Gary He

Quiz 2: Digital Signal Processing

Prof. Jordan C. Hanson

April 9, 2025

1. (a) Derive the Fourier series of a square wave, listing which coefficients are non-zero. (b) Write an `octave` script that creates a time series corresponding to the Fourier series, with a fundamental frequency of 1.4 MHz and a maximum time $T_{\max}$ corresponding to about 10 periods. (c) Compute the DFT of the time series. (d) Using the `octave` routines `figure()` and `subplot()`, plot the time series in one graph, and in the same figure, plot the magnitude of the DFT in another graph.

2. Do you observe the **Gibbs effect** in your Fourier series in the previous exercise? Why or why not?

3. (a) Define an $N = 1000$ sample $\delta[n]$ signal in an `octave` script, with the non-zero value in the first sample. (b) Compute the magnitude and phase of the DFT, and graph them versus frequency. (c) Advance the non-zero sample in the $\delta[n]$ by 100 samples, and recompute the graphs. (d) What is happening to the phase? Use the `unwrap()` function in `octave` to graph the linear relationship between phase and frequency. (e) Use the slope of the phase versus frequency to measure the group delay of the $\delta[n]$. Do you obtain the right result? (f) **Bonus:** what happens to the group delay measurement if the $\delta[n]$ signal has noise?

4. **Clipping** in DSP data. (a) Using `octave`, create a sine wave with the following properties: a sampling rate of 10 MHz, a frequency of 100 kHz, a $T_{\max}$ of 6 ms, and an amplitude of 1.0. The data should have more than 10^4 samples, so there is no need to graph it. (b) Using the `find()` function in `octave`, set all samples *greater* than 0.75 to 0.75. Set all samples *less* than -0.75 to -0.75. Here is a clue as to how this works:

   ```
   x(find(x>=0.75)) = 0.75
   ```

   The resulting signal is now **clipped**. Clipping often results when DSP data falls outside a nominal digital range. (c) Plot the magnitude of the DFT, and locate the frequency spike at 100 kHz. (d) Do you observe harmonics? In your own words, explain the spectrum of harmonics given that the signal is clipped.

5. Download the "NASDAQ closing prices 2024-2025" file from the course Moodle page. The left column has units of days (starting with April 10th, 2024), and the right column is the value of the NASDAQ stock index in US dollars. (a) Plot the data, and in the same plot, apply a 1-week moving average filter to smooth the data. (b) Note where the announcement of US tariffs marks a sharp drop in value. (c) Note where the value rises sharply after the US President announced a 90 day pause in tariffs during April 2025. (d) Does this moving average capture rapid shifts in economic policy? Why or why not? (e) **Bonus**: measure the *lag* of the moving average filter. Lag represents the time it takes the filter to respond to the data.

6. Perform each of the following in `octave`. (a) Assume our task is to isolate an AM radio station with a carrier frequency of 740 kHz. Create a low-pass, windowed-sinc filter designed to filter noise above 745 kHz. The number of samples M in the filter kernel is your choice. (b) Using *spectral inversion*, create a high-pass windowed-sinc filter designed to filter noise below 735 kHz. (c) Combine these filters to create a band-pass filter, and plot the frequency response. (d) Mix a 740 kHz carrier with a 2.5 kHz audio signal *plus noise*, and plot the magnitude of the DFT. (e) Use your band-pass filter on the data, and plot the filtered spectrum.

7. **FFT convolution.** Perform the following with FFT convolution. (a) Show that the convolution of two square pulses is a triangle wave. (b) Show that the convolution of one period of a sawtooth wave with itself is a "quadratic wave." **Bonus:** Generate code to play these signals as audio to hear the differences.

8. (a) Create a step pulse in `octave`, and run it through a recursive LP filter. (b) Plot the *phase* of the output versus frequency. (c) Are the results *linear* or non-linear?

9. (a) Reverse the order of the step pulse samples in the previous exercise. (b) Run the reversed step through the recursive LP filter, and plot the phase response. (c) Run the step through the recursive LP filter, reverse the output, and run it through again, to show the phase response is linear (or zero). *For a clue, see Fig. 19-8 in the course textbook.*

10. **Bonus: chirping signals**. Suppose we have a signal with a frequency that depends on time:

$$f(t) = f_0 - \beta t \tag{1}$$

The start frequency is f_0 in MHz, and the "chirp rate" is β, with units of MHz/μs. The chirp signal is

$$s(t) = A\cos\left(2\pi f(t)t\right) \tag{2}$$

If we delay the signal by a time $t_{\rm d}$, this corresponds to $s(t - t_{\rm d})$. (a) Show that mixing $s(t)$ and $s(t - t_{\rm d})$ results in two frequency components, and the lower one is $\beta t_{\rm d}$. (b) Using `octave`, create a chirping signal with $\beta = 2$ MHz/μs, $f_0 = 5$ MHz, and $t_{\rm d} = 0.5$ μs. Add plenty of noise to the signal. (c) Using the filter of your choice, isolate the low-frequency component. Show that your code produces the correct value for the "downconverted" low-frequency, $\beta t_{\rm d}$. DSP for chirping signals can arise in the analysis of radar echos for fast-moving objects.

1. (a) Derive the Fourier series of a square wave, listing which coefficients are non-zero. (b) Write an `octave` script that creates a time series corresponding to the Fourier series, with a fundamental frequency of 1.4 MHz and a maximum time $T_{\max}$ corresponding to about 10 periods. (c) Compute the DFT of the time series. (d) Using the `octave` routines `figure()` and `subplot()`, plot the time series in one graph, and in the same figure, plot the magnitude of the DFT in another graph.

a). $f(t) = \sum_{n=-\infty}^{\infty} C_n \underbrace{e^{j2\pi n f_0 t}}_{\cos(2\pi n f_0 t) + \sin(2\pi n f_0 t)}$

$$= a_0 + \sum_{n=1}^{\infty} [a_n \cos(2\pi n f_0 t) + b_n \sin(2\pi n f_0 t)]$$

①. $a_0 = \frac{1}{T} \int_{\frac{T}{2}}^{\frac{T}{2}} f(t)\, dt$ where $f(t) = \begin{cases} 1 & -\frac{T}{2} < t < 0 \\ -1 & 0 < t < \frac{T}{2} \end{cases}$, odd

since $f(t)$ odd function

$a_0 = 0$

②. $a_n = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} \underbrace{f(t)}_{\text{odd}} \cdot \underbrace{\cos(2\pi n f_0 t)}_{\text{even}}\, dt$

odd · even = odd function

Thus $a_n = 0$

③ $b_n = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} \underbrace{f(t)}_{\text{odd}} \underbrace{\sin(2\pi n f_0 t)}_{\text{odd}}\, dt$

odd · odd = even function

Thus b_n is the nonzero part.

$$b_n = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\sin(2\pi n f_0 t)\,dt$$

$$\vdots$$

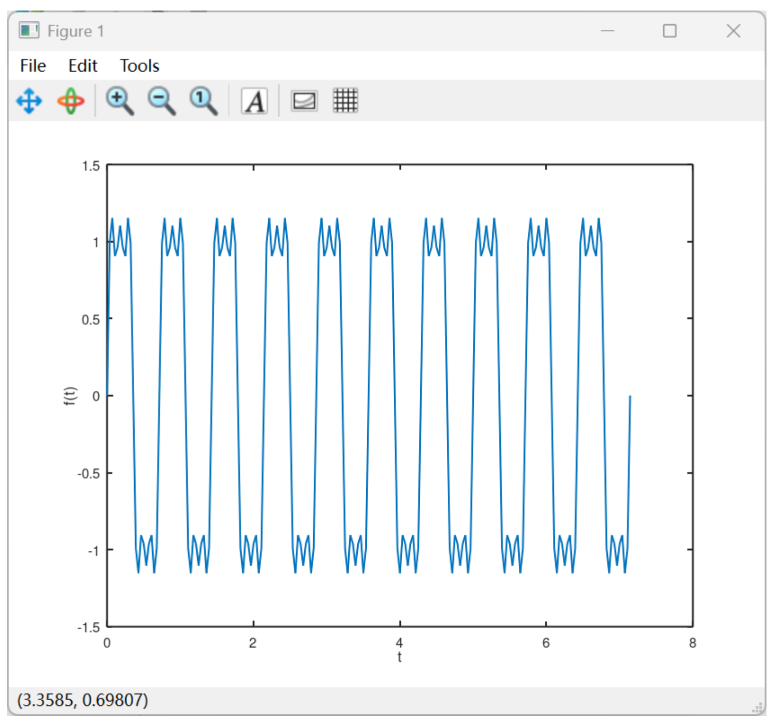
$$= \frac{4}{T}\cdot\frac{1}{2\pi n\cdot\frac{1}{T}}\left((-1)^n-1\right) = \frac{4}{2\pi n}\left[(-1)^n-1\right] = \frac{2}{\pi n}\left[(-1)^n-1\right]$$

$$b_n = \begin{cases} \frac{4}{n\pi} & n \text{ is odd} \\ 0 & n \text{ is even} \end{cases}$$

$$\text{Thus} \quad f(t) = \sum_{n=1,3,5,\ldots}^{\infty} \frac{4}{n\pi}\sin(2\pi n f_0 t)$$

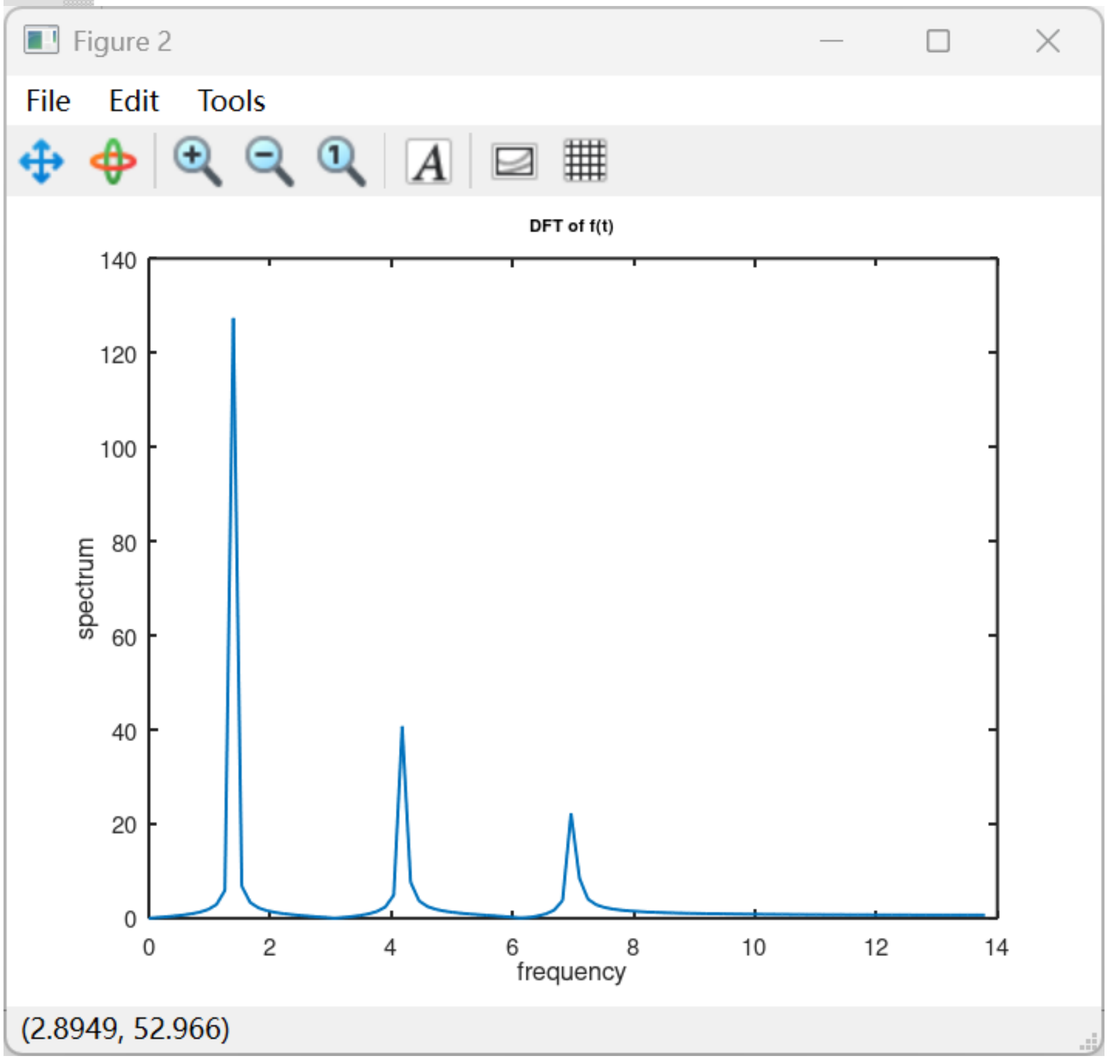
b).

```
%1.b====================================
f0 = 1.4e6;
T = 1 / f0;
Tmax = 10 * T;
N = 5;%odd N
fs = 20 * f0;
dt = 1 / fs;
t = 0 : dt : Tmax;


f = zeros(size(t));

%nonzero components
for n = 1:2:N
    bn = 4/(n * pi);
    f =f+bn*sin(2*pi*n*f0*t);
end


figure 1;
plot(t * 1e6, f);
xlabel('t');
ylabel('f(t)');

```



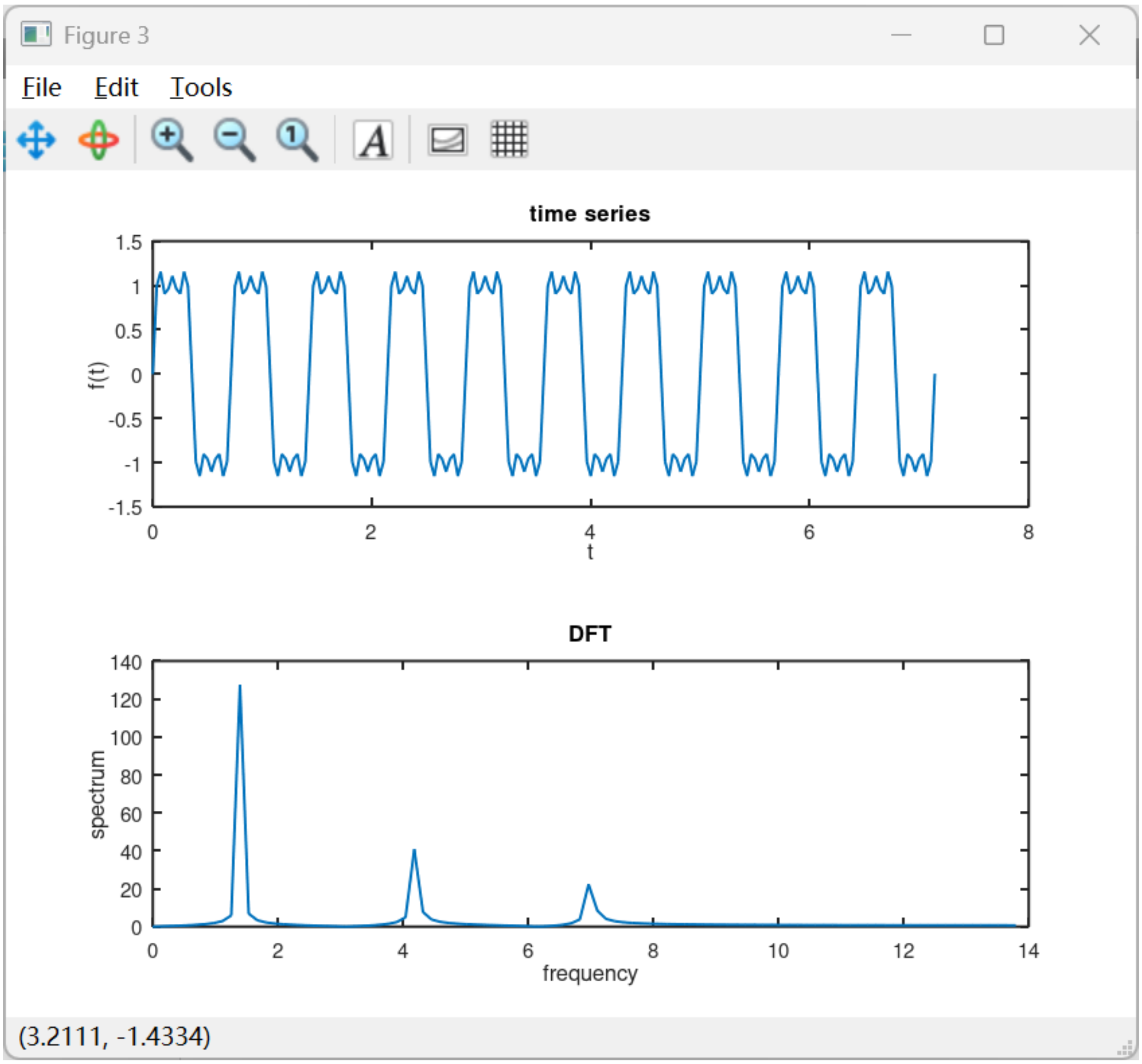
c).

```
% 1.c===========================================================
F = fft(f);
Np = length(f);% n sampling points
freq = (0:Np-1) * fs / Np;


figure 2;
plot(freq(1:Np/2) / 1e6, abs(F(1:Np/2)));%show half since symmtry
xlabel('frequency');
ylabel('spectrum');
title('DFT of f(t)');

```

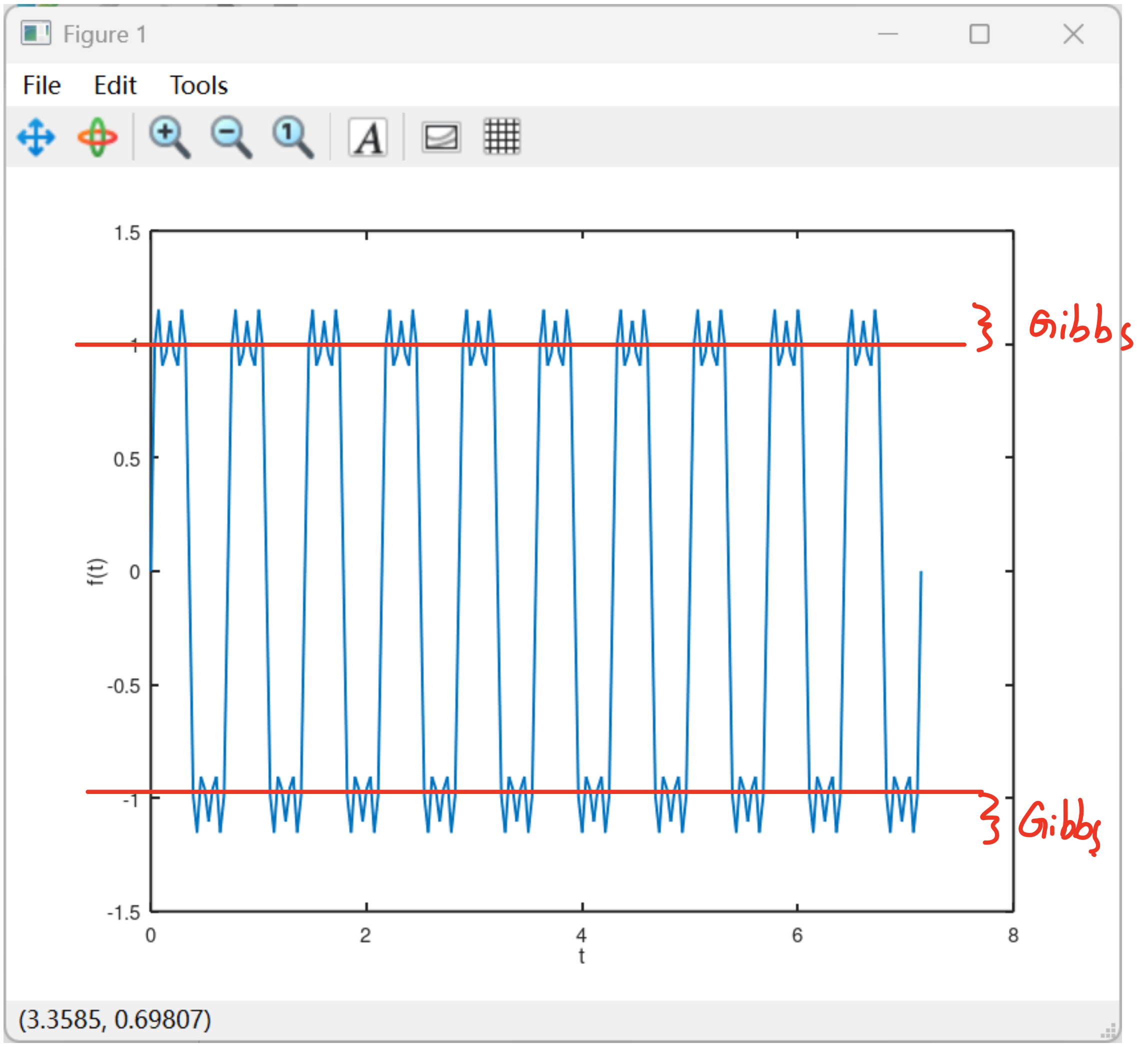


d).

```
% 1.d========================================================
figure 3;
subplot(2,1,1); %time series
plot(t * 1e6, f);
xlabel('t');
ylabel('f(t)');
title('time series');


subplot(2,1,2); %magnitude
plot(freq(1:Np/2) / 1e6, abs(F(1:Np/2)));
xlabel('frequency');
ylabel('spectrum');
title('DFT');
```



2. Do you observe the **Gibbs effect** in your Fourier series in the previous exercise? Why or why not?

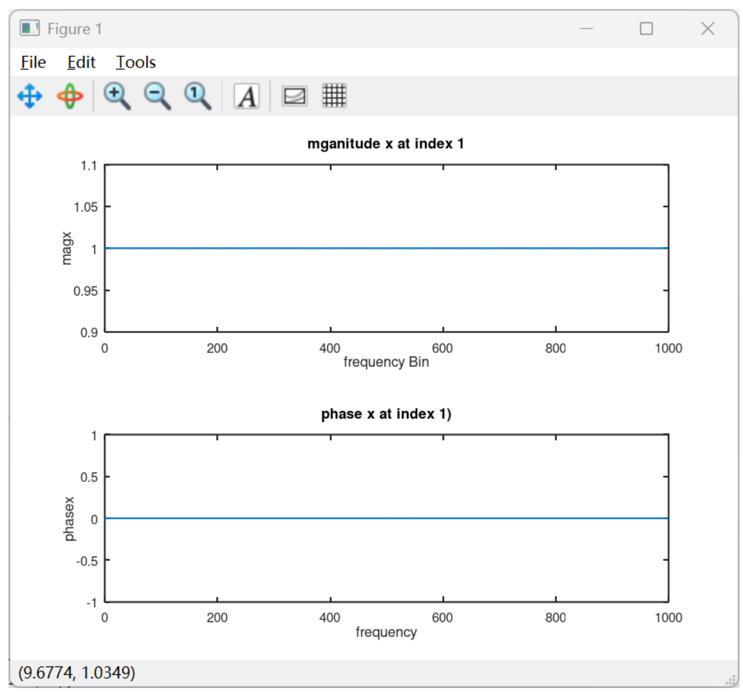


Yes. since $f(t) = \sum_{n=1,3,5\ldots}^{N} \frac{4}{n\pi} \sin(2\pi n f_0 t)$, The graph is really a combination of sinwaves, its impossible To get a perfect square wave, But we can use LPF or Blackman window to reduce Gibbs effect.

3. (a) Define an $N = 1000$ sample $\delta[n]$ signal in an `octave` script, with the non-zero value in the first sample. (b) Compute the magnitude and phase of the DFT, and graph them versus frequency. (c) Advance the non-zero sample in the $\delta[n]$ by 100 samples, and recompute the graphs. (d) What is happening to the phase? Use the `unwrap()` function in `octave` to graph the linear relationship between phase and frequency. (e) Use the slope of the phase versus frequency to measure the group delay of the $\delta[n]$. Do you obtain the right result? (f) **Bonus:** what happens to the group delay measurement if the $\delta[n]$ signal has noise?

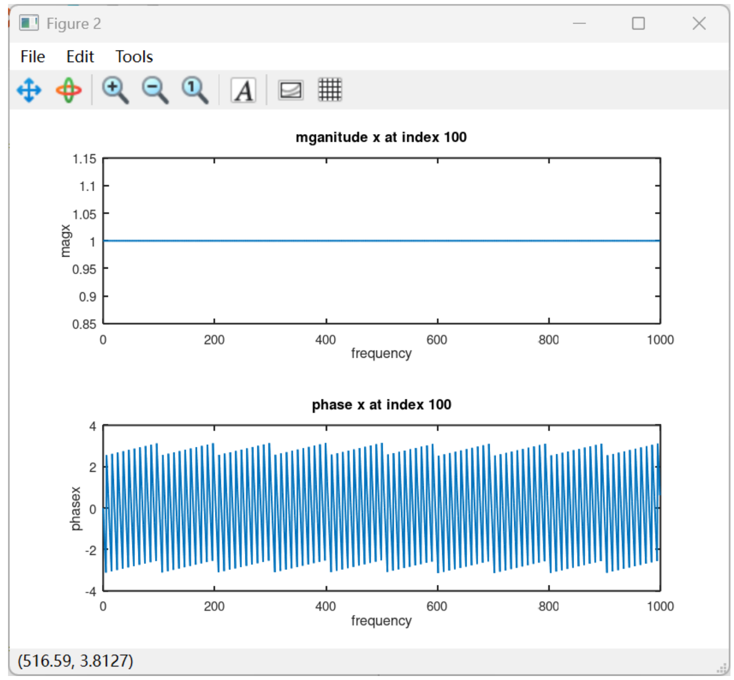
(a), (b)

```
% 3.a ===================================
N = 1000;
x = zeros(1, N);
x(1) = 1;

% 3.b ===================================
X = fft(x);
f = 0:N-1;

magX = abs(X);% magnitude
phaseX = angle(X);% phase

figure(1);
subplot(2,1,1);
plot(f, magX);
title('mganitude x at index 1');
xlabel('frequency');   // frequency of n
ylabel('magx');

subplot(2,1,2);
plot(f, phaseX);
title('phase x at index 1)');
xlabel('frequency');
ylabel('phasex');
```



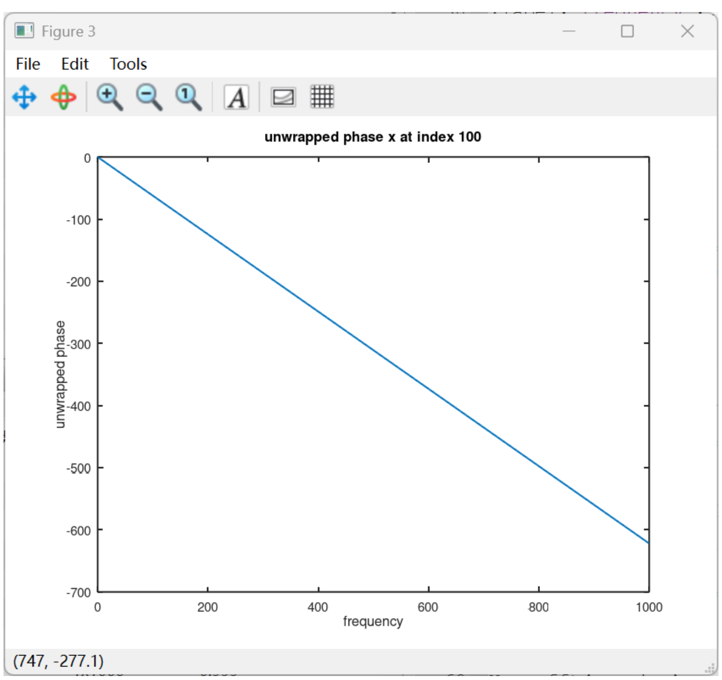
c)

```
% 3.c =================================
x2 = zeros(1, N);
x2(100) = 1; %delta at index 100

X2 = fft(x2);
magX2 = abs(X2);
phaseX2 = angle(X2);

figure(2);
subplot(2,1,1);
plot(f, magX2);
title('mganitude x at index 100');
xlabel('frequency');
ylabel('magx');


subplot(2,1,2);
plot(f, phaseX2);
title('phase x at index 100');
xlabel('frequency');
ylabel('phasex');
```



d)

```
% 3.d ========================================
unwrapped = unwrap(phaseX2);

figure(3);
plot(f, unwrapped);
title('unwrapped phase x at index 100');
xlabel('frequency');
ylabel('unwrapped phase');
```



e) Group delay $= -\dfrac{d\phi}{df}$

```
%% 3.e ========================================================
k1 = 1;%start
k2 = 100;%end
fs = 1;
freq = (0:N-1) * fs / N;

delta_phi = unwrapped(k2) - unwrapped(k1);% phase diff
delta_f = freq(k2) - freq(k1);% frequency diff

group_delay = - delta_phi / (2 * pi * delta_f); %divde 2 pi get samples
fprintf('d = %.2f samples', group_delay);
```

```
d = 99.00 samples>> |
```

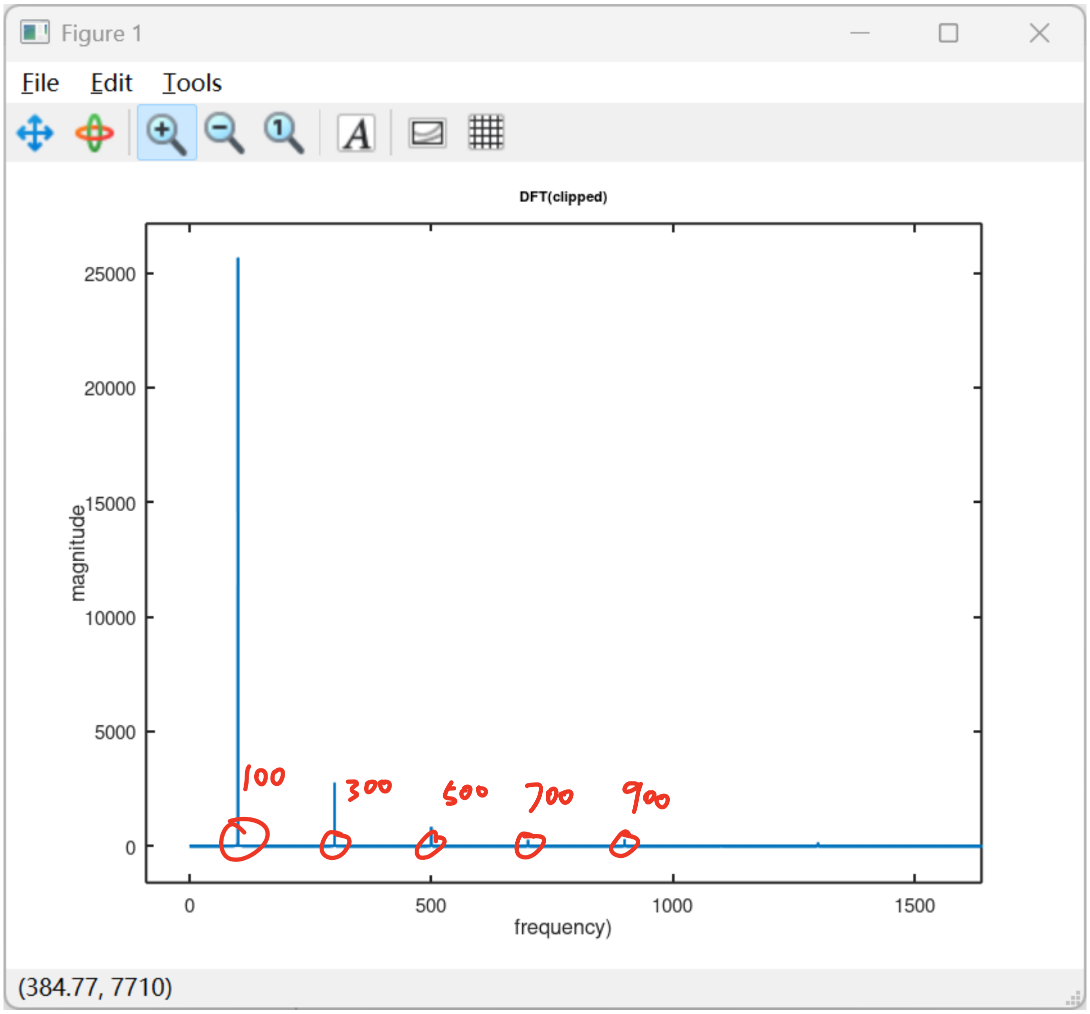
f) If noise is big, Then it would affect on phase, and unwrap(phase) will not be linear. which means $\dfrac{\Delta\phi}{\Delta f}$ will have uncertainties.

4. **Clipping** in DSP data. (a) Using `octave`, create a sine wave with the following properties: a sampling rate of 10 MHz, a frequency of 100 kHz, a T_{max} of 6 ms, and an amplitude of 1.0. The data should have more than 10^4 samples, so there is no need to graph it. (b) Using the `find()` function in `octave`, set all samples *greater* than 0.75 to 0.75. Set all samples *less* than -0.75 to -0.75. Here is a clue as to how this works:

```
x(find(x>=0.75)) = 0.75
```

The resulting signal is now **clipped**. Clipping often results when DSP data falls outside a nominal digital range. (c) Plot the magnitude of the DFT, and locate the frequency spike at 100 kHz. (d) Do you observe harmonics? In your own words, explain the spectrum of harmonics given that the signal is clipped.

```
% 4. a)
fs = 10e6;
f = 100e3;
Tmax = 6e-3;
A = 1.0;

t = 0:1/fs:Tmax;
x = A * sin(2*pi*f*t);

% 4.b =====================================
x(find(x >= 0.75)) = 0.75;% ceiling
x(find(x <= -0.75)) = -0.75;% floor

% 4.c =====================================

X = fft(x);
N = length(x);
f_axis = (0:N-1) * fs / N;

figure(1);
plot(f_axis(1:N/2) / 1e3, abs(X(1:N/2)));
xlabel('frequency)');
ylabel('magnitude');
title('DFT(clipped)');

```

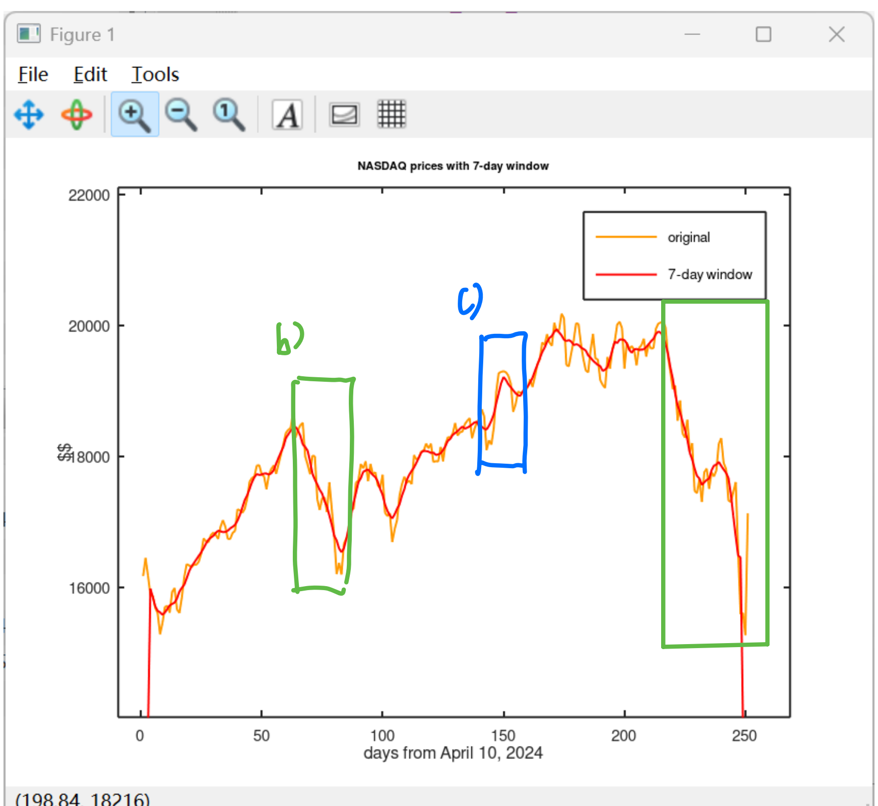


d). Yes, there is harmonics
where $f_0 = 100kHz$ is the main components (original))
and $f = 3f_0, 5f_0, 7f_0 \ldots$ become harmonics
after clipping

5. Download the "NASDAQ closing prices 2024-2025" file from the course Moodle page. The left column has units of days (starting with April 10th, 2024), and the right column is the value of the NASDAQ stock index in US dollars. (a) Plot the data, and in the same plot, apply a 1-week moving average filter to smooth the data. (b) Note where the announcement of US tariffs marks a sharp drop in value. (c) Note where the value rises sharply after the US President announced a 90 day pause in tariffs during April 2025. (d) Does this moving average capture rapid shifts in economic policy? Why or why not? (e) **Bonus**: measure the *lag* of the moving average filter. Lag represents the time it takes the filter to respond to the data.

a).

```
%5.a ==============================================
cd 'C:\Users\49902\Desktop';
data = load('nasdaq_2024_2025.dat');

days = data(:, 1);
prices = data(:, 2);

window = 7;
avg_filter = ones(1, window) / window;
smoothed = conv(prices, avg_filter, 'same');


figure(1);
plot(days, prices, 'Color', [1 0.6 0]); hold on;
plot(days, smoothed, 'r');
xlabel('days from April 10, 2024');
ylabel('$$');
title('NASDAQ prices with 7-day window');
legend('original', '7-day window');
```



b. at day 60 and at day 210 there is a sharp drop.

d). No, a 7-day moving average filter does not capture some sudden change in price, especially when there is new policies. The smoothing causes loss of informations.

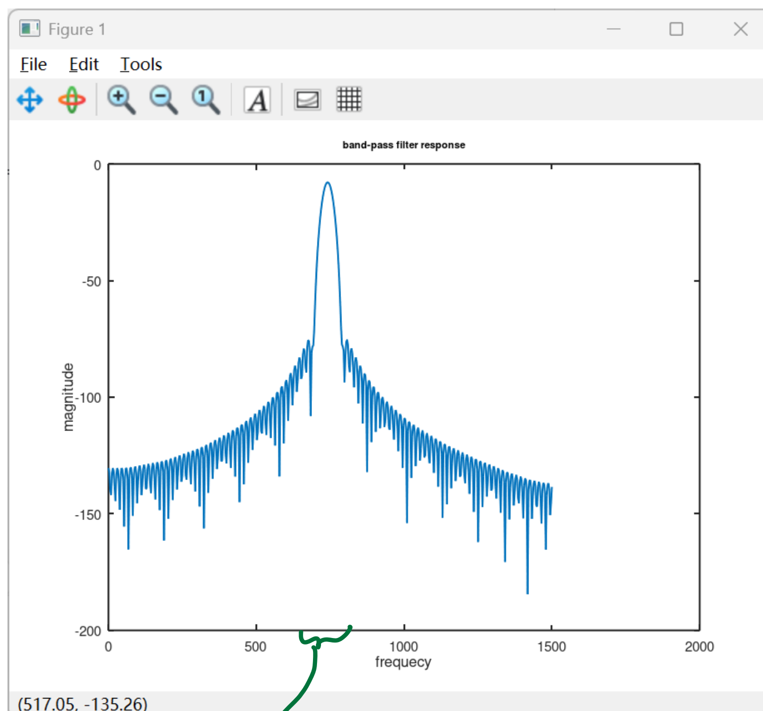
e). $$lag = \frac{N-1}{2}$$ $N = 7$ day filter.

$$= 3 \text{ days.}$$

6. Perform each of the following in `octave`. (a) Assume our task is to isolate an AM radio station with a carrier frequency of 740 kHz. Create a low-pass, windowed-sinc filter designed to filter noise above 745 kHz. The number of samples M in the filter kernel is your choice. (b) Using *spectral inversion*, create a high-pass windowed-sinc filter designed to filter noise below 735 kHz. (c) Combine these filters to create a band-pass filter, and plot the frequency response. (d) Mix a 740 kHz carrier with a 2.5 kHz audio signal *plus noise*, and plot the magnitude of the DFT. (e) Use your band-pass filter on the data, and plot the filtered spectrum.

(a,b,c)

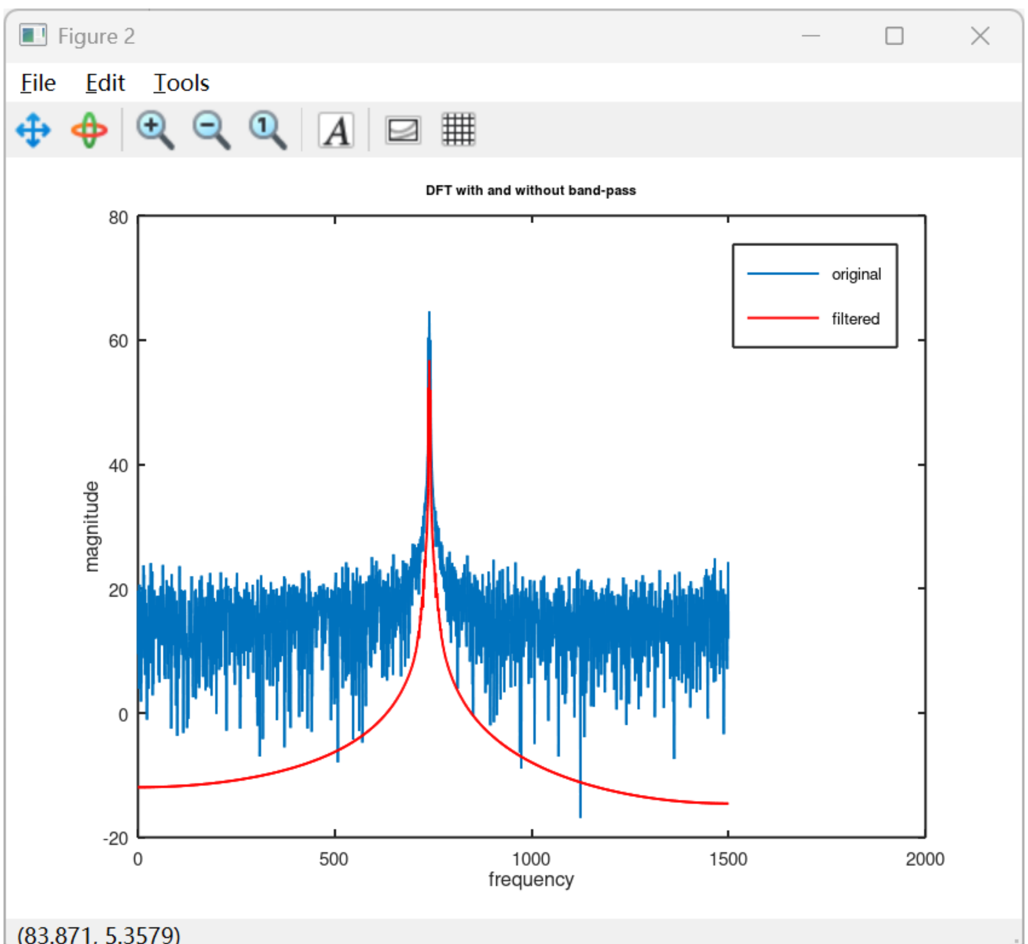
```
%6.a ==============================================
fs = 3e6;
fc = 740e3;
fcl = 745e3 / fs; %low - pass
fch = 735e3 / fs; %high - pass
M = 201;
n = 0:M-1;
alpha = (M-1)/2;

%low-pass filter cutoff = 745kHz
h_lp = sin(2*pi*fcl*(n-alpha))./(pi*(n-alpha));
h_lp(alpha+1) = 2*fcl;
h_lp = transpose(h_lp).*blackman(M);


%6.b ==============================================
%high-pass filter cutoff = 735kHz
h_hp = sin(2*pi*fch*(n-alpha))./(pi*(n-alpha));
h_hp(alpha+1) = 2*fch;
h_hp = transpose(h_hp).*blackman(M);
h_hp = -h_hp;
h_hp(alpha+1) = 1 + h_hp(alpha+1);

%6.c ==============================================
%band-pass filter = lp*hp
h_bp = conv(h_lp, h_hp);

figure(1);
H = abs(fft(h_bp, 1024));
f = linspace(0, fs/2, 512);
plot(f/1e3, 20*log10(H(1:512)));
xlabel('frequecy');
ylabel('magnitude');
title('band-pass filter response');
```



735 kHz ≤ B_p ≤ 745 kHz

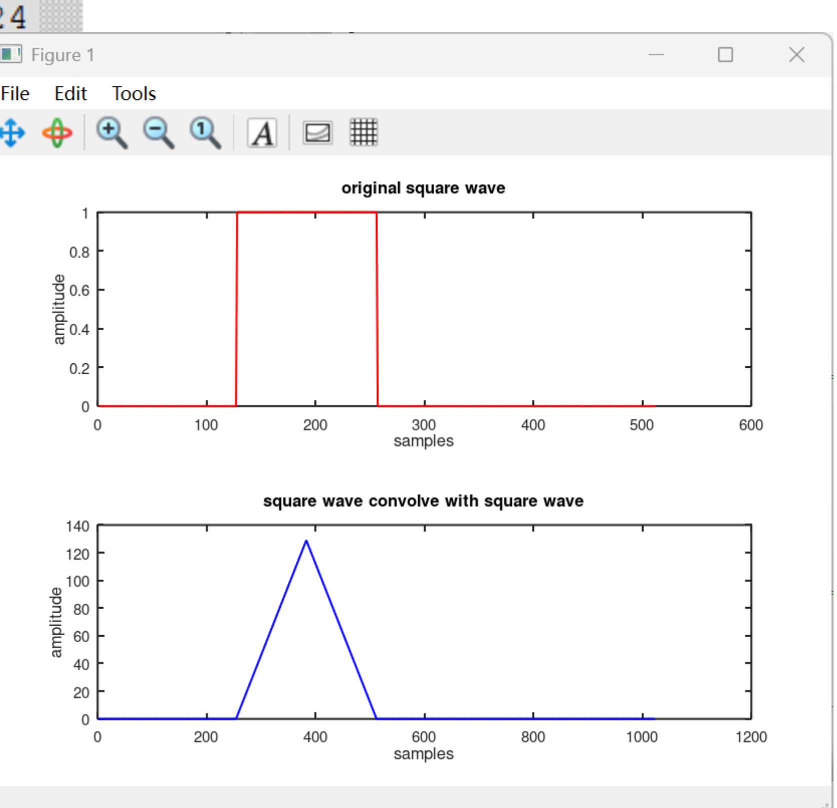
(d),(e)

```
%6.d ====================================================
tmax = 0.01;
t = 0:1/fs:tmax;
f_audio = 2.5e3;

audio = cos(2*pi*f_audio*t);
carrier = cos(2*pi*fc*t);
am = (1 + audio) .* carrier; %am with carrier

noise = 0.1 * randn(size(t));
x = am + noise;% original signal

N = 4096;
X = fft(x, N);
X_mag = 20*log10(abs(X(1:N/2)));
f = (0:N/2-1) * fs / N;

%6.e ====================================================
y = conv(x, h_bp,'same');%same letght
Y = fft(y, N);
Y_mag = 20*log10(abs(Y(1:N/2)));

figure(2);
plot(f/1e3, X_mag,'DisplayName','original'); hold on;
plot(f/1e3, Y_mag, 'r','DisplayName','filtered');
xlabel('frequency');
ylabel('magnitude');
title('DFT with and without band-pass');
legend;

```

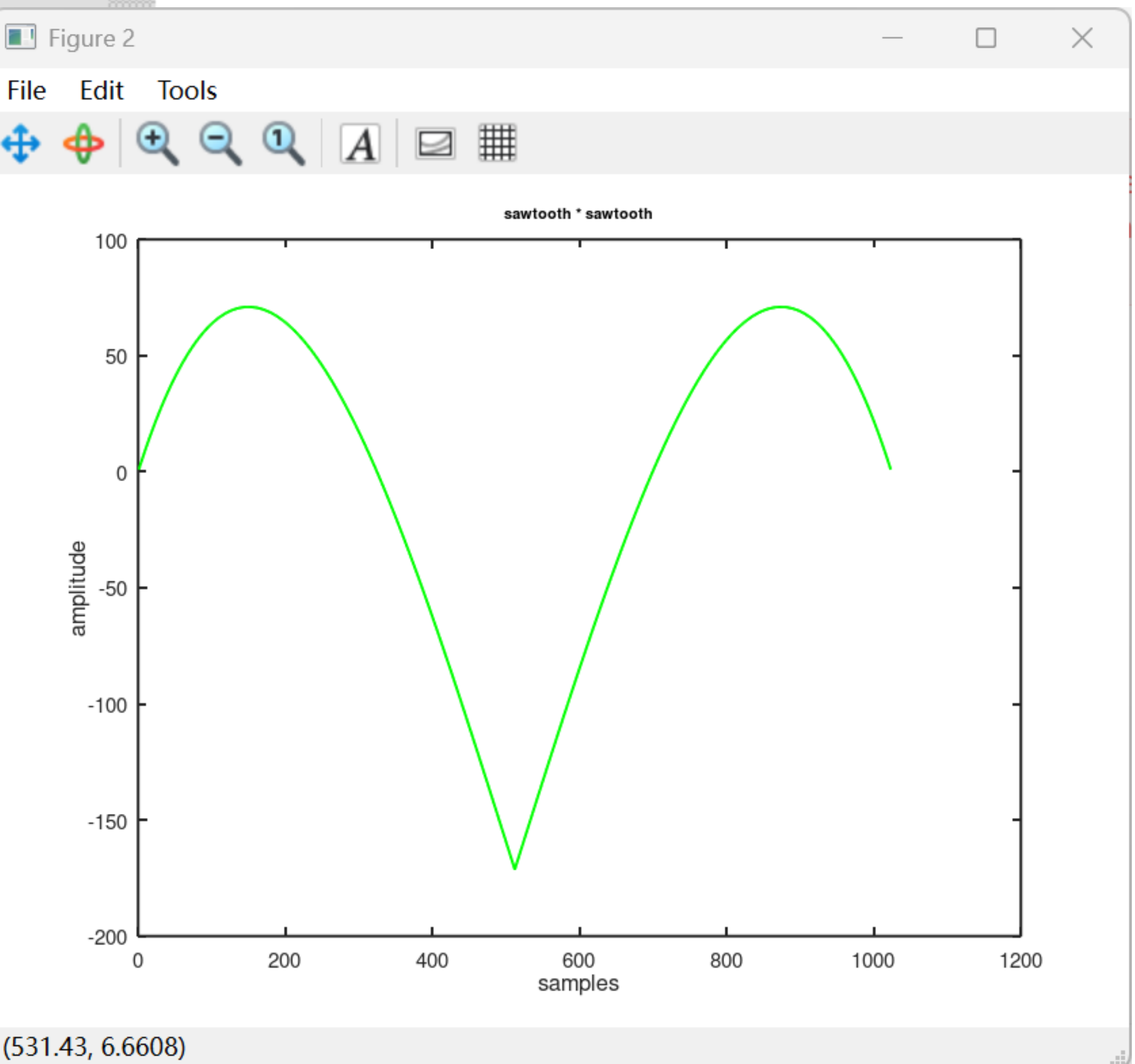


7. **FFT convolution.** Perform the following with FFT convolution. (a) Show that the convolution of two square pulses is a triangle wave. (b) Show that the convolution of one period of a sawtooth wave with itself is a "quadratic wave." **Bonus:** Generate code to play these signals as audio to hear the differences.

```
%7.a ==================================================
% square wave
fs = 8000;
N = 512;
square = zeros(1, N);
square(N/4:N/2) = 1;

%square wave * square wave
A = fft(square, 2*N);
B = fft(square, 2*N);
triangle = conv(square, square);

figure(1);
subplot(2,1,1);
plot(square, 'r');
title('original square wave');
xlabel('samples'); ylabel('amplitude');

subplot(2,1,2);
plot(triangle, 'b');
title('square wave convolve with square wave');
xlabel('samples'); ylabel('amplitude');
```



```
%7.b ==========================================
saw = linspace(-1, 1, N);
A = fft(saw, 2*N);
quad = conv(saw, saw);

figure(2);
plot(quad, 'g');
title('saw wave * saw wave');
xlabel('samples'); ylabel('amplitude');

```

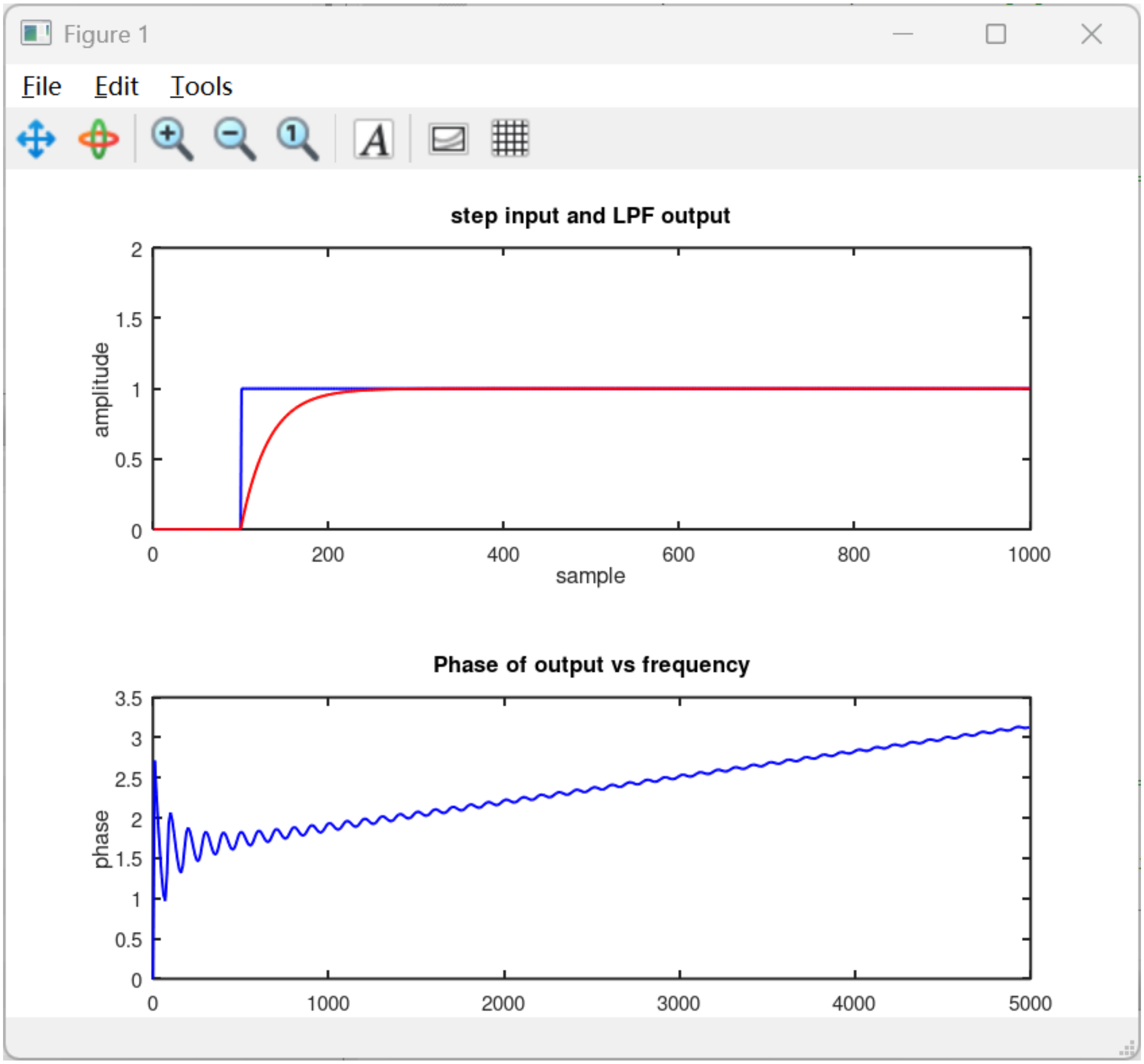


C).

```
%Bonus ====================================
signal = triangle / max(abs(triangle));
player = audioplayer(signal, fs);
play(player);
```

8. (a) Create a step pulse in `octave`, and run it through a recursive LP filter. (b) Plot the *phase* of the output versus frequency. (c) Are the results *linear* or non-linear?

```
%8.a ===========================================
fs = 1e4;
fc = 50;
x = exp(-2*pi*fc/fs);
a0 = 1 - x;
b1 = x;
N = 1000;
M = 100;

st = [zeros(M,1); ones(N-M,1)]; % step pulse
y = zeros(size(st));
y(1) = a0 * st(1);
for i = 2:N
  y(i) = a0 * st(i) + b1 * y(i-1);
endfor

% 8.b ===================================
Y = fft(y);
phi = angle(Y);
f = (0:N-1) * fs / N;

figure(1);
subplot(2,1,1);
plot(st,'color','b');
hold on;
plot(y,'color','r');
axis([0 N+1 0 2]);
xlabel('sample');
ylabel('amplitude');
title('step input and LPF output');


subplot(2,1,2);
plot(f(1:N/2), phi(1:N/2),'b');
xlabel('frequency');
ylabel('phase');
title('Phase of output vs frequency');
```



C). Recursive filter: is linear

$$y[n] = \sum a_i \cdot [n-i] + \sum b_i \cdot y[n-i]$$

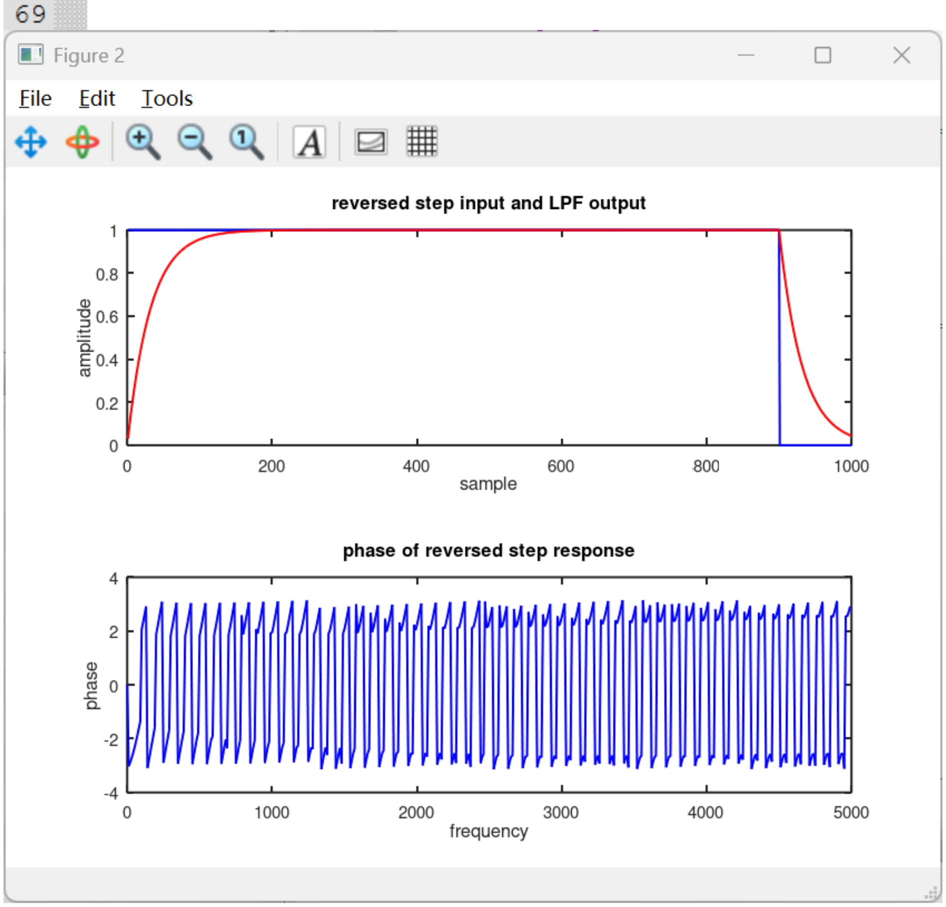
If a_i b_i constant, Then

$y[n]$ is linear

The phase from (b) shows a consistent and smooth line formed with sample impact of FFT and delay.

9. (a) Reverse the order of the step pulse samples in the previous exercise. (b) Run the reversed step through the recursive LP filter, and plot the phase response. (c) Run the step through the recursive LP filter, reverse the output, and run it through again, to show the phase response is linear (or zero). *For a clue, see Fig. 19-8 in the course textbook.*

```
%9.a ==========================================================
st_rev = flipud(st); % reverse step signal
y_rev = zeros(size(st_rev));
y_rev(1) = a0 * st_rev(1);
for i = 2:N
  y_rev(i) = a0 * st_rev(i) + b1 * y_rev(i-1);
endfor


%9.b ==========================================================
Y_rev = fft(y_rev);
phi_rev = angle(Y_rev);

figure(2);
subplot(2,1,1);
plot(st_rev, 'b'); hold on;
plot(y_rev, 'r');
xlabel('sample'); ylabel('amplitude');
title('reversed step input and LPF output');

subplot(2,1,2);
plot(f(1:N/2), phi_rev(1:N/2), 'b');
xlabel('frequency'); ylabel('phase');
title('phase of reversed step response');

```

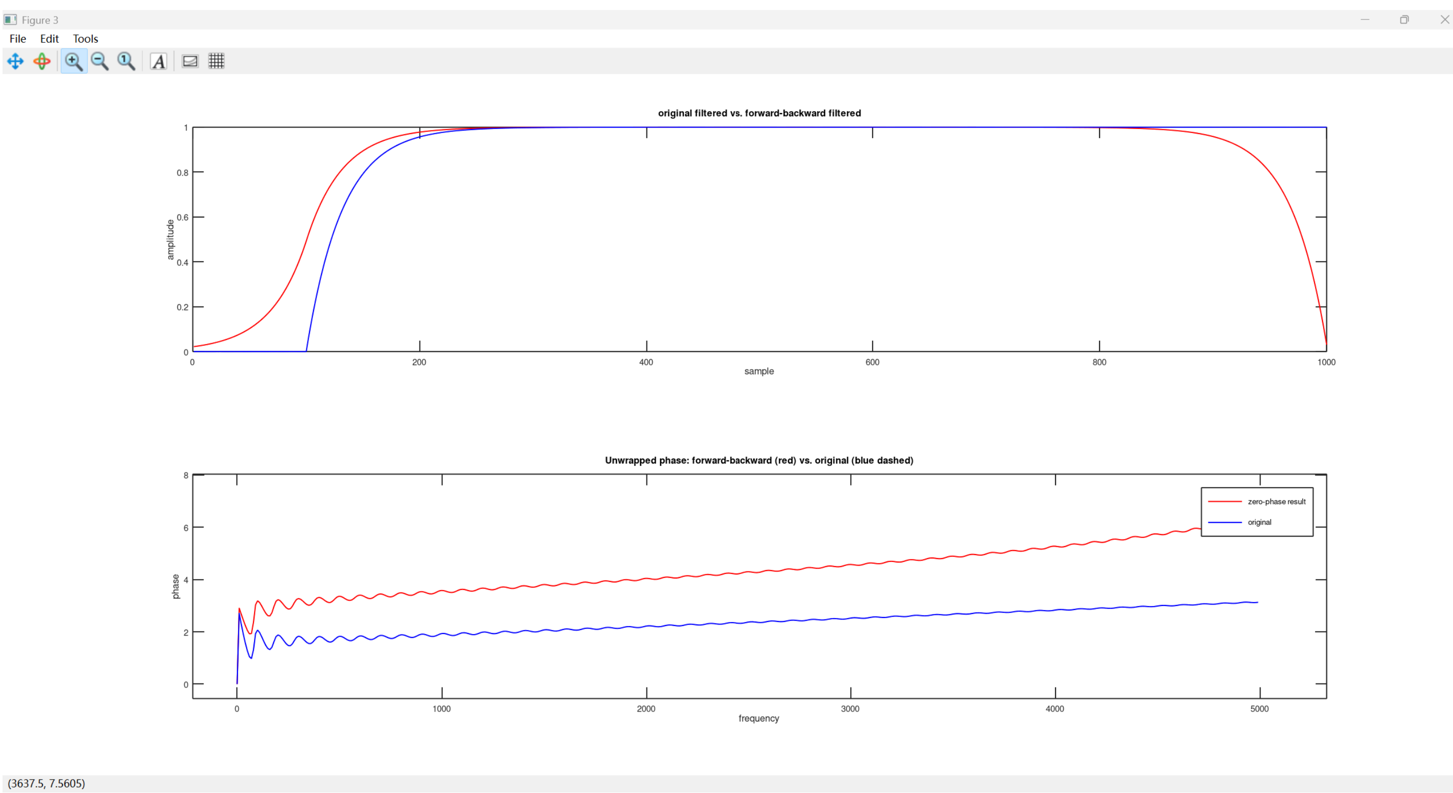


0-900 High

phase non-linear since frequency reversed.

C)

```
% 9.c ========================================================
y_fwd = zeros(size(st));
y_fwd(1) = a0 * st(1);
for i = 2:N
  y_fwd(i) = a0 * st(i) + b1 * y_fwd(i-1);
endfor

y_bwd = flipud(y_fwd);
y_temp = zeros(size(y_bwd));
y_temp(1) = a0 * y_bwd(1);
for i = 2:N
  y_temp(i) = a0 * y_bwd(i) + b1 * y_temp(i-1);
endfor

y_final = flipud(y_temp);

Y_final = fft(y_final);
phi_final = unwrap(angle(Y_final));
phi_orig = unwrap(angle(Y));

figure(3);
subplot(2,1,1);
plot(y_final, 'r'); hold on;
plot(y, 'b');
xlabel('sample'); ylabel('amplitude');
title('original filtered vs. forward-backward filtered');

subplot(2,1,2);
plot(f(1:N/2), phi_final(1:N/2), 'r'); hold on;
plot(f(1:N/2), phi_orig(1:N/2), 'b');
xlabel('frequency'); ylabel('phase');
title('Unwrapped phase: forward-backward (red) vs. original (blue dashed)');
legend('zero-phase result', 'original');
```

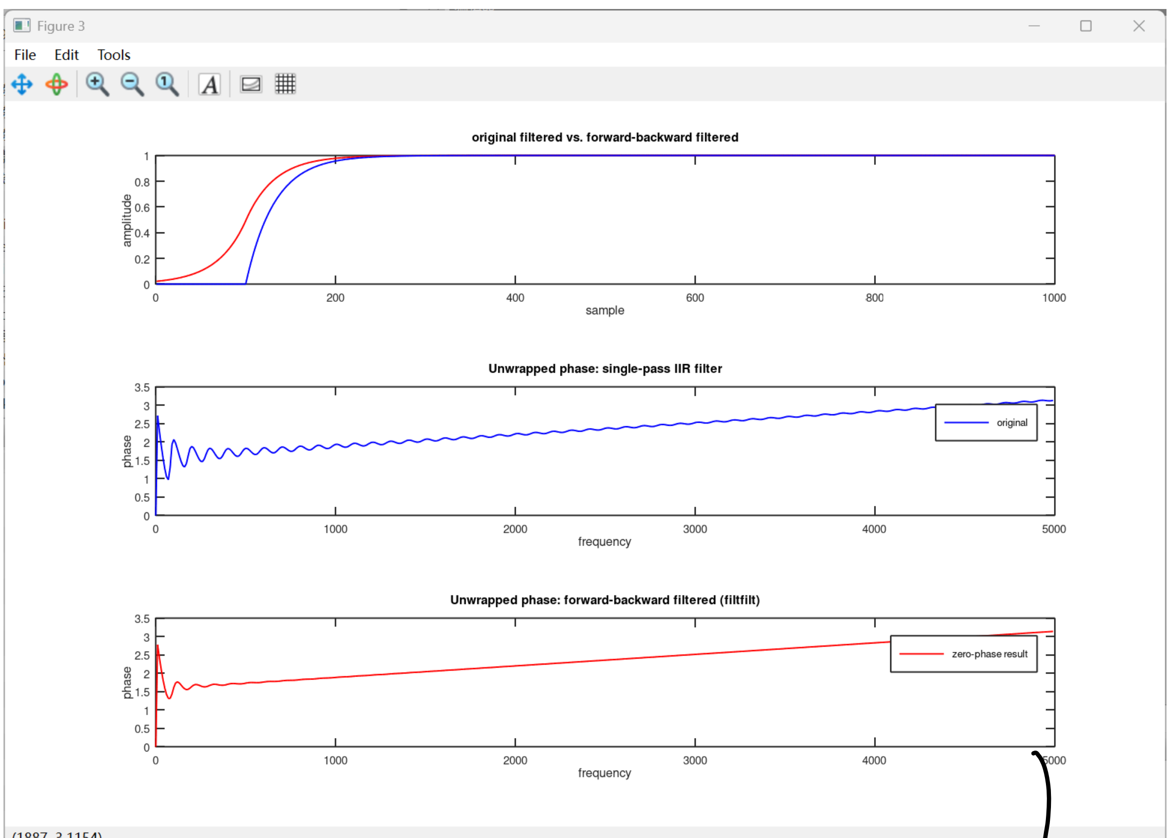


Even less linear than original, Don't know why.

phase should be canceled

· apply filtfilt (Built-in zero phase)

```
% 9.c filtfilt ========================================================

b = a0;
a = [1, -b1];

y_final = filtfilt(b, a, st);%

Y_final = fft(y_final);
phi_final = unwrap(angle(Y_final));

y_fwd = filter(b, a, st);
phi_orig = unwrap(angle(fft(y_fwd)));

figure(3);
subplot(3,1,1);
plot(y_final, 'r'); hold on;
plot(y_fwd, 'b');
xlabel('sample'); ylabel('amplitude');
title('original filtered vs. forward-backward filtered');

subplot(3,1,2);
plot(f(1:N/2), phi_orig(1:N/2), 'b');
xlabel('frequency'); ylabel('phase');
title('Unwrapped phase: single-pass IIR filter');
legend('original');

subplot(3,1,3);
plot(f(1:N/2), phi_final(1:N/2), 'r');
xlabel('frequency'); ylabel('phase');
title('Unwrapped phase: forward-backward filtered (filtfilt)');
legend('zero-phase result');
```



now linear, and no phase distortion.

10. **Bonus: chirping signals**. Suppose we have a signal with a frequency that depends on time:

$$f(t) = f_0 - \beta t \tag{1}$$

The start frequency is f_0 in MHz, and the "chirp rate" is β, with units of MHz/μs. The chirp signal is

$$s(t) = A\cos\left(2\pi f(t)t\right) \tag{2}$$

If we delay the signal by a time t_d, this corresponds to $s(t-t_\mathrm{d})$. (a) Show that mixing $s(t)$ and $s(t-t_\mathrm{d})$ results in two frequency components, and the lower one is βt_d. (b) Using `octave`, create a chirping signal with $\beta = 2$ MHz/μs, $f_0 = 5$ MHz, and $t_\mathrm{d} = 0.5$ μs. Add plenty of noise to the signal. (c) Using the filter of your choice, isolate the low-frequency component. Show that your code produces the correct value for the "downconverted" low-frequency, βt_d. DSP for chirping signals can arise in the analysis of radar echos for fast-moving objects.

a) $s(t) = A\cos(2\pi f(t)t)$

$= A\cos\left(2\pi\left(f_0 t - \frac{1}{2}\beta t^2\right)\right)$

$s(t-t_d) = A\cos\left(2\pi\left(f_0(t-t_d) - \frac{1}{2}\beta(t-t_d)^2\right)\right)$

$s(t)\cdot s(t-t_d)$