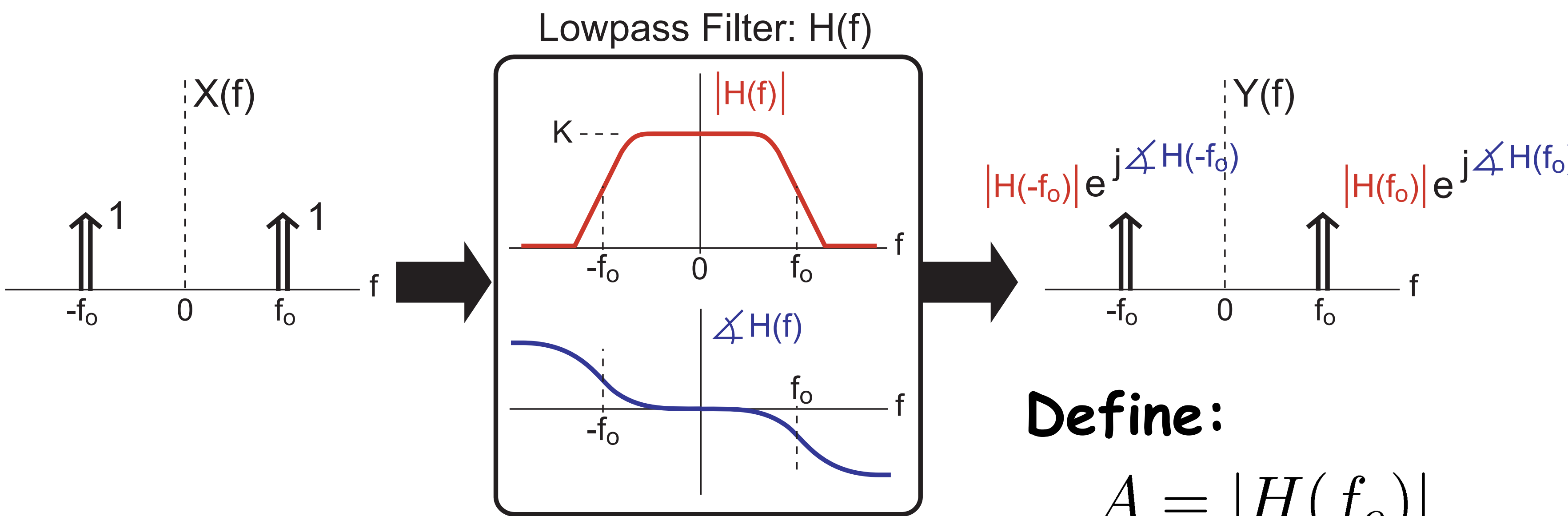
$$x(t) = 2\cos(2\pi f_o t)$$

- **Fourier transform analysis:**

$$Y(f) = H(f)X(f)$$

- **Key properties of practical filters**
  - **Magnitude response is even:** $|H(f_o)| = |H(-f_o)|$
  - **Phase response is odd:** $\angle H(f_o) = -\angle H(-f_o)$
  - **We'll explain why this is true in 6.003 …**

Compute Fourier Transform



Define:

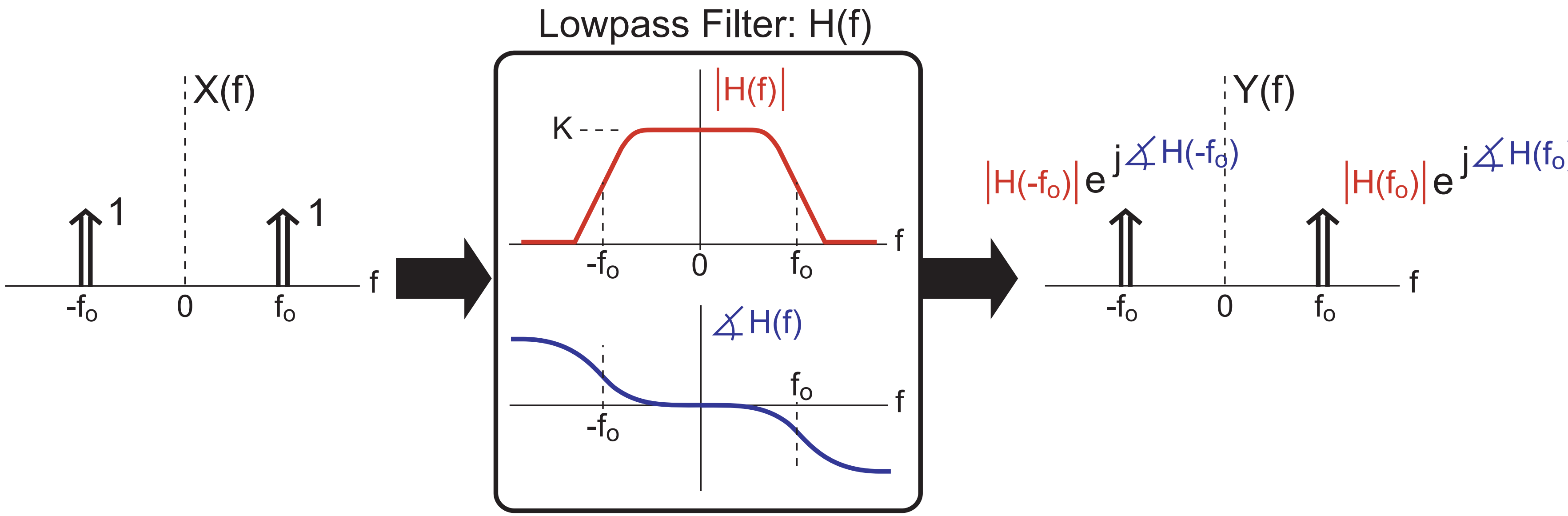
$$A = |H(f_o)|$$

$$\Phi = \angle H(f_o)$$

- **Fourier transform of output:**

$$Y(f) = H(f)X(f)$$
$$= Ae^{-j\Phi}\delta(f+f_o) + Ae^{j\Phi}\delta(f-f_o)$$
$$= A\cos(\Phi)(\delta(f+f_o) + \delta(f-f_o))$$
$$-A\sin(\Phi)(j\delta(f+f_o) - j\delta(f-f_o))$$

Compute Time-Domain Response

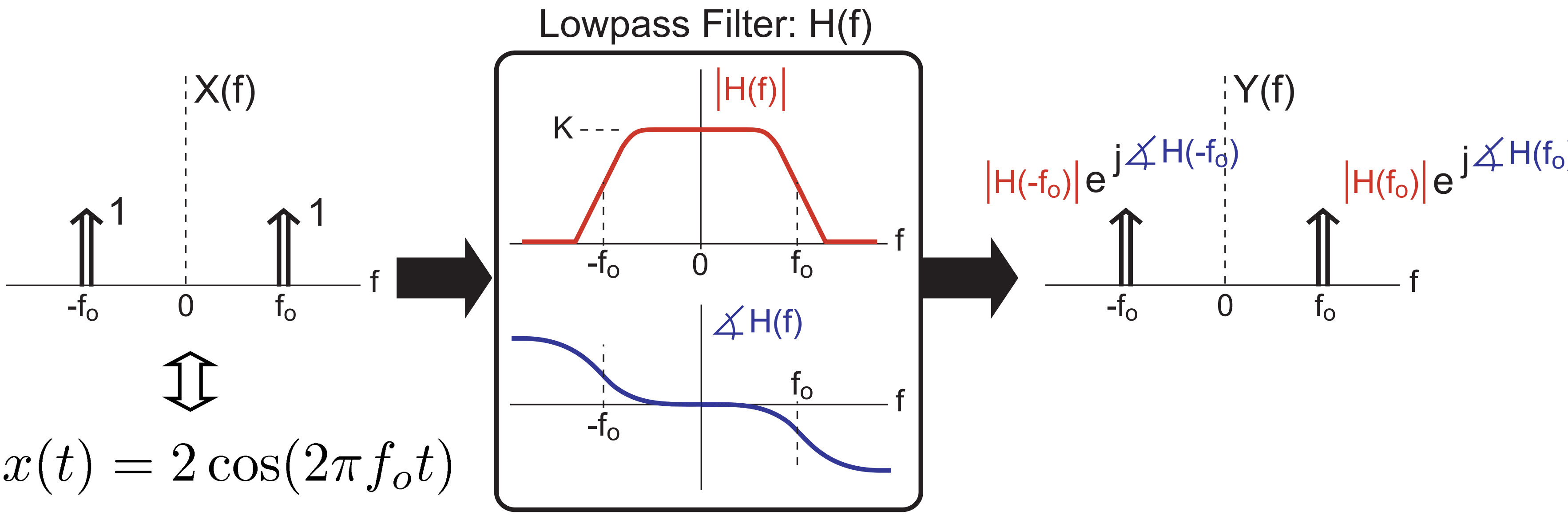


$$Y(f) = A\cos(\Phi)(\delta(f+f_o) + \delta(f-f_o))$$
$$-A\sin(\Phi)(j\delta(f+f_o) - j\delta(f-f_o))$$

- **Transform back to time domain:**

$$y(t) = A\cos(\Phi)2\cos(2\pi f_o t) - A\sin(\Phi)2\sin(2\pi f t)$$
$$= 2A\cos(2\pi f_o t + \Phi)$$
$$= |H(f_o)|2\cos(2\pi f_o t + \angle H(f_o))$$

Key Observations of Filter Response

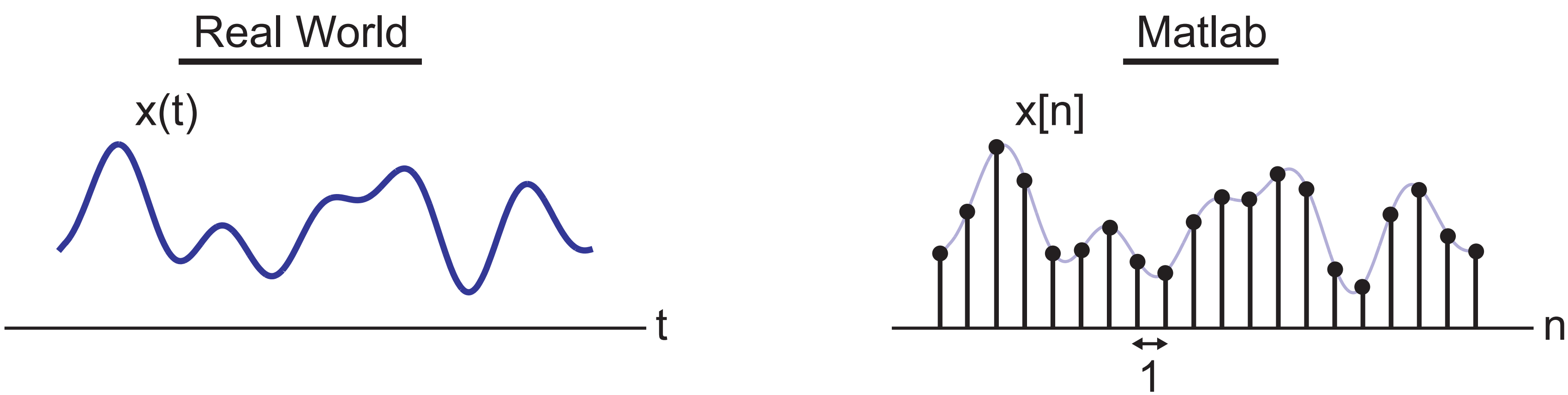


$$x(t) = 2\cos(2\pi f_o t)$$

$$y(t) = |H(f_o)| 2\cos(2\pi f_o t + \angle H(f_o))$$

- Input cosine wave is *scaled* in amplitude and *phase-shifted* in time
  - Scale factor set by magnitude of *H(f)* at $f=f_o$
  - Phase shift set by phase of *H(f)* at $f=f_o$
- We typically focus only on the *magnitude* of the *frequency response, H(f),* of the filter

Designing and Using Filters Within Matlab



- Our lab exercises will have you design and use filters in Matlab
  - Matlab will interface to the USRP board in order to receive "real world" signals from the antenna
- Matlab framework is based on *discrete-time sequences* (which are indexed on integer values)
  - Correspond to *samples* of corresponding real world signals (which are *continuous-time* in nature)

We need another Fourier analysis tool

The Discrete-Time Fourier Transform

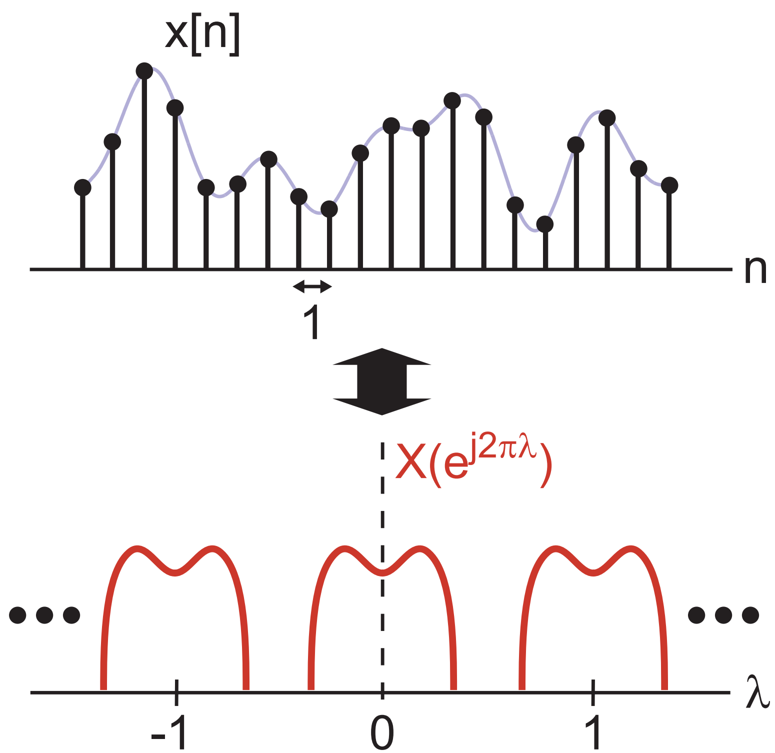
- Allows us to deal with *non-periodic, discrete-time* signals
- Frequency domain signal is *periodic* in this case

$$x[n] \Leftrightarrow X(e^{j2\pi\lambda})$$

- **Where:**

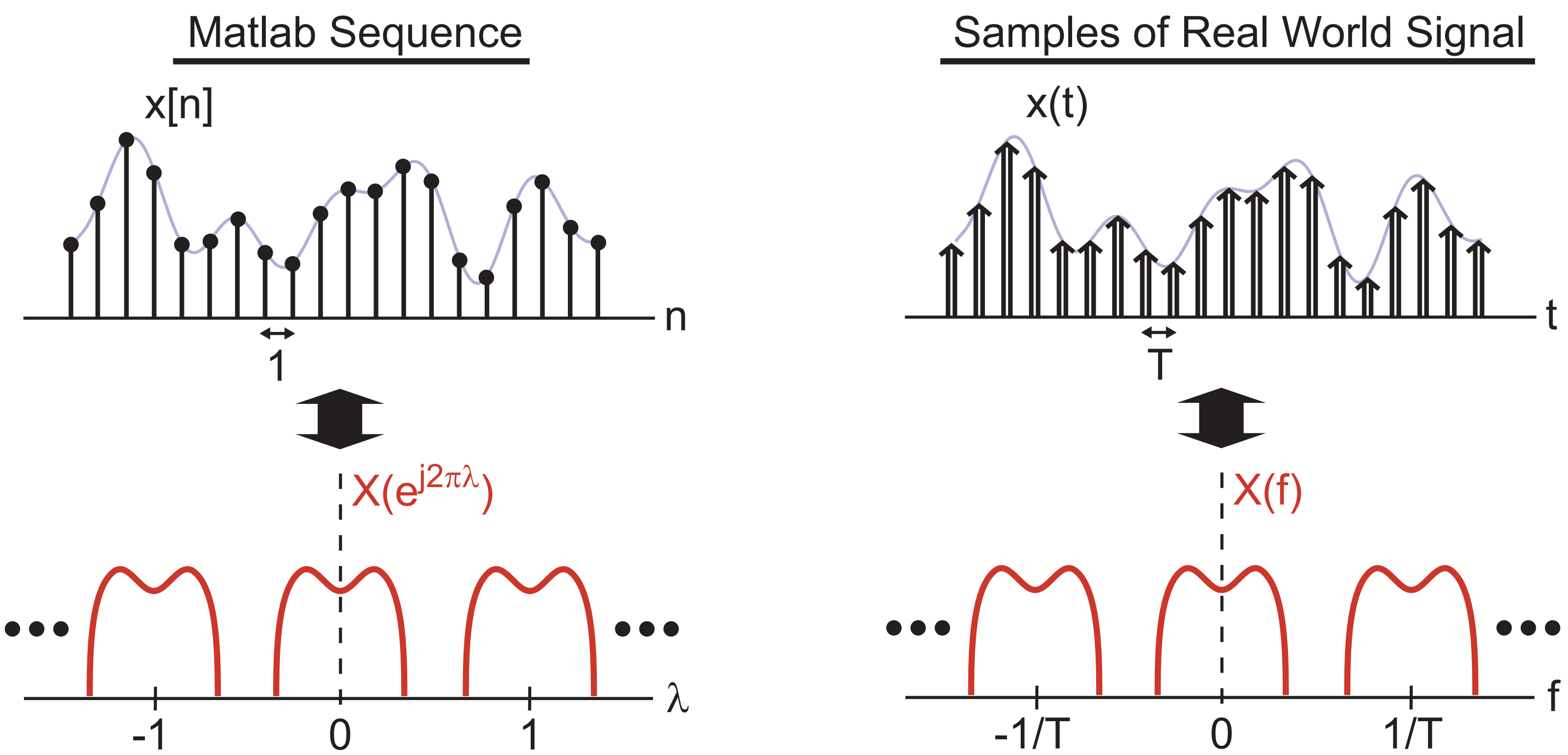
$$x[n] = \int_{-1/2}^{1/2} X(e^{j2\pi\lambda}) e^{j2\pi\lambda n} d\lambda$$

$$X(e^{j2\pi\lambda}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j2\pi\lambda n}$$



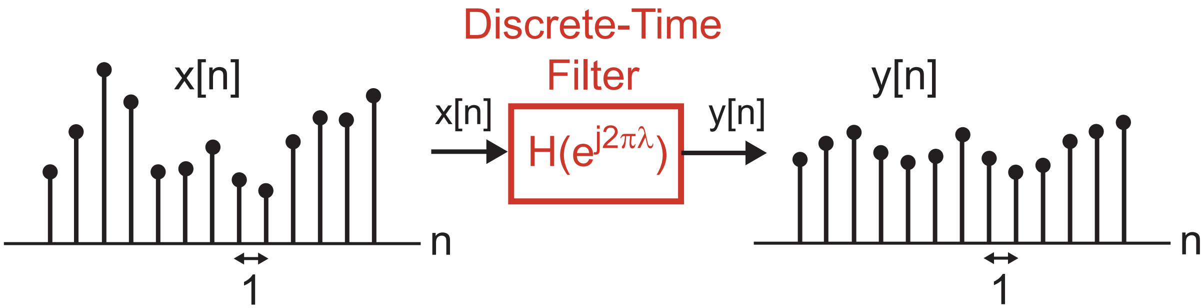
Note: ***fft*** **function in Matlab used to compute** ***DTFT***

Relating to Samples of `Real World' Signals



- Samples of a continuous-time signal with sample period *T* leads to frequency domain signal with period *1/T*
  - We simply scale frequency axis of *fft* in Matlab
- We will say much more about *sampling* later …

Filters Within Matlab



- Implemented as *difference equations*
  - Current output, y[n], depends on weighted values of previous output samples and current and previous input samples, x[n]

$$y[n] = \sum_{k=1}^{M} a_k y[n-k] + \sum_{k=0}^{N} b_k x[n-k]$$

- Group *a* and *b* coefficients as vectors:

$$\mathbf{a} = [a_0\ a_1\ \cdots\ a_M], \quad \mathbf{b} = [b_0\ b_1\ \cdots\ b_N]$$

- Execute filter using the *filter* command:

$$y = \mathrm{filter}(b, a, x);$$

Impact of Delay on DTFT

- **Consider a signal that is a delayed version of another signal:**

$$y[n] = x[n - n_o]$$

- **Compute DTFT of *y[n]***

$$Y(e^{j2\pi\lambda}) = \sum_{n=-\infty}^{\infty} y[n] e^{-j2\pi\lambda n}$$

$$= \sum_{n=-\infty}^{\infty} x[n - n_o] e^{-j2\pi\lambda n}$$

$$= \sum_{m=-\infty}^{\infty} x[m] e^{-j2\pi\lambda(m+n_o)} \quad (\text{where } m = n - n_o)$$

$$= e^{-j2\pi\lambda n_o} \sum_{m=-\infty}^{\infty} x[m] e^{-j2\pi\lambda m} = \boxed{e^{-j2\pi\lambda n_o} X(e^{j2\pi\lambda})}$$

Compute Filter Response using DTFT

$$y[n] = \sum_{k=1}^{M} a_k y[n-k] + \sum_{k=0}^{N} b_k x[n-k]$$

- **Make use of the time shift property:**

$$Y(e^{j2\pi\lambda}) = \sum_{k=1}^{M} a_k e^{-j2\pi\lambda k} Y(e^{j2\pi\lambda}) + \sum_{k=0}^{N} b_k e^{-j2\pi\lambda k} X(e^{j2\pi\lambda})$$

$$\Rightarrow \quad Y(e^{j2\pi\lambda}) \left(1 - \sum_{k=1}^{M} a_k e^{-j2\pi\lambda k}\right) = X(e^{j2\pi\lambda}) \sum_{k=0}^{N} b_k e^{-j2\pi\lambda k}$$

- **Filter response is simply ratio of output over input:**

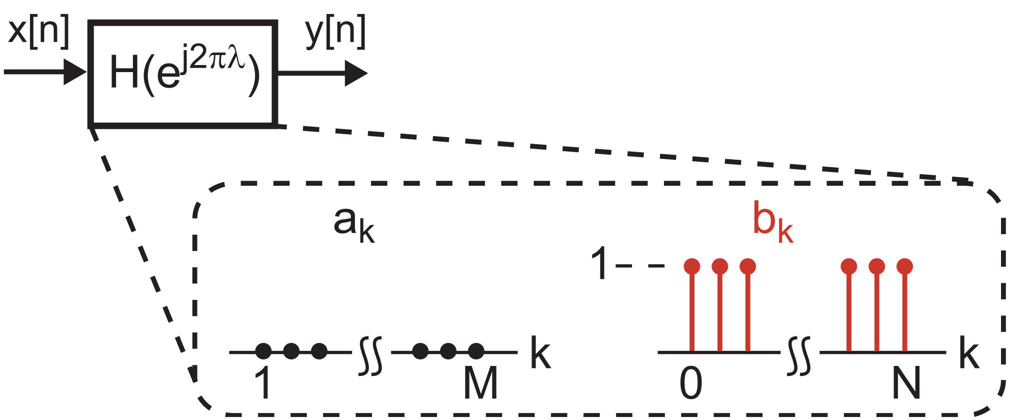
$$H(e^{j2\pi\lambda}) = \frac{Y(e^{j2\pi\lambda})}{X(e^{j2\pi\lambda})} = \frac{\sum_{k=0}^{N} b_k e^{-j2\pi\lambda k}}{1 - \sum_{k=1}^{M} a_k e^{-j2\pi\lambda k}}$$

FIR Filters

- **Finite Impulse Response (FIR) filters use only *b* coefficients in their implementation**

$$y[n] = \sum_{k=0}^{N} b_k x[n-k] \quad \Rightarrow \quad H(e^{j2\pi\lambda}) = \sum_{k=0}^{N} b_k e^{-j2\pi\lambda k}$$

- **Example:**

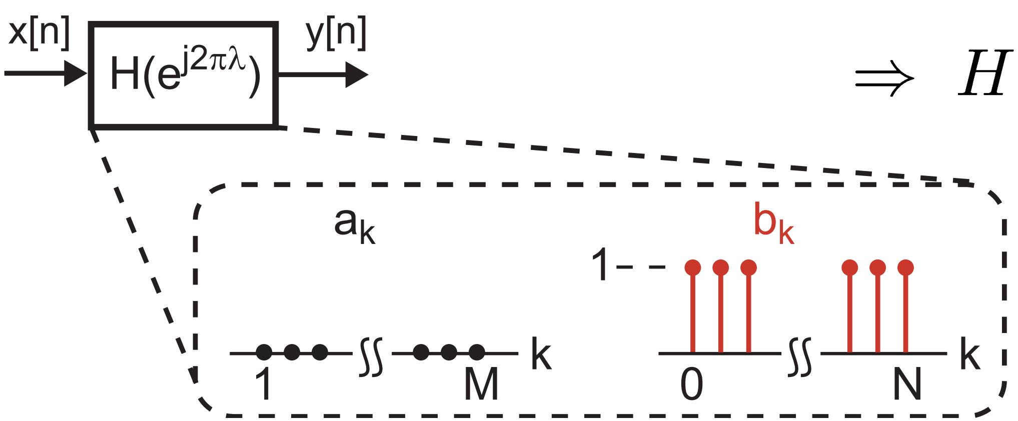


Note:

$$\sum_{k=0}^{N} r^k = \frac{1-r^{N+1}}{1-r}$$

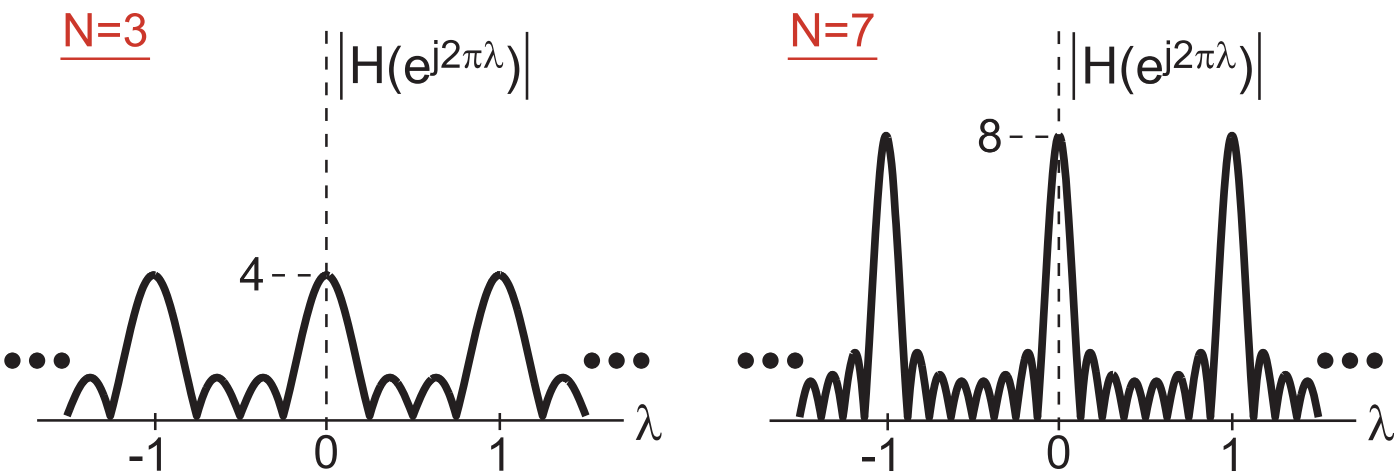
$$H(e^{j2\pi\lambda}) = \sum_{k=0}^{N} 1 e^{-j2\pi\lambda k} = \frac{1-e^{-j2\pi\lambda(N+1)}}{1-e^{-j2\pi\lambda}}$$

Filter Order for FIR Filters



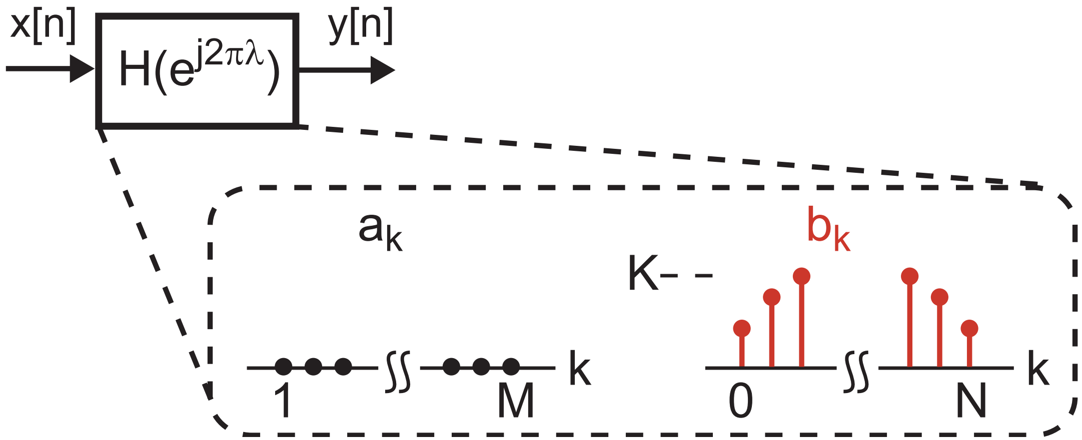
$$\Rightarrow \quad H(e^{j2\pi\lambda}) = \frac{1 - e^{-j2\pi\lambda(N+1)}}{1 - e^{-j2\pi\lambda}}$$

- **Consider two different values for *N***



- **Higher *N* leads to steeper filter response**
  - **We refer to *N* as the *order* of the filter**

FIR Filter Design in Matlab



- Lowpass, highpass, and bandpass filters can be realized by appropriately scaling the relative value of the *b* coefficients
  - Higher order (i.e., higher *N*) leads to steeper responses
- Perform FIR filter design using *fir1* command
- Frequency response observed with *freqz* command

See Prelab portion of Lab 3 for details …

Summary

- Filters can generally be classified according to
  - Lowpass, highpass, bandpass operation
  - Bandwidth and order of filter
- Given a cosine input to a filter, output is:
  - Scaled in amplitude by magnitude of filter frequency response
  - Shifted in phase by phase of filter frequency response
- Matlab operates on discrete-time signals
  - Use DTFT for analytical analysis
  - Use commands such as *fir1*, *freqz*, and *filter* for design and implementation of FIR filters
- Next lecture: introduce I/Q modulation and further discuss continuous-time filtering