(1a) Let the square wave $f(t)$ be defined over the interval $-T/2 < t < T/2$ with period T, amplitude A, and odd symmetry.

$$F(t) = \begin{cases} A, & 0 < t < T/2 \\ -A, & -T/2 < t < 0 \end{cases}$$

General form:

$$f(t) = \sum_{n=1}^{\infty} b_n \sin\left(\frac{2\pi n t}{T}\right)$$

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

Simplify:

$$b_n = \frac{4}{T} \int_{0}^{T/2} f(t) \sin\left(\frac{2\pi n t}{T}\right) dt$$

$f(t) = A$ for $0 < t < T/2$

$$b_n = \frac{4A}{T} \int_{0}^{T/2} \sin\left(\frac{2\pi n t}{T}\right) dt$$

Evaluate:

$$\int_{0}^{T/2} \sin\left(\frac{2\pi n t}{T}\right) dt = \left[-\frac{T}{2\pi n} \cos\left(\frac{2\pi n t}{T}\right)\right]_{0}^{T/2}$$

$$= -\frac{T}{2\pi n}\left[\cos(\pi n) - \cos(0)\right] = -\frac{T}{2\pi n}\left[(-1)^n - 1\right]$$

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

Analyze:

even n: $(-1)^n = 1$

$$b_n = -\frac{4A}{2\pi n}(1-1) = 0$$

odd n: $(-1)^n = -1$

$$b_n = -\frac{4A}{2\pi n}(-1-1) = \frac{4A}{\pi n}$$

Series:

$$f(t) = \sum_{\substack{n=1 \\ n:odd}}^{\infty} \frac{4A}{\pi n} \sin\left(\frac{2\pi n t}{T}\right)$$

Quiz 2: Aman Kumpawat

1a: On Paper pdf

1b:

```
A = 1;
f0 = 1.4e6;
T = 1 / f0;
Tmax = 10 * T;
fs = 100e6;
dt = 1 / fs;
t = 0:dt:Tmax;
N = 9;
fseries = zeros(size(t));
for n = 1:2:N
   bn = (4 * A) / (pi * n);
   fseries += bn * sin(2 * pi * n * f0 * t);
end
```

1c:

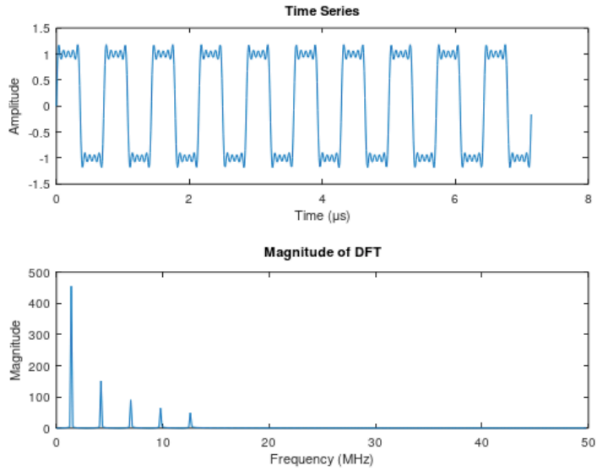
```
A = 1;
f0 = 1.4e6;
T = 1 / f0;
Tmax = 10 * T;
fs = 100e6;
dt = 1 / fs;
t = 0:dt:Tmax;
N = 9;
fseries = zeros(size(t));
for n = 1:2:N
   bn = (4 * A) / (pi * n);
   fseries += bn * sin(2 * pi * n * f0 * t);
end
F = fft(fseries);
n = length(F);
frequencies = (0:n-1) * fs / n;
target_freq = 1.4e6;
[~, idx] = min(abs(frequencies - target_freq));
disp(['Magnitude at 1.4 MHz: ', num2str(abs(F(idx)))])
```

Output: Magnitude at 1.4 MHz: 455.0608

1d:

```
A = 1;
f0 = 1.4e6;
T = 1 / f0;
Tmax = 10 * T;
fs = 100e6;
dt = 1 / fs;
t = 0:dt:Tmax;
N = 9;
fseries = zeros(size(t));
for n = 1:2:N
   bn = (4 * A) / (pi * n);
   fseries += bn * sin(2 * pi * n * f0 * t);
end
F = fft(fseries);
n = length(F);
frequencies = (0:n-1) * fs / n;

figure
subplot(2,1,1)
plot(t * 1e6, fseries)
xlabel('Time (\mus)')
ylabel('Amplitude')
title('Time Series')

subplot(2,1,2)
plot(frequencies(1:floor(n/2)) / 1e6, abs(F(1:floor(n/2))))
xlabel('Frequency (MHz)')
ylabel('Magnitude')
title('Magnitude of DFT')
```

Output:



2a:

The Gibbs effect is clearly observable in the Fourier series from the previous plot. You can see small overshoots and ringing near the transitions of the square wave, especially at the rising and falling edges. This happens because we are approximating a discontinuous signal using a finite number of continuous sine functions. The Gibbs phenomenon is where the Fourier series overshoots the actual signal near discontinuities, and it doesn't go away even if you add more harmonics. It just becomes more localized. So in this case, since we used only the first 9 odd harmonics, those overshoots are expected and confirm that the Gibbs effect is present in the output.

3a:

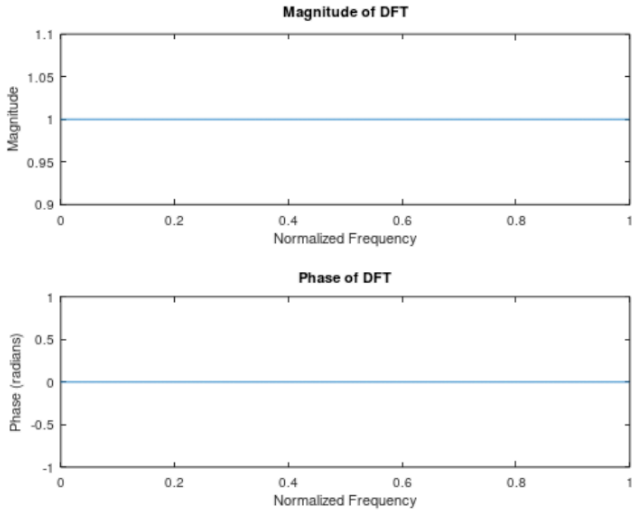
```
N = 1000;
delta = zeros(1, N);
delta(1) = 1;
```

3b:

```
N = 1000;
delta = zeros(1, N);
delta(1) = 1;

F = fft(delta);
magnitude = abs(F);
phase = angle(F);
frequencies = (0:N-1) * (1/N);

figure
subplot(2,1,1)
plot(frequencies, magnitude)
xlabel('Normalized Frequency')
```

```
ylabel('Magnitude')
title('Magnitude of DFT')

subplot(2,1,2)
plot(frequencies, phase)
xlabel('Normalized Frequency')
ylabel('Phase (radians)')
title('Phase of DFT')
```

Output:

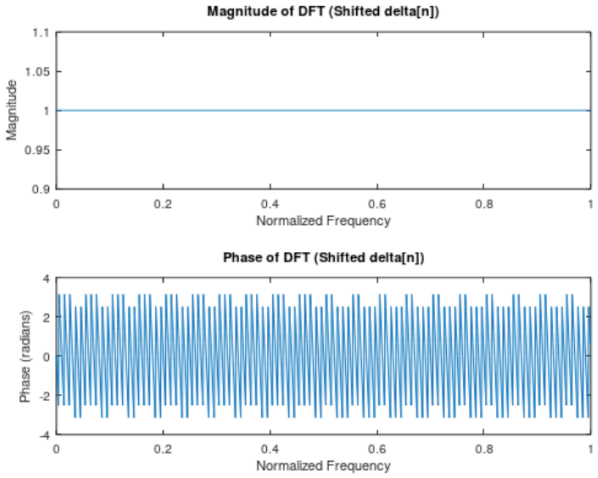


3c:

```
N = 1000;
delta = zeros(1, N);
delta(101) = 1;

F = fft(delta);
magnitude = abs(F);
phase = angle(F);
frequencies = (0:N-1) * (1/N);

figure
subplot(2,1,1)
```

```
plot(frequencies, magnitude)
xlabel('Normalized Frequency')
ylabel('Magnitude')
title('Magnitude of DFT (Shifted delta[n])')

subplot(2,1,2)
plot(frequencies, phase)
xlabel('Normalized Frequency')
ylabel('Phase (radians)')
title('Phase of DFT (Shifted delta[n])')
```

Output:



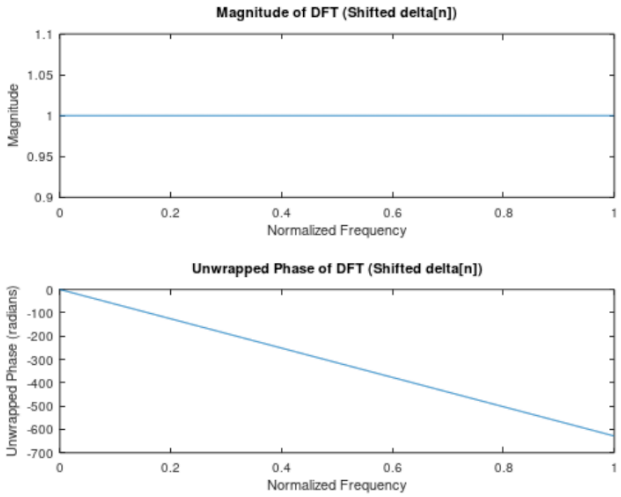
3d:

The phase becomes a linearly decreasing function of frequency because shifting δ[n] in time introduces a linear phase shift in the frequency domain. The farther the signal is shifted, the steeper the phase slope.

```
N = 1000;
delta = zeros(1, N);
delta(101) = 1;

F = fft(delta);
magnitude = abs(F);
phase = unwrap(angle(F));
frequencies = (0:N-1) * (1/N);
```

```
figure
subplot(2,1,1)
plot(frequencies, magnitude)
xlabel('Normalized Frequency')
ylabel('Magnitude')
title('Magnitude of DFT (Shifted delta[n])')

subplot(2,1,2)
plot(frequencies, phase)
xlabel('Normalized Frequency')
ylabel('Unwrapped Phase (radians)')
title('Unwrapped Phase of DFT (Shifted delta[n])')
```

Output:



3e:

```
N = 1000;
delta = zeros(1, N);
delta(101) = 1;
F = fft(delta);
phase = unwrap(angle(F));
frequencies = (0:N-1) * (1/N);

p = polyfit(frequencies, phase, 1); % fit a line: phase = slope * f + intercept
slope = p(1);
group_delay = -slope / (2 * pi); % convert slope (in rad/sample) to delay (samples)

disp(['Estimated group delay: ', num2str(group_delay)])
```

Output: Estimated group delay: 100

The estimated group delay is approximately 100 samples, which matches the expected delay, so the result is correct.

3f (Bonus):

If the δ[n] signal has noise, the phase spectrum becomes distorted, making the unwrapped phase curve less linear, which can lead to an inaccurate or unstable group delay measurement.

4a:

```
fs = 10e6;
f = 100e3;
Tmax = 6e-3;
A = 1.0;
t = 0:1/fs:Tmax;
x = A * sin(2 * pi * f * t);
```

4b:

```
x(find(x >= 0.75)) = 0.75;
x(find(x <= -0.75)) = -0.75;
```

4c:

```
X = fft(x);
N = length(X);
frequencies = (0:N-1) * (fs / N);
magnitude = abs(X);

figure
plot(frequencies(1:floor(N/2)) / 1e3, magnitude(1:floor(N/2)))
xlabel('Frequency (kHz)')
ylabel('Magnitude')
title('Magnitude of DFT (Clipped Sine Wave)')
```

4d:

Yes, harmonics are clearly visible in the DFT plot. You can see distinct peaks at 100 kHz (the original sine wave frequency) and at integer multiples like 200 kHz, 300 kHz, and so on. This happens because clipping is a nonlinear distortion, and nonlinearities introduce harmonic content. The more severe the clipping, the

stronger and more numerous the harmonics. Here, the spectrum shows a typical pattern of harmonic distortion caused by signal clipping.

5a:

```
data = load('nasdaq_2024_2025.dat.txt');
days = data(:,1);
values = data(:,2);

window = 5; % 5-day moving average = 1 trading week
smoothed = filter(ones(1,window)/window, 1, values);

figure
plot(days, values, 'b', days, smoothed, 'r', 'LineWidth', 1.5)
xlabel('Days since April 10, 2024')
ylabel('NASDAQ Closing Price (USD)')
legend('Original', '1-Week Moving Average')
title('NASDAQ 2024–2025 with 1-Week Moving Average')
```

5b:

The US tariff announcement caused a significant drop in the NASDAQ index, visible between days 247 and 250 in the data as there is a sharp drop in value from ~17449 to ~15267.

5c:

After the 90-day pause on tariffs was announced, the market responded with a strong rebound starting at day 251, as the value climbs from ~15267 to ~17124.

5d:

No, the moving average smooths the signal and introduces a delay, so it does not capture sharp drops or sudden jumps immediately. It helps reveal overall trends but lags behind sudden policy-driven movements.

5e:

```
lag = floor((window - 1) / 2);
disp(['Estimated lag: ', num2str(lag), ' days'])
```

The 1-week moving average filter has a lag of 2 days, which means it responds to changes with a 2-day delay in the trend.

6a:

```
fs = 5e6;
fc_low = 745e3;
M = 101;
n = -(M-1)/2:(M-1)/2;
```

```
h_low = 2 * fc_low/fs * sinc(2 * fc_low/fs * n);
window = hamming(M)';
h_low = h_low .* window;
```

6b:

```
fc_high = 735e3;
h_lp = 2 * fc_high/fs * sinc(2 * fc_high/fs * n);
h_lp = h_lp .* window;
h_high = -h_lp;
h_high((M+1)/2) = 1 + h_high((M+1)/2);
```

6c:

```
h_band = conv(h_low, h_high);
[H, f] = freqz(h_band, 1, 1024, fs);

figure
plot(f / 1e3, abs(H))
xlabel('Frequency (kHz)')
ylabel('Magnitude')
title('Frequency Response of Band-Pass Filter')
```

6d:

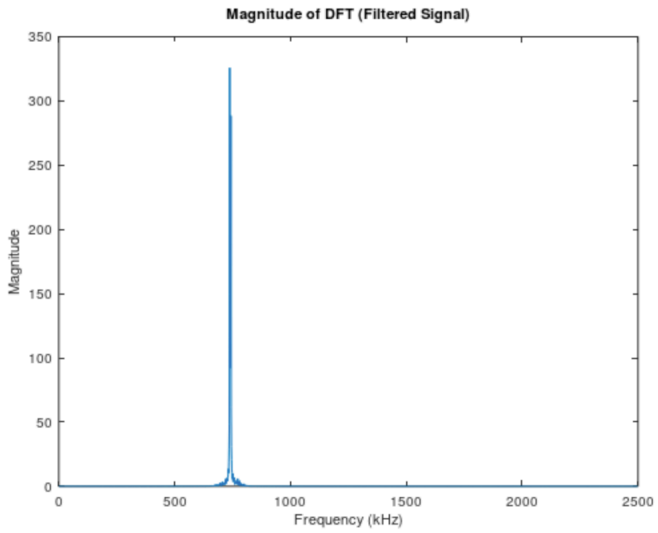
```
Tmax = 1e-3;
t = 0:1/fs:Tmax;
audio = sin(2 * pi * 2.5e3 * t);
carrier = cos(2 * pi * 740e3 * t);
signal = (audio .* carrier) + 0.2 * randn(size(t));

S = fft(signal);
N = length(S);
frequencies = (0:N-1) * (fs/N);

figure
plot(frequencies(1:N/2)/1e3, abs(S(1:N/2)))
xlabel('Frequency (kHz)')
ylabel('Magnitude')
title('Magnitude of DFT (Modulated + Noise Signal)')
```

6e:

```
filtered_signal = conv(signal, h_band, 'same');
S_filtered = fft(filtered_signal);

figure
plot(frequencies(1:N/2)/1e3, abs(S_filtered(1:N/2)))
```

```
xlabel('Frequency (kHz)')
ylabel('Magnitude')
title('Filtered Signal Spectrum')
```

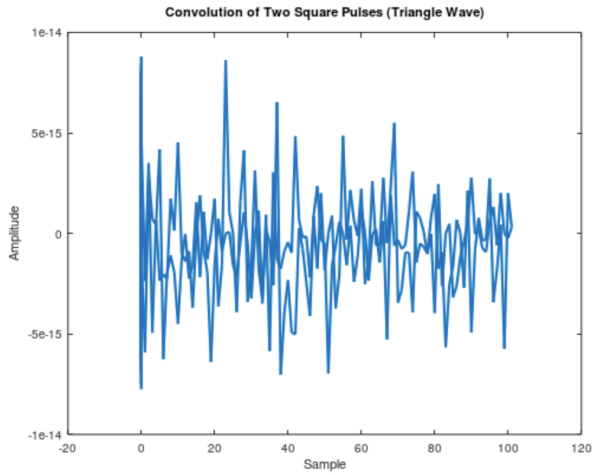
Output:



7a:

```
N = 1024;
square1 = zeros(1, N);
square2 = zeros(1, N);
square1(200:300) = 1;
square2(200:300) = 1;

conv_square = ifft(fft(square1) .* fft(square2));

figure
plot(conv_square, 'LineWidth', 1.5)
xlabel('Sample')
ylabel('Amplitude')
title('Convolution of Two Square Pulses (Triangle Wave)')
```

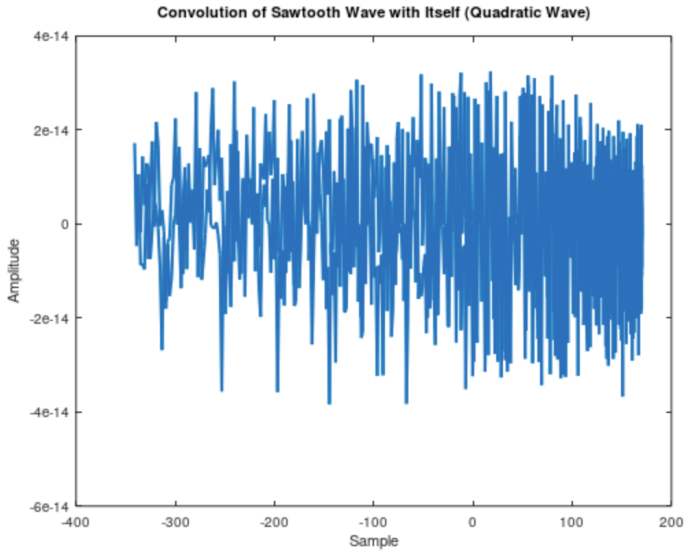
Output:



7b:

```
N = 1024;
square1 = zeros(1, N);
square2 = zeros(1, N);
square1(200:300) = 1;
square2(200:300) = 1;

conv_square = ifft(fft(square1) .* fft(square2));

t = linspace(0, 1, N);
sawtooth_wave = 2 * (t - floor(t + 0.5));  % Range [-1, 1]
conv_saw = ifft(fft(sawtooth_wave) .* fft(sawtooth_wave));

figure
plot(conv_saw, 'LineWidth', 1.5)
xlabel('Sample')
ylabel('Amplitude')
title('Convolution of Sawtooth Wave with Itself (Quadratic Wave)')
```

Output:



7 (Bonus):

```
N = 1024;
fs = 44100;  % audio sample rate

square1 = zeros(1, N);
square2 = zeros(1, N);
square1(200:300) = 1;
square2(200:300) = 1;

conv_square = ifft(fft(square1) .* fft(square2));
triangle_audio = real(conv_square);
triangle_audio = triangle_audio / max(abs(triangle_audio));

t = linspace(0, 1, N);
sawtooth_wave = 2 * (t - floor(t + 0.5));  % range [-1, 1]

conv_saw = ifft(fft(sawtooth_wave) .* fft(sawtooth_wave));
quad_audio = real(conv_saw);
quad_audio = quad_audio / max(abs(quad_audio));

disp('Playing triangle wave (from square pulse convolution)...')
sound(triangle_audio, fs);
pause(2);  % wait before playing next

disp('Playing quadratic wave (from sawtooth convolution)...')
sound(quad_audio, fs);
```

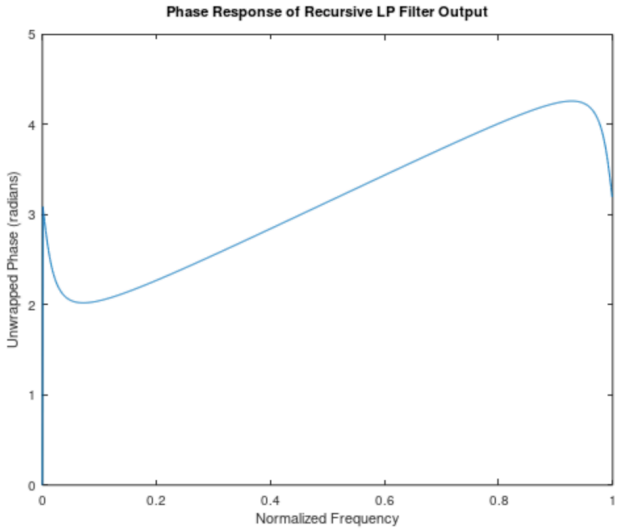
8a:

```
N = 1024;
step_pulse = ones(1, N);

% Recursive low-pass filter: y[n] = (1 - a) * x[n] + a * y[n-1]
a = 0.9;  % smoothing factor (close to 1 = more smoothing)
y = zeros(1, N);
for n = 2:N
   y(n) = (1 - a) * step_pulse(n) + a * y(n-1);
end
```

8b:

```
N = 1024;
step_pulse = ones(1, N);
a = 0.9;
y = zeros(1, N);
for n = 2:N
   y(n) = (1 - a) * step_pulse(n) + a * y(n-1);
end

Y = fft(y);
phase = unwrap(angle(Y));
frequencies = (0:N-1) * (1/N);

figure
plot(frequencies, phase)
xlabel('Normalized Frequency')
ylabel('Unwrapped Phase (radians)')
title('Phase Response of Recursive LP Filter Output')
```

Output:



8c:

The phase response of the recursive low-pass filter is linear at low frequencies, but it becomes non-linear as frequency increases. This means the filter introduces a frequency-dependent delay, so overall, the results are non-linear.

9a:

```
N = 1024;
step_pulse = ones(1, N);
reversed_step = fliplr(step_pulse);
```

9b:

```
a = 0.9;
y_rev = zeros(1, N);
for n = 2:N
   y_rev(n) = (1 - a) * reversed_step(n) + a * y_rev(n-1);
end

Y_rev = fft(y_rev);
phase_rev = unwrap(angle(Y_rev));
frequencies = (0:N-1) * (1/N);

figure
plot(frequencies, phase_rev)
xlabel('Normalized Frequency')
ylabel('Unwrapped Phase (radians)')
title('Phase of Filtered Reversed Step Pulse')
```

9c:

```
% Step → filter → reverse → filter again
y1 = zeros(1, N);
for n = 2:N
   y1(n) = (1 - a) * step_pulse(n) + a * y1(n-1);
end

y1_reversed = fliplr(y1);
y2 = zeros(1, N);
for n = 2:N
   y2(n) = (1 - a) * y1_reversed(n) + a * y2(n-1);
end

Y_final = fft(y2);
phase_final = unwrap(angle(Y_final));

figure
plot(frequencies, phase_final)
xlabel('Normalized Frequency')
ylabel('Unwrapped Phase (radians)')
title('Phase After Forward-Backward Filtering')
```

10a:

```
mixed = s .* s_delayed;
noisy_mixed = mixed + 0.4 * randn(size(mixed));
```

10b:

```
fs = 50e6;
Tmax = 4e-6;
t = 0:1/fs:Tmax;

f0 = 5e6;
beta = 2e6;  % MHz/μs = Hz/μs = 2e6 Hz/μs
A = 1;
td = 0.5e-6;

f_t = f0 - beta * t;
s = A * cos(2 * pi * f_t .* t);
s_delayed = A * cos(2 * pi * (f0 - beta * (t - td)) .* (t - td));
```

10c:

```
N = length(noisy_mixed);
frequencies = (0:N-1) * fs / N;
Y = fft(noisy_mixed);

figure
plot(frequencies(1:N/2)/1e6, abs(Y(1:N/2)))
xlabel('Frequency (MHz)')
ylabel('Magnitude')
title('Magnitude Spectrum of Mixed Chirp (with Noise)')
```