Lab lecture 5

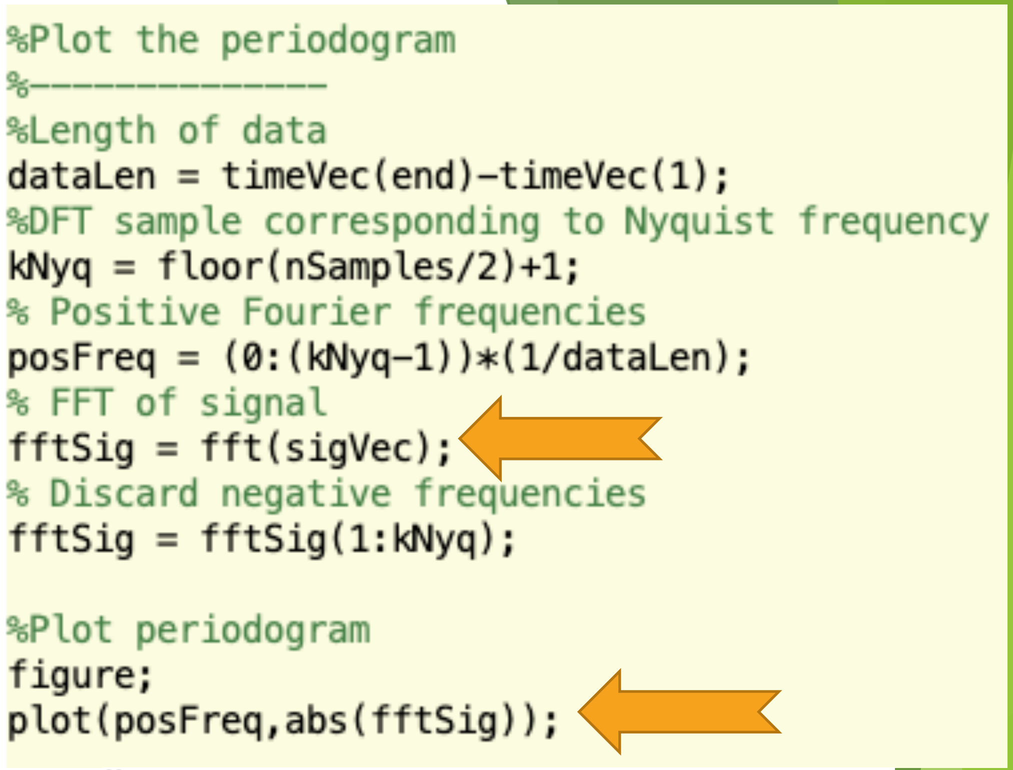
Learning objectives

- Compute FFTs and understand how the FFT output in Matlab is arranged
- FIR filter design and application

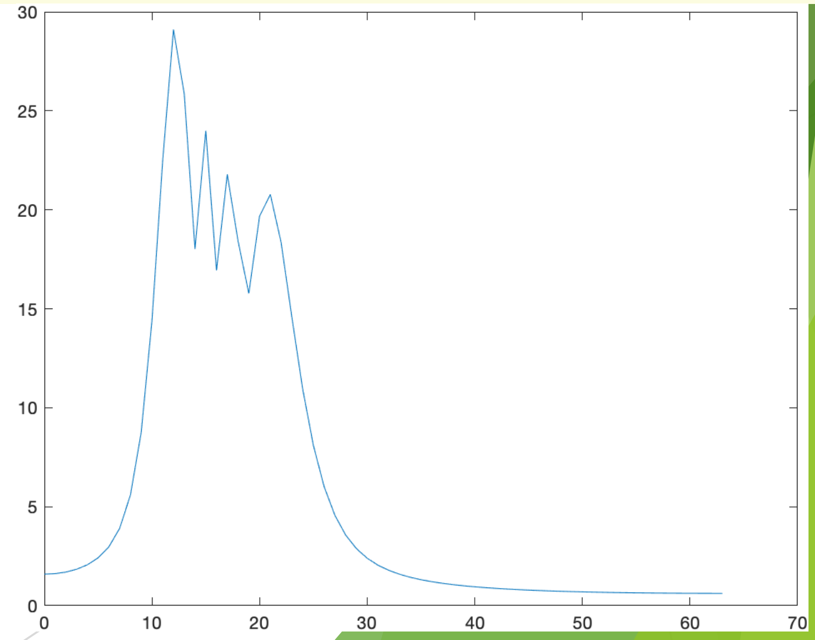
FFT

Periodogram

- For your assigned signal …
- Task: Modify the `test<funcname>.m` script that you wrote to plot the periodogram of the signal
  - Example: see DATASCIENCE_COURSE/SIGNALS/`testcrcbgenqcsig.m`)
- Periodogram: Magnitude of the FFT
- Note: The signal sampling frequency should be set to 5 times the (guessed) Nyquist frequency
- The FFT has positive frequency values in the first half of the FFT output and negative frequency values in the 2nd half
  - Since, for a real sequence, they are complex conjugates of each other, they have the same magnitudes and the negative frequency part can be discarded if we only wish to look at the periodogram

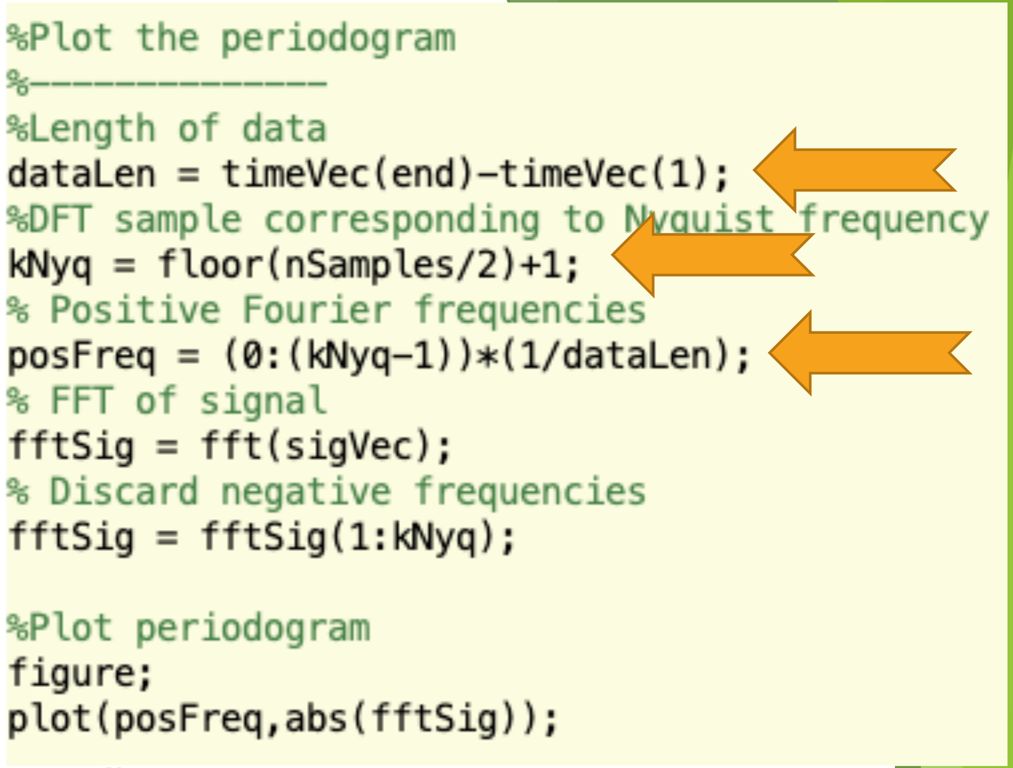


```
%Plot the periodogram
%---------------
%Length of data
dataLen = timeVec(end)-timeVec(1);
%DFT sample corresponding to Nyquist frequency
kNyq = floor(nSamples/2)+1;
% Positive Fourier frequencies
posFreq = (0:(kNyq-1))*(1/dataLen);
% FFT of signal
fftSig = fft(sigVec);
% Discard negative frequencies
fftSig = fftSig(1:kNyq);

%Plot periodogram
figure;
plot(posFreq,abs(fftSig));
```

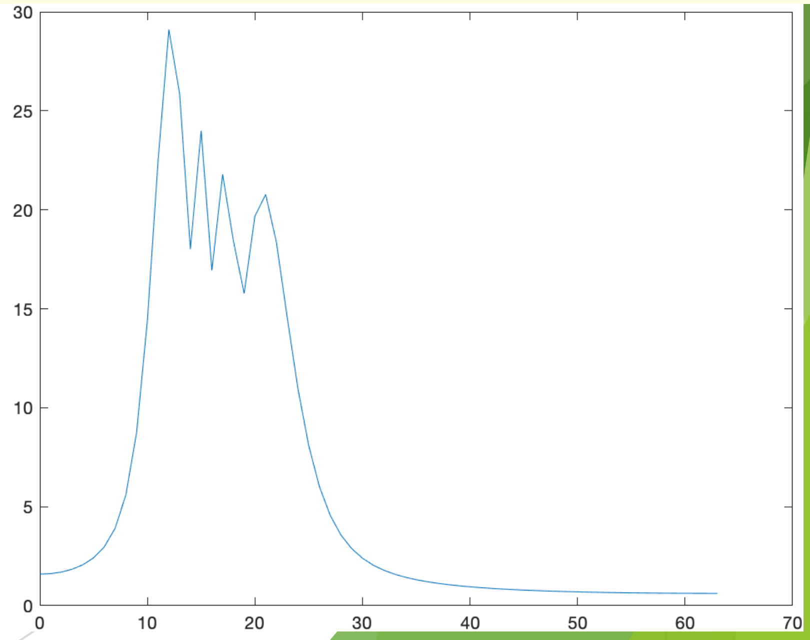


Periodogram

- For your assigned signal …
- Task: Modify the `test<funcname>.m` script that you wrote to plot the periodogram of the signal
  - Example: see DATASCIENCE_COURSE/SIGNALS/`testcrcbgenqcsig.m`)
- Periodogram: Magnitude of the FFT
- Note: The signal sampling frequency should be set to 5 times the (guessed) Nyquist frequency
- The FFT has positive frequency values in the first half of the FFT output and negative frequency values in the 2nd half
  - Since, for a real sequence, they are complex conjugates of each other, they have the same magnitudes and the negative frequency part can be discarded if we only wish to look at the periodogram

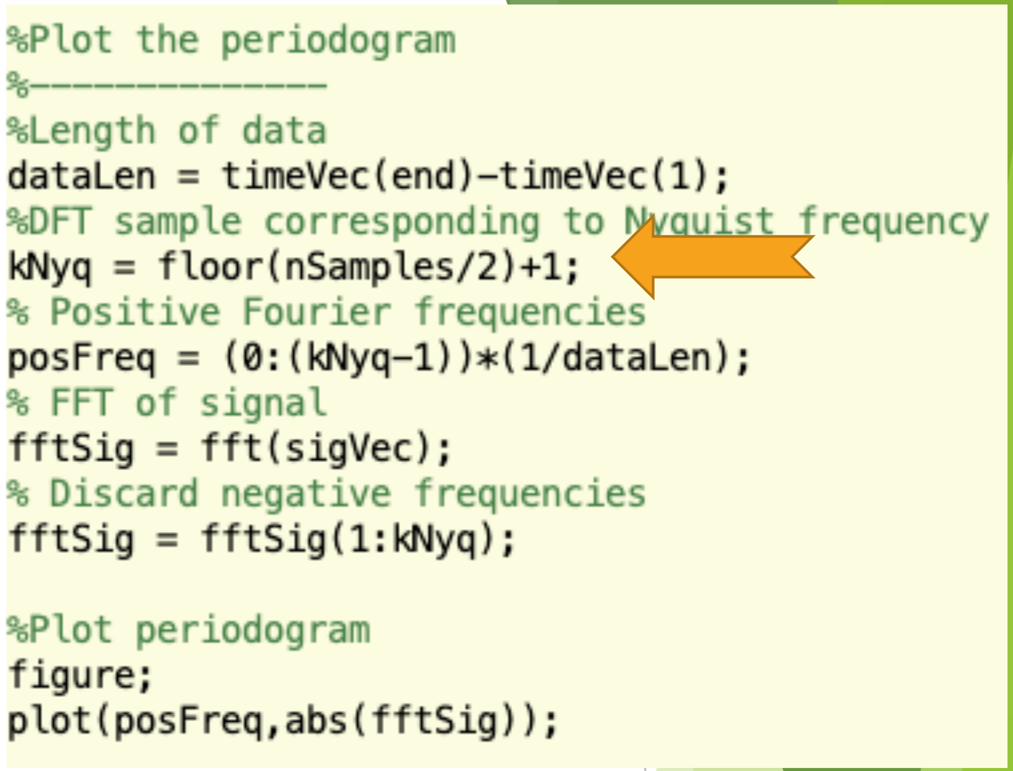


```
%Plot the periodogram
%--------------
%Length of data
dataLen = timeVec(end)-timeVec(1);
%DFT sample corresponding to Nyquist frequency
kNyq = floor(nSamples/2)+1;
% Positive Fourier frequencies
posFreq = (0:(kNyq-1))*(1/dataLen);
% FFT of signal
fftSig = fft(sigVec);
% Discard negative frequencies
fftSig = fftSig(1:kNyq);

%Plot periodogram
figure;
plot(posFreq,abs(fftSig));
```



Frequencies in a DFT

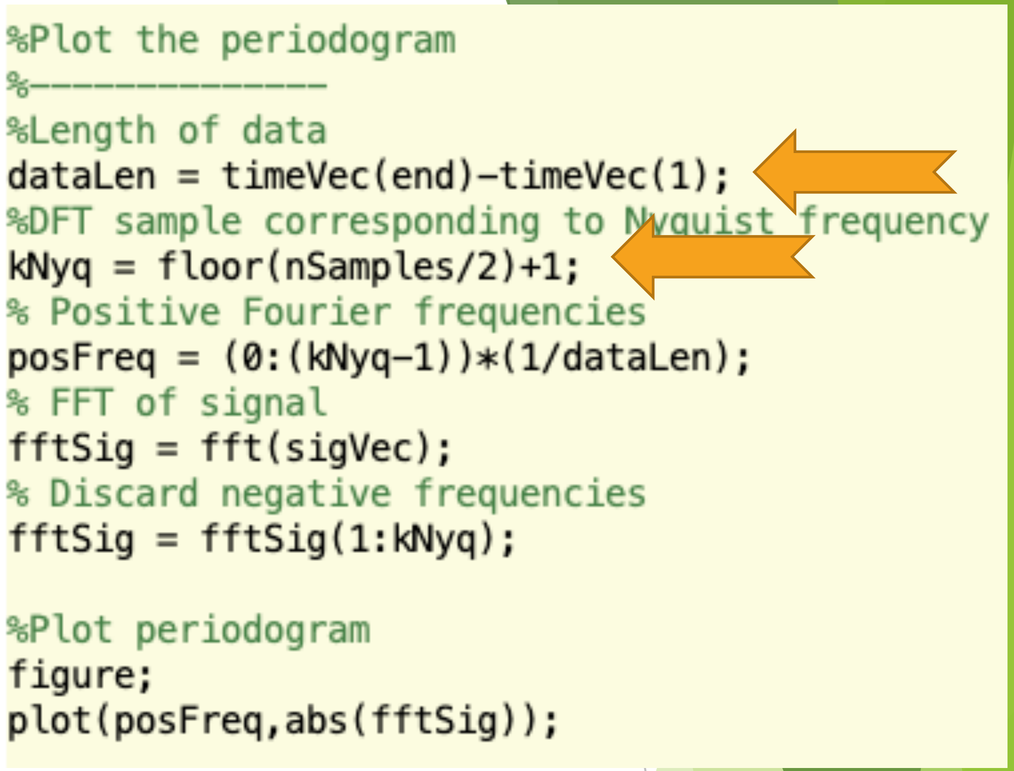
- Task: Generate the correct frequency values for your periodogram plots
- For the FFT of a sequence with N samples:
  - Positive frequency components of FFT go from *index number*:
    - 1 to `floor(N/2)+1`
  - Negative frequency components go from index number:
    - `floor(N/2)+2` to N



```
%Plot the periodogram
%---------------
%Length of data
dataLen = timeVec(end)-timeVec(1);
%DFT sample corresponding to Nyquist frequency
kNyq = floor(nSamples/2)+1;
% Positive Fourier frequencies
posFreq = (0:(kNyq-1))*(1/dataLen);
% FFT of signal
fftSig = fft(sigVec);
% Discard negative frequencies
fftSig = fftSig(1:kNyq);

%Plot periodogram
figure;
plot(posFreq,abs(fftSig));
```

Frequencies in a DFT

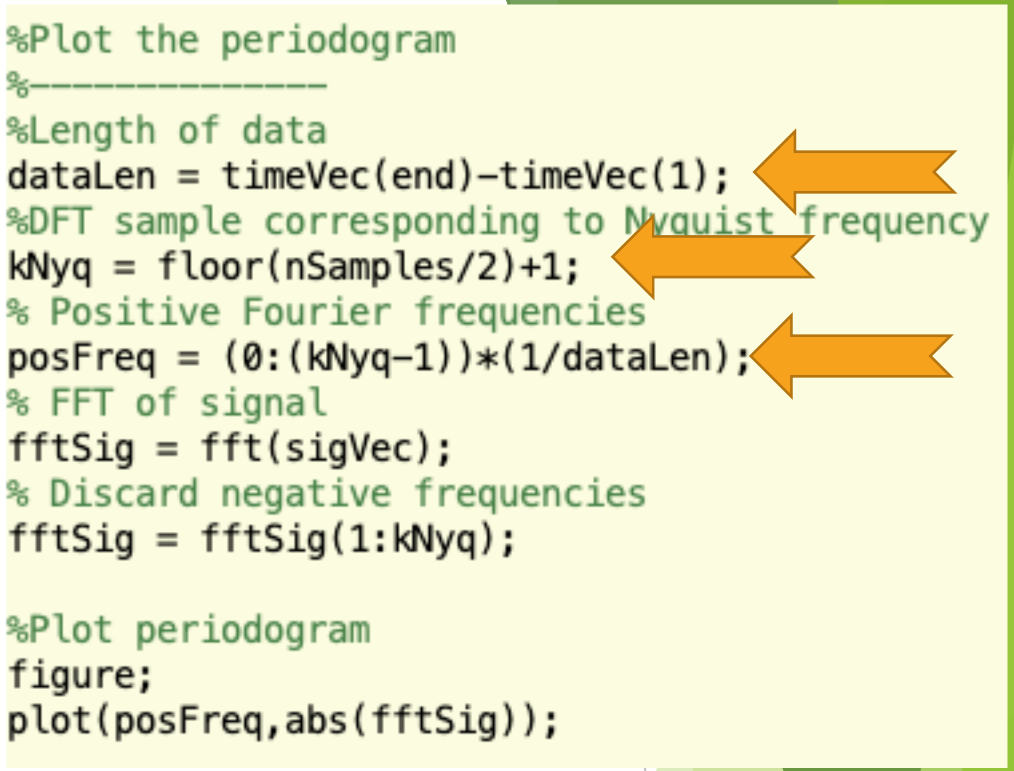
- Task: Generate the correct frequency values for your periodogram plots
- For the FFT of a sequence with N samples:
  - Positive frequency components of FFT go from index number:
    - 1 to `floor(N/2)+1`
  - Negative frequency components go from index number:
    - `floor(N/2)+2` to N
- DFT Frequency spacing is $1/T$ where T is the duration of the data



```
%Plot the periodogram
%--------------
%Length of data
dataLen = timeVec(end)-timeVec(1);
%DFT sample corresponding to Nyquist frequency
kNyq = floor(nSamples/2)+1;
% Positive Fourier frequencies
posFreq = (0:(kNyq-1))*(1/dataLen);
% FFT of signal
fftSig = fft(sigVec);
% Discard negative frequencies
fftSig = fftSig(1:kNyq);

%Plot periodogram
figure;
plot(posFreq,abs(fftSig));
```

Frequencies in a DFT

- Task: Generate the correct frequency values for your periodogram plots
- For the FFT of a sequence with N samples:
  - Positive frequency components of FFT go from index number:
    - 1 to `floor(N/2)+1`
  - Negative frequency components go from index number:
    - `floor(N/2)+2` to N
- DFT Frequency spacing is $1/T$ where T is the duration of the data

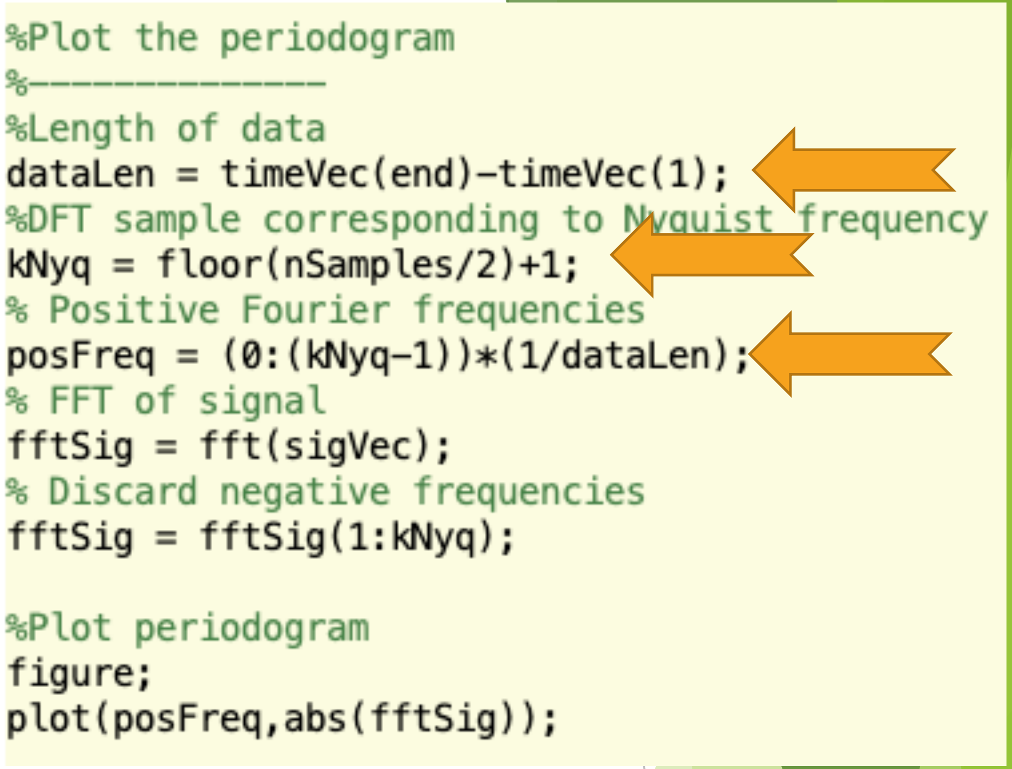


```
%Plot the periodogram
%--------------
%Length of data
dataLen = timeVec(end)-timeVec(1);
%DFT sample corresponding to Nyquist frequency
kNyq = floor(nSamples/2)+1;
% Positive Fourier frequencies
posFreq = (0:(kNyq-1))*(1/dataLen);
% FFT of signal
fftSig = fft(sigVec);
% Discard negative frequencies
fftSig = fftSig(1:kNyq);

%Plot periodogram
figure;
plot(posFreq,abs(fftSig));
```

Frequencies in a DFT

- Task: Generate the correct frequency values for your periodogram plots
- For the FFT of a sequence with N samples:
  - Positive frequency components of FFT go from index number:
    - 1 to `floor(N/2)+1`
  - Negative frequency components go from index number:
    - `floor(N/2)+2` to N
- DFT Frequency spacing is $1/T$ where T is the duration of the data
- Positive frequencies:

$$f_k = k \times \frac{1}{T}, \qquad k = 0,1,\ldots,floor(\frac{N}{2})$$



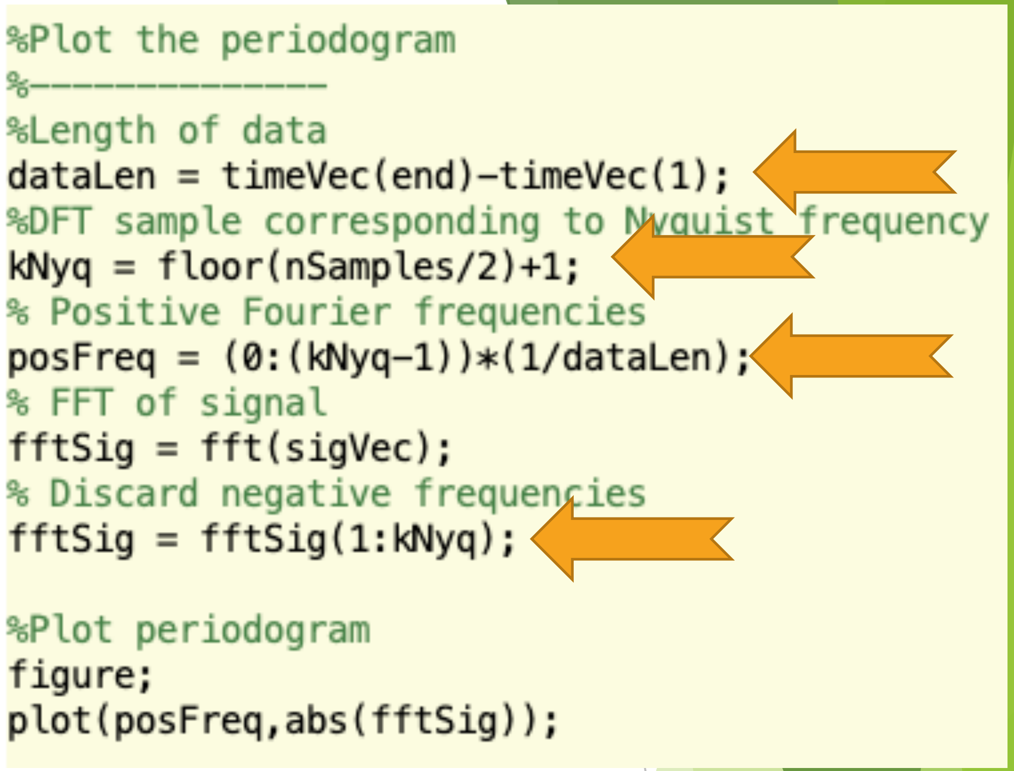
```
%Plot the periodogram
%---------------
%Length of data
dataLen = timeVec(end)-timeVec(1);
%DFT sample corresponding to Nyquist frequency
kNyq = floor(nSamples/2)+1;
% Positive Fourier frequencies
posFreq = (0:(kNyq-1))*(1/dataLen);
% FFT of signal
fftSig = fft(sigVec);
% Discard negative frequencies
fftSig = fftSig(1:kNyq);

%Plot periodogram
figure;
plot(posFreq,abs(fftSig));
```

Frequencies in a DFT

- Task: Generate the correct frequency values for your periodogram plots
- For the FFT of a sequence with N samples:
  - Positive frequency components of FFT go from index number:
    - 1 to `floor(N/2)+1`
  - Negative frequency components go from index number:
    - `floor(N/2)+2` to N
- DFT Frequency spacing is $1/T$ where T is the duration of the data
- Positive frequencies:

$$f_k = k\times\frac{1}{T}, \qquad k = 0,1,\ldots,floor(\frac{N}{2})$$

- Extract the positive frequency part of the FFT



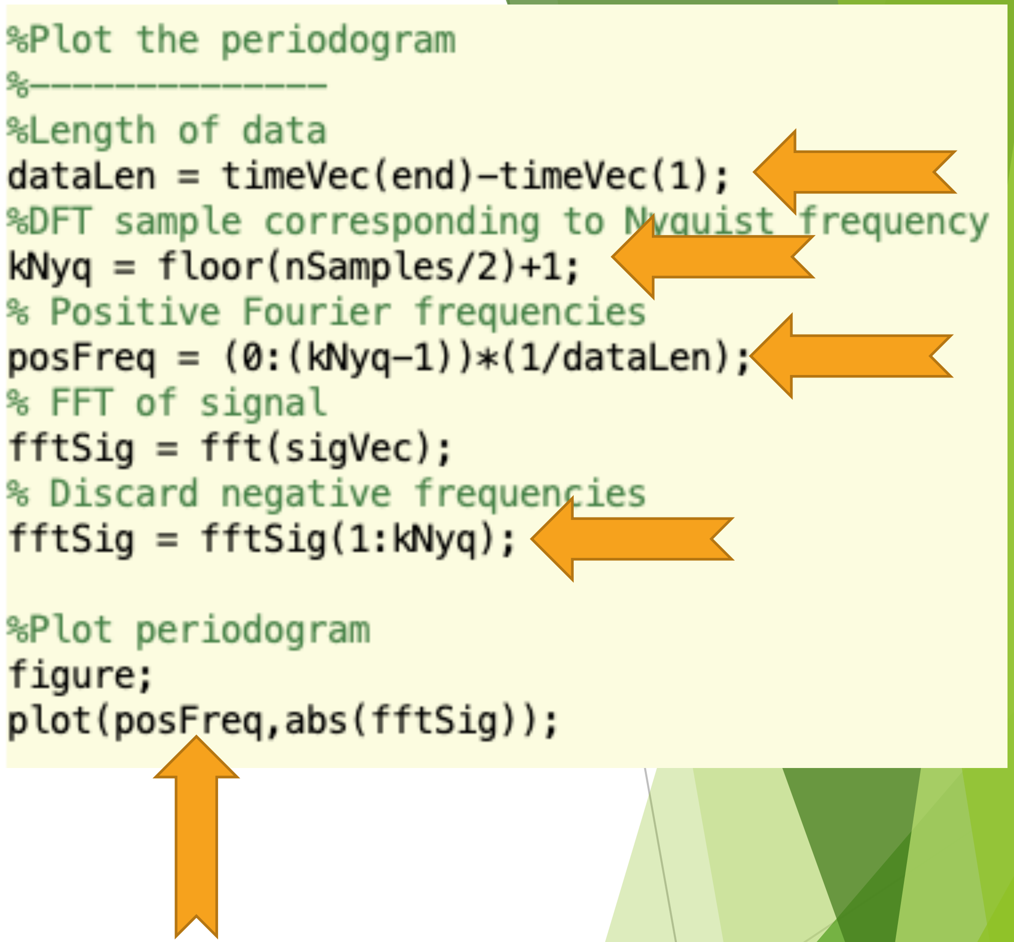
```
%Plot the periodogram
%---------------
%Length of data
dataLen = timeVec(end)-timeVec(1);
%DFT sample corresponding to Nyquist frequency
kNyq = floor(nSamples/2)+1;
% Positive Fourier frequencies
posFreq = (0:(kNyq-1))*(1/dataLen);
% FFT of signal
fftSig = fft(sigVec);
% Discard negative frequencies
fftSig = fftSig(1:kNyq);

%Plot periodogram
figure;
plot(posFreq,abs(fftSig));
```

Frequencies in a DFT

- Task: Generate the correct frequency values for your periodogram plots
- For the FFT of a sequence with N samples:
  - Positive frequency components of FFT go from index number:
    - 1 to `floor(N/2)+1`
  - Negative frequency components go from index number:
    - `floor(N/2)+2` to N
- DFT Frequency spacing is $1/T$ where T is the duration of the data
- Positive frequencies:

$$f_k = k\times\frac{1}{T}, \qquad k = 0,1,\ldots,floor(\frac{N}{2})$$

- Extract the positive frequency part of the FFT and plot



```
%Plot the periodogram
%--------------
%Length of data
dataLen = timeVec(end)-timeVec(1);
%DFT sample corresponding to Nyquist frequency
kNyq = floor(nSamples/2)+1;
% Positive Fourier frequencies
posFreq = (0:(kNyq-1))*(1/dataLen);
% FFT of signal
fftSig = fft(sigVec);
% Discard negative frequencies
fftSig = fftSig(1:kNyq);

%Plot periodogram
figure;
plot(posFreq,abs(fftSig));
```

Digital filtering

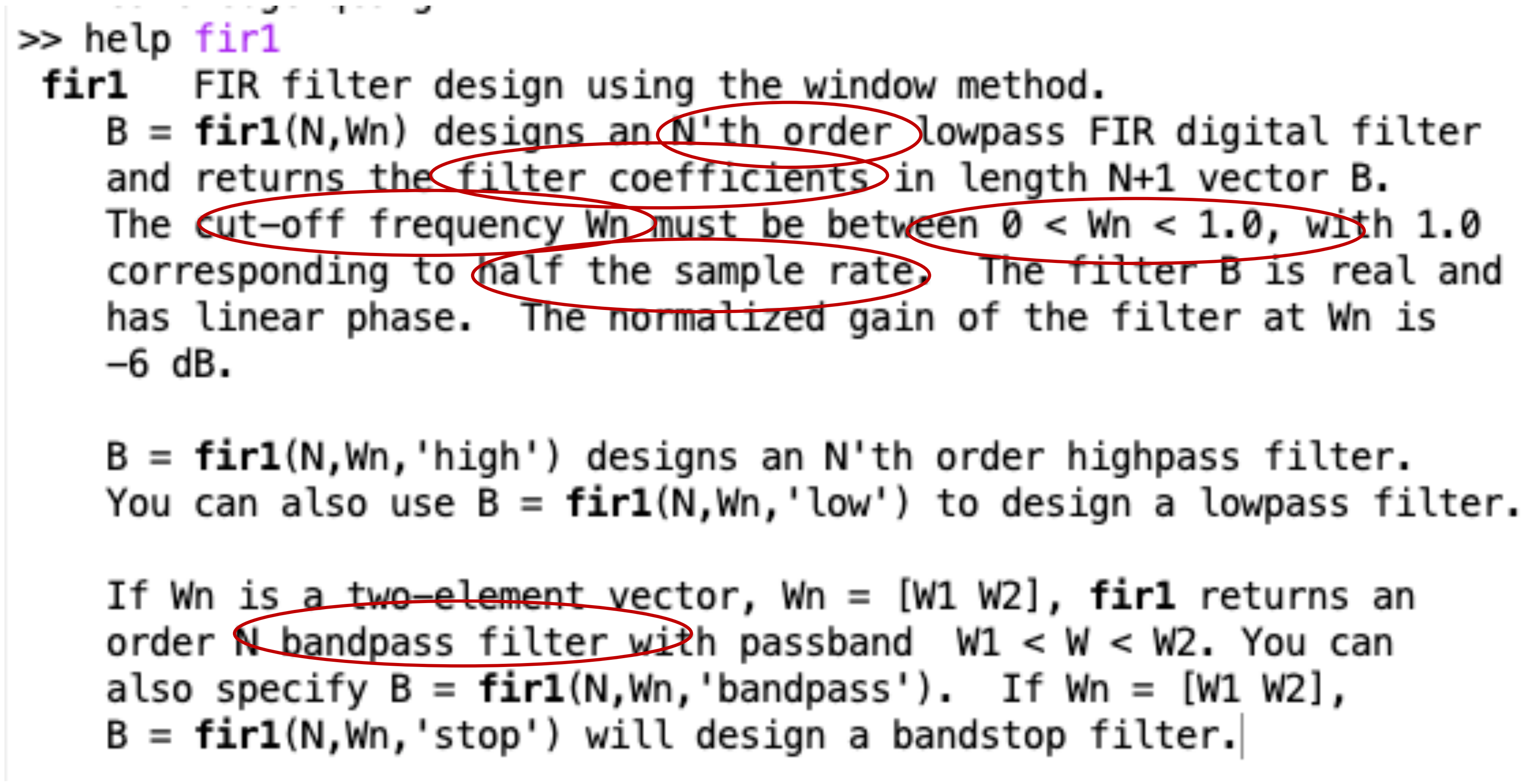
Filtering

- We will design a digital for a given specification of the filter transfer function
- The main thing about Matlab’s filter design functions is that frequencies are specified relative to half the sampling frequency of the input data

```
>> help fir1
```

FIR filter design using the window method. B = fir1(N,Wn) designs an N'th order lowpass FIR digital filter and returns the filter coefficients in length N+1 vector B. The cut-off frequency Wn must be between 0 < Wn < 1.0, with 1.0 corresponding to half the sample rate.

- `fir1` is the simplest FIR filter design method (`fir2` and `firls` are more sophisticated methods): Good for designing standard low pass, high pass and bandpass filters.



```
>> help fir1
 fir1   FIR filter design using the window method.
    B = fir1(N,Wn) designs an N'th order lowpass FIR digital filter
    and returns the filter coefficients in length N+1 vector B.
    The cut-off frequency Wn must be between 0 < Wn < 1.0, with 1.0
    corresponding to half the sample rate.  The filter B is real and
    has linear phase.  The normalized gain of the filter at Wn is
    -6 dB.

    B = fir1(N,Wn,'high') designs an N'th order highpass filter.
    You can also use B = fir1(N,Wn,'low') to design a lowpass filter.

    If Wn is a two-element vector, Wn = [W1 W2], fir1 returns an
    order N bandpass filter with passband  W1 < W < W2. You can
    also specify B = fir1(N,Wn,'bandpass').  If Wn = [W1 W2],
    B = fir1(N,Wn,'stop') will design a bandstop filter.
```

Filtering

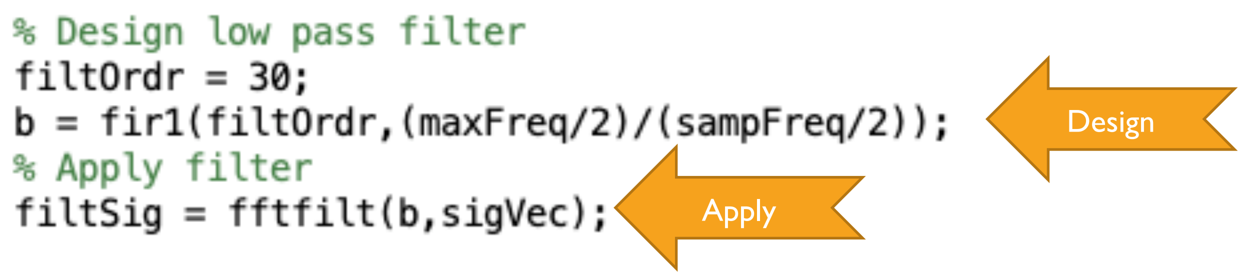
- Task: Generate a signal containing the **sum of three sinusoids** with the following parameters
- Number of samples: 2048
- Sampling frequency: 1024 Hz
  - What is the maximum frequency of the discrete time sinusoid you can generate with this sampling frequency? Hint: Nyquist theorem

	$\bar{s}_1$	$\bar{s}_2$	$\bar{s}_3$
A	10	5	2.5
f_0	100	200	300
ϕ_0	0	$\pi/6$	$\pi/4$

- The signal to be generated: $\bar{s} = \bar{s}_1 + \bar{s}_2 + \bar{s}_3$

Filtering

- Task: Use Matlab's `fir1` function to design 3 different filters such that filter #i allows only $\bar{s}_i$ to pass through
  - You have to design all of the above three types of filters with the appropriate frequency limits
  - Study the script DSP/LowPassFilterDemo.m to see how a low pass filter is designed and how it is applied



```
% Design low pass filter
filtOrdr = 30;
b = fir1(filtOrdr,(maxFreq/2)/(sampFreq/2));
% Apply filter
filtSig = fftfilt(b,sigVec);
```

Filtering

- Task: Use Matlab's `fir1` function to design 3 different filters such that filter #i allows only $\bar{s}_i$ to pass through
  - You have to design all of the above three types of filters with the appropriate frequency limits
  - Study the script DSP/LowPassFilterDemo.m to see how a low pass filter is designed and how it is applied (to the quadratic chirp signal)
- Task: Write a Matlab script that applies each filter to the signal $\bar{s}$ and plots the corresponding periodograms of the input (i.e., $\bar{s}$) and the filter output
  - Learn about the `subplot` function
  - Make the plots of the input and output periodograms as the top and bottom panels of the same figure