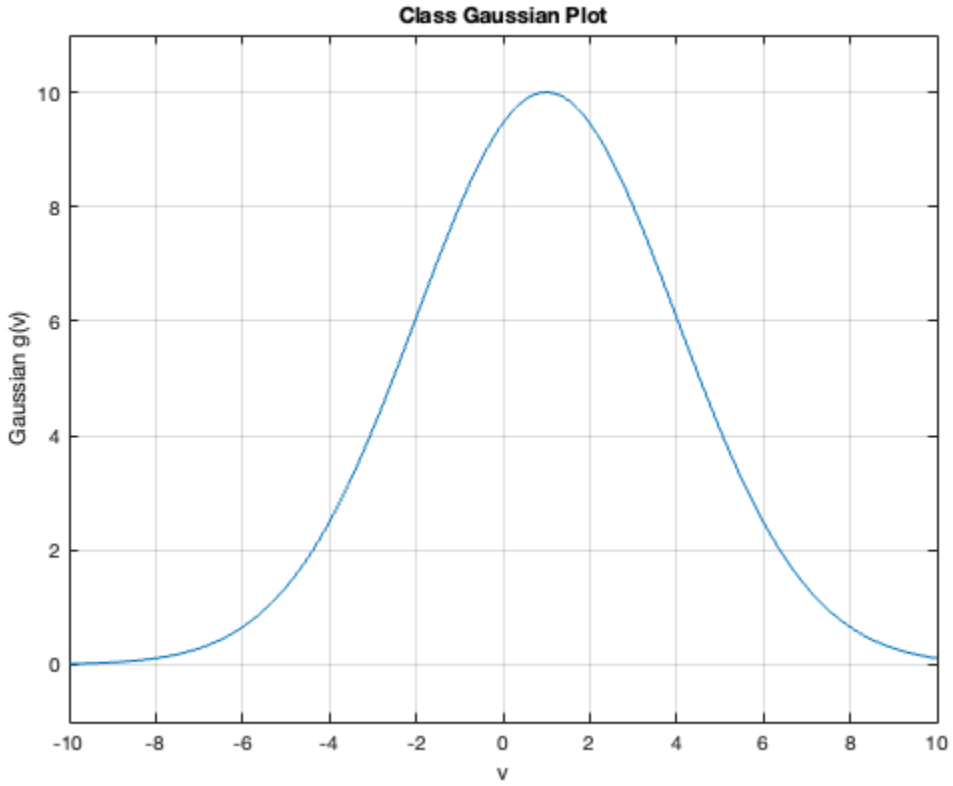
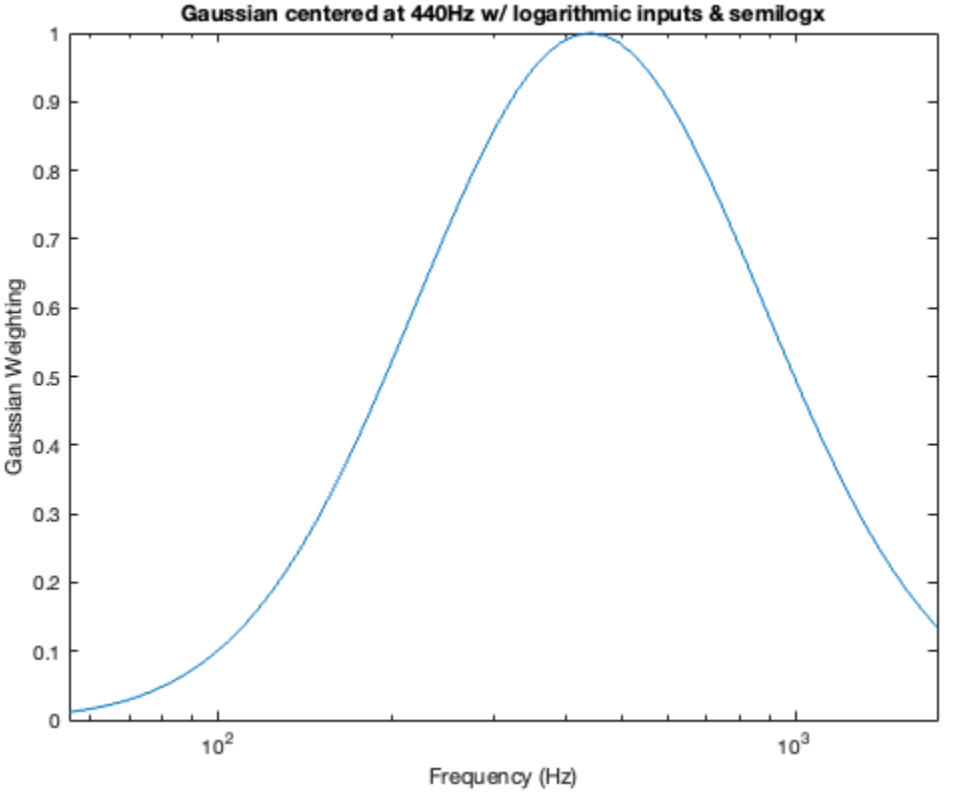
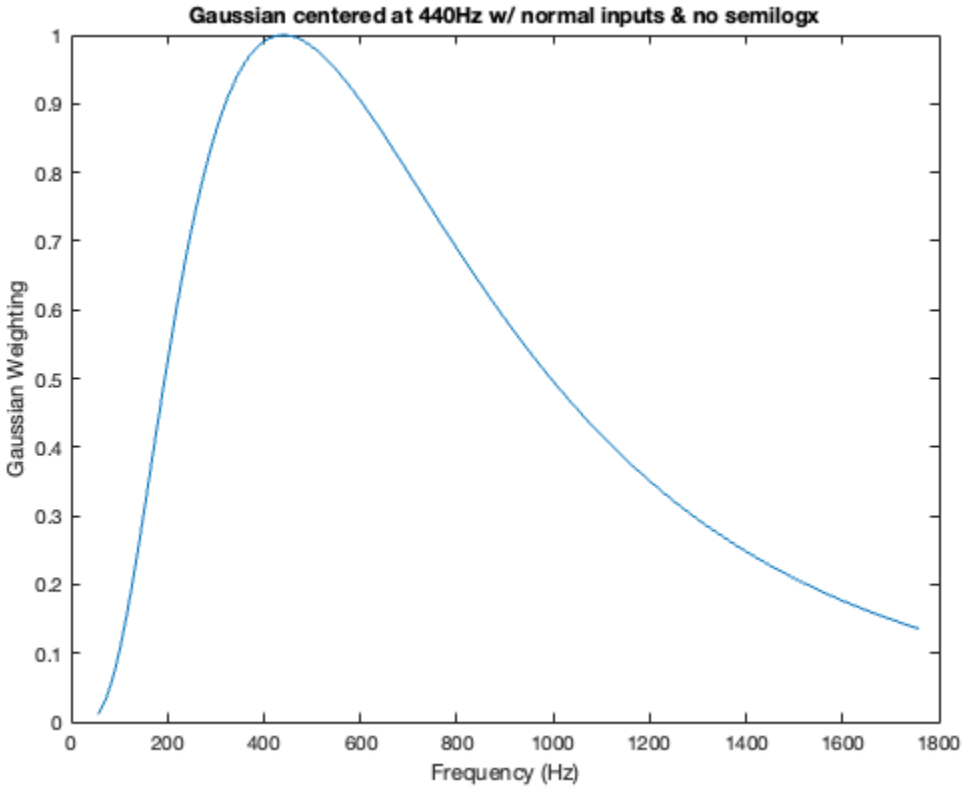
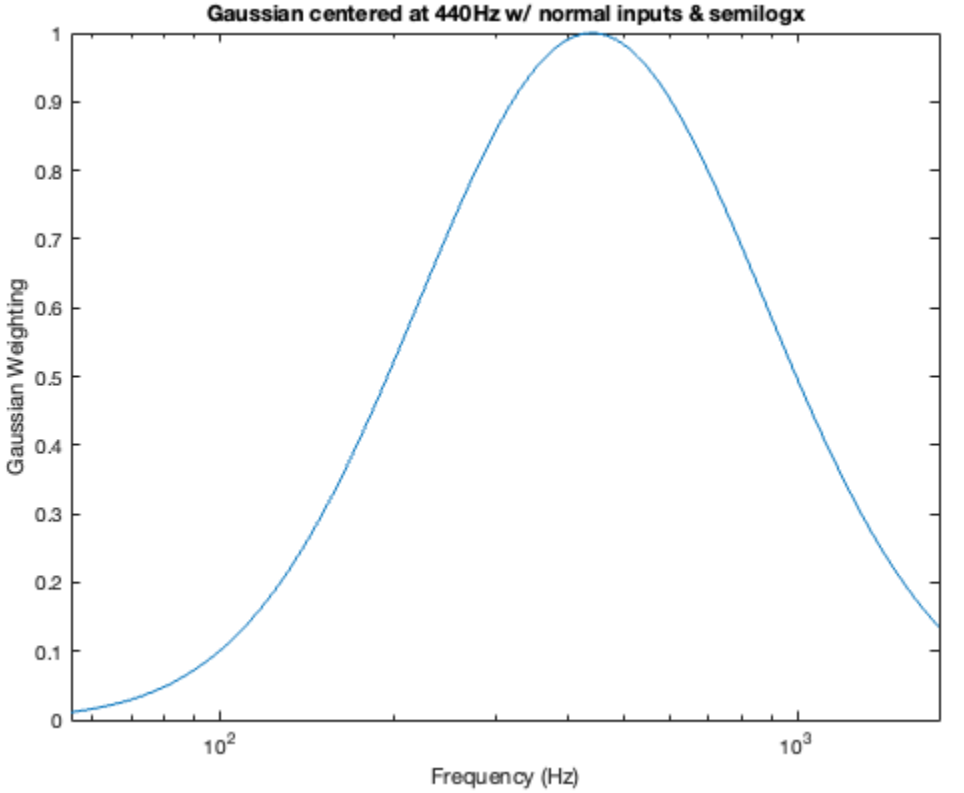
```
% Tune-Up #3

% Copy this file into a Matlab script window, add your code and answers to the
% questions as Matlab comments, hit "Publish", and upload the resulting PDF
file
% to this page for the tune-up assignment.  Please do not submit a link to a
file
% but instead upload the file itself.   Late penalty: 2 points per minute
late.

% Shepard Scale Demo
% https://www.illusionsindex.org/i/shepard-scale-illusionsLinks to an
external site.

% Part A.  We will answer the question in Section 2 of mini-project #1 on
% Shepard Scale Synthesis.
% https://users.ece.utexas.edu/~bevans/courses/signals/homework/fall2024/
miniproject1.pdfLinks to an external site.

% Write a few lines of MATLAB code to make a plot of
% 10 exp( -(v-1)^2 / (2 (3)^2) over the range -10 <= v <= 10.
% This is g(v) where alpha = 10, mu = 1, and sigma=3.
% This question is in Section 2.2 of the lab assignment
% https://dspfirst.gatech.edu/chapters/DSP1st2eLabs/ShepardScaleLab.pdfLinks
to an external site.
fs = 8000;
Ts = 1/fs;
v = -10 : Ts : 10;

alpha = 10;
mu = 1;
sigma = 3;

g = alpha * exp( -(v - mu).^2 / (2 * (sigma^2)) );
figure;
plot(v, g);
ylim( [-1, alpha+1] );
title('Class Gaussian Plot');
xlabel('v');
ylabel('Gaussian g(v)');
grid on;


% Part B. We will answer the questions in Section 3.1 of mini-project #1 on
% Shepard Scale Synthesis.
% https://users.ece.utexas.edu/~bevans/courses/signals/homework/fall2024/
miniproject1.pdfLinks to an external site.
ff_log = 2.^(log2(55): 1/12 : log2(1760));
ff_normal = 55: 2 : 1760;
fc = 440;
sig = 1; % width equal to one according to log scale (each octave is 2x prev)
```

```
% 3.1 part c
% Creating Frequency Weighting Objects
g_1 = FrequencyWeighting(fc, sig, ff_log);
g_2 = FrequencyWeighting(fc, sig, ff_normal);
g_3 = FrequencyWeighting(fc, sig, ff_normal);

figure;
semilogx(ff_log, g_1);
xlabel('Frequency (Hz)');
ylabel('Gaussian Weighting');
title('Gaussian centered at 440Hz w/ logarithmic inputs & semilogx');


% 3.1 part d, the Gaussian appears distorted because the plot command is
% on a linear basis regarding frequency. However, the Gaussian function
% utilizes a logarithmic scale and since the bell shape is compressed
% towards the center, distortion is created.

%However, when you use semilogx, the x-axis becomes logarithmic which
%aligns with how the Gaussian function is defined. This creates the desired
bell
%shape because the x-axis has proper frequency spacing by Gaussian
%distribution.

figure;
plot(ff_normal, g_2)
xlabel('Frequency (Hz)');
ylabel('Gaussian Weighting');
title('Gaussian centered at 440Hz w/ normal inputs & no semilogx');

figure;
semilogx(ff_normal, g_3);
xlabel('Frequency (Hz)');
ylabel('Gaussian Weighting');
title('Gaussian centered at 440Hz w/ normal inputs & semilogx');
```


Class Gaussian Plot
Gaussian g(v)
v
10
8
6
4
2
0
-10
-8
-6
-4
-2
0
2
4
6
8
10

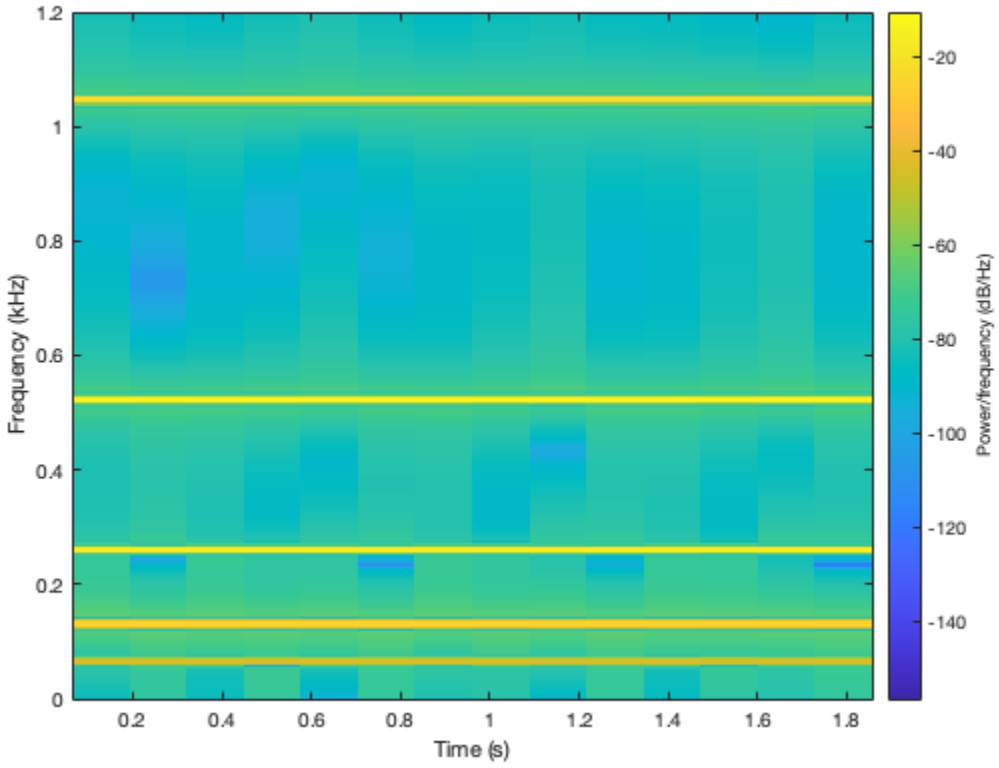

Gaussian centered at 440Hz w/ logarithmic inputs & semilogx
Gaussian Weighting
Frequency (Hz)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10^2
10^3


Gaussian centered at 440Hz w/ normal inputs & no semilogx
Gaussian Weighting
Frequency (Hz)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
200
400
600
800
1000
1200
1400
1600
1800


Gaussian centered at 440Hz w/ normal inputs & semilogx
Gaussian Weighting
Frequency (Hz)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10^2
10^3

Published with MATLAB® R2024a

```
% Q3.2
% Gaussian weighting function & parameters
sig_c = 1;
fs_c = 8000;
fcenterc = 261.63;
weighted_c = MusicalWeightingC(fcenterc, sig_c); % Gaussian function
% Spectrogram parameters
window_c = hamming(2048);
noverlap_c = 1024;
nfft_c = 2048;
figure;
spectrogram(weighted_c, window_c, noverlap_c, nfft_c, fs_c, 'yaxis'); %
Plotting spectrogram
ylim([0 1.2]);
```

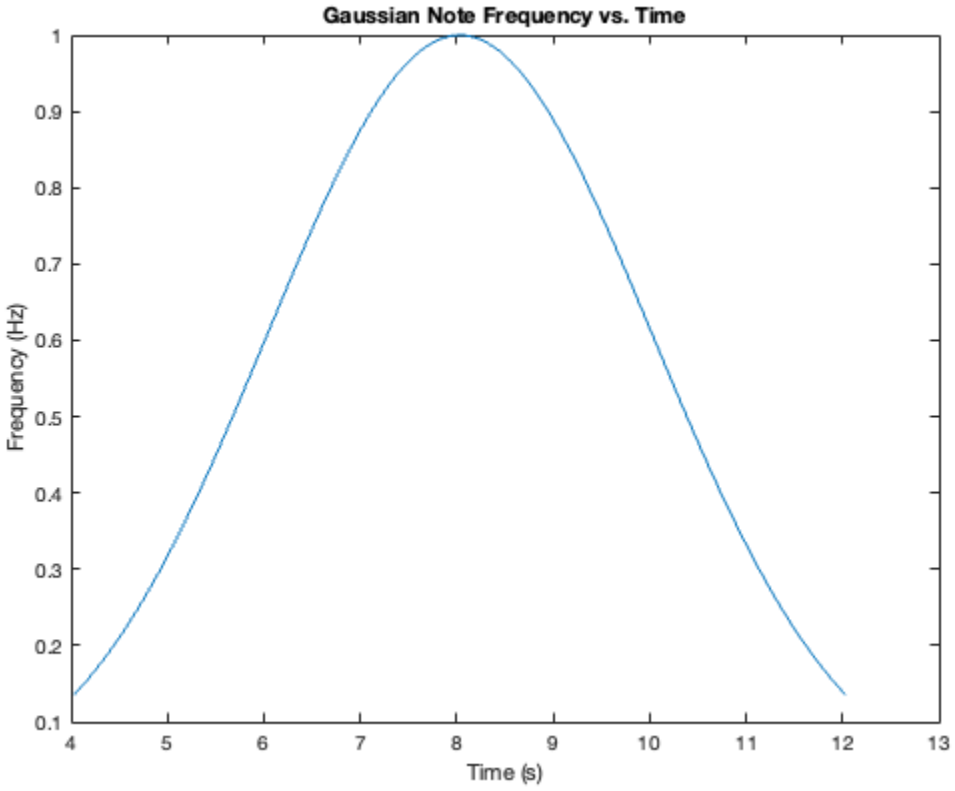


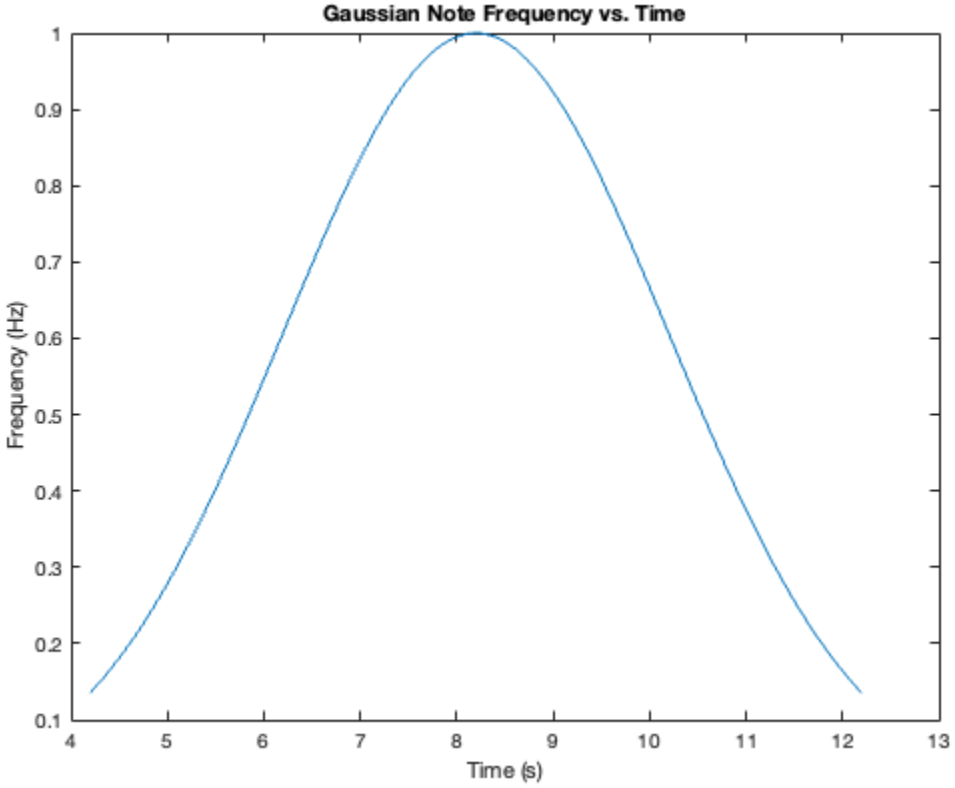
Published with MATLAB® R2024a

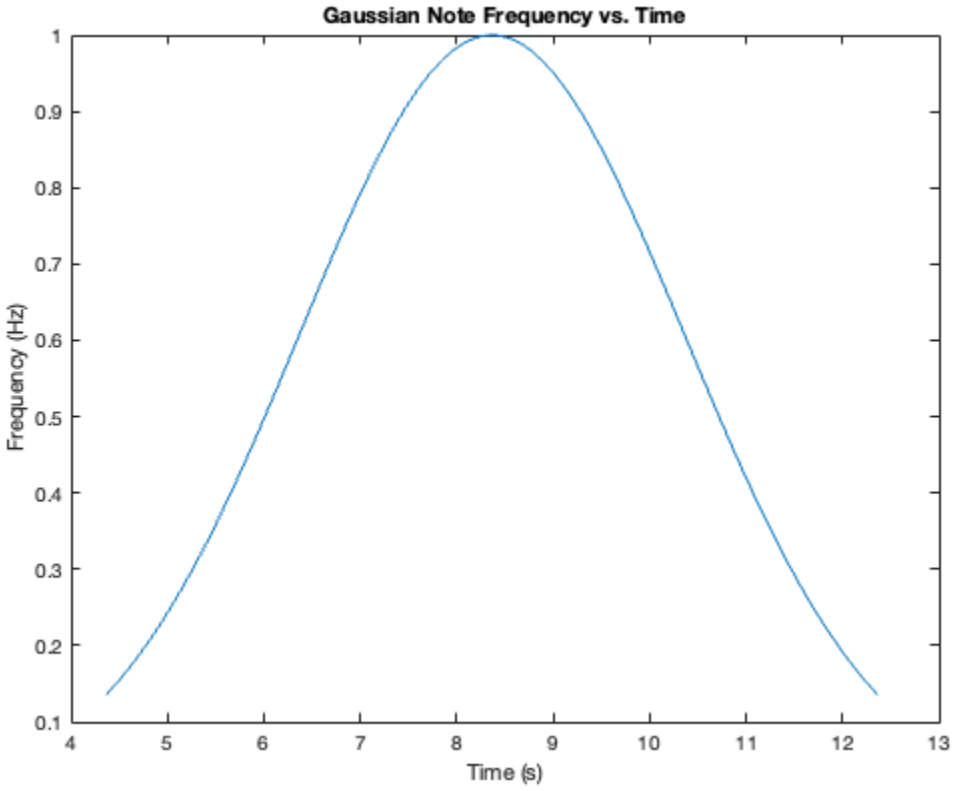
```
% Q4a - Synthesize a C major scale

% Note frequencies derived from https://www.intmath.com/trigonometric-graphs/
music.php
[C261_63, fs261_63] = NoteHarmonics(261.63);
[D293_66, fs293_66] = NoteHarmonics(293.66);
[E329_63, fs329_63] = NoteHarmonics(329.63);
[F349_23, fs349_23] = NoteHarmonics(349.23);
[G392, fs392] = NoteHarmonics(392);
[A440, fs440] = NoteHarmonics(440);
[B493_88, fs493_88] = NoteHarmonics(493.88);
[C523_25, fs523_25] = NoteHarmonics(523.25);

% Q4b - Play the C-Major scale five times
for i = 1:5
sound(C261_63, fs261_63);
pause(3);
sound(D293_66, fs293_66);
pause(3);
sound(E329_63, fs329_63);
pause(3);
sound(F349_23, fs349_23);
pause(3);
sound(G392, fs392);
pause(3);
sound(A440, fs440);
pause(3);
sound(B493_88, fs493_88);
pause(3);
sound(C523_25, fs523_25);
pause(3);
end

sigma = 2;

% Q4c - Introduce amplitude weighting using a Gaussian form

[GC261_63, logf261_63] = MusicalWeighting(261.63, sigma); % Gaussian notes so
log2
[GD293_66, logf293_66] = MusicalWeighting(293.6, sigma);
[GE329_63, logf329_63] = MusicalWeighting(329.63, sigma);
[GF349_23, logf349_23] = MusicalWeighting(349.23, sigma);
[GG392, logf392] = MusicalWeighting(392, sigma);
[GA440, logf440] = MusicalWeighting(440, sigma);
[GB493_88, logf493_88] = MusicalWeighting(493.88, sigma);
[GC523_25, logf523_25] = MusicalWeighting(523.25, sigma);

figure;
plot(logf261_63, GC261_63);
xlabel('Time (s)');
ylabel('Frequency (Hz)');
title('Gaussian Note Frequency vs. Time');
```

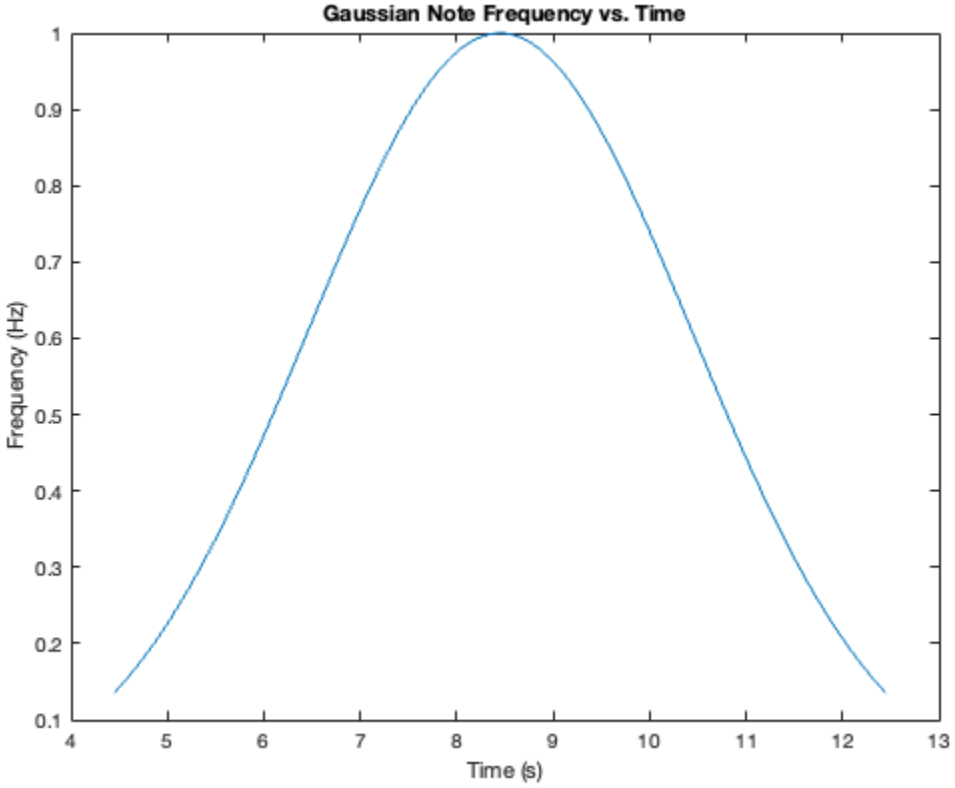
```
figure;
plot(logf293_66, GD293_66);
xlabel('Time (s)');
ylabel('Frequency (Hz)');
title('Gaussian Note Frequency vs. Time');

figure;
plot(logf329_63, GE329_63);
xlabel('Time (s)');
ylabel('Frequency (Hz)');
title('Gaussian Note Frequency vs. Time');

figure;
plot(logf349_23, GF349_23);
xlabel('Time (s)');
ylabel('Frequency (Hz)');
title('Gaussian Note Frequency vs. Time');

figure;
plot(logf392, GG392);
xlabel('Time (s)');
ylabel('Frequency (Hz)');
title('Gaussian Note Frequency vs. Time');

figure;
plot(logf440, GA440);
xlabel('Time (s)');
ylabel('Frequency (Hz)');
title('Gaussian Note Frequency vs. Time');

figure;
plot(logf493_88, GB493_88);
xlabel('Time (s)');
ylabel('Frequency (Hz)');
title('Gaussian Note Frequency vs. Time');

figure;
plot(logf523_25, GC523_25);
xlabel('Time (s)');
ylabel('Frequency (Hz)');
title('Gaussian Note Frequency vs. Time');

% Q4d - Produce the illusion by using the Gaussian weight function

sig = 0.75;
Fs1 = 22050;

[DC261_63] = MusicalWeightingDiscrete(261.63, sig);
[DDf271_18] = MusicalWeightingDiscrete(271.18, sig);
[DD293_66] = MusicalWeightingDiscrete(293.66, sig);
[DEf311_13] = MusicalWeightingDiscrete(311.13, sig);
[DE329_63] = MusicalWeightingDiscrete(329.63, sig);
[DF349_23] = MusicalWeightingDiscrete(349.23, sig);
```

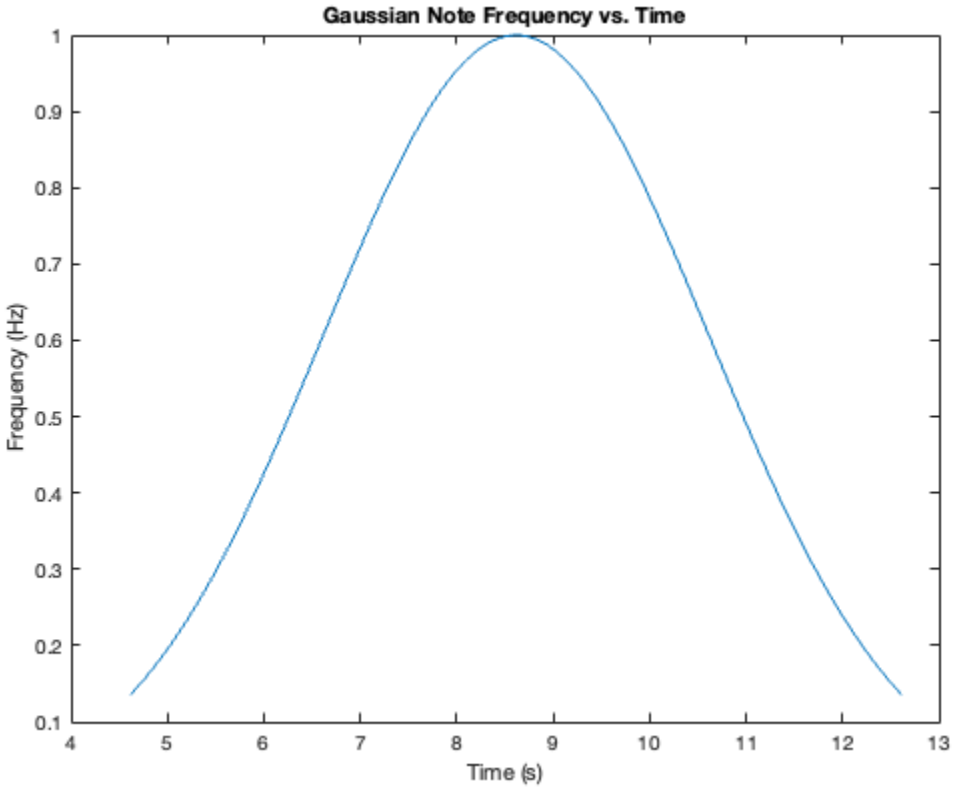
```
[DGf369_99]     = MusicalWeightingDiscrete(369.99, sig);
[DG392]     = MusicalWeightingDiscrete(392, sig);
[DAb415_30]     = MusicalWeightingDiscrete(415.30, sig);
[DA440]     = MusicalWeightingDiscrete(440, sig);
[DBb466_16]  = MusicalWeightingDiscrete(466.16, sig);
[DB493_88]  = MusicalWeightingDiscrete(493.88, sig);


silVector = zeros(1, round(Fs1 * 0.4));

audiovector = [DC261_63, silVector, DD293_66, silVector, DE329_63, silVector,
DF349_23, silVector, DG392, silVector, DA440, silVector, DB493_88, silVector];

repeatedAudio = repmat(audiovector, 1, 3);
soundsc(repeatedAudio, Fs1);

% Q4e - Make a spectrogram of playing the scale three times, there is a
% clear illusion when observing the spectrogram. You can see that the
% frequencies increase as the scale progresses, and that there is an
% overlap between the higher pitched notes vs. the lower ones, this overlap
% is producing the effect because the listener perceives the overlap as a
% continuous sound climbing the frequency ladder.

window = hamming(1024);
noverlap = 512;
nfft = 1024;
spectrogram(repeatedAudio, window, noverlap, nfft, Fs1, 'yaxis');

% Q4f - Variations: Play every note within the octave, the illusion does sound
% better because it has more frequencies in between, reducing the step
% makes the illusion more convincing because it's not as much as an
% auditory jump so the listener perceives the differences as marginal.

audiovector1 = [DC261_63, silVector, DDf271_18, silVector, DD293_66,
silVector, DEf311_13, silVector, DE329_63, silVector, DF349_23, silVector,
DGf369_99, silVector, DG392, silVector, DAb415_30, silVector, DA440,
silVector, DBb466_16, silVector, DB493_88, silVector];

repeatedAudio1 = repmat(audiovector1, 1, 3);
soundsc(repeatedAudio1, Fs1);
spectrogram(repeatedAudio1, window, noverlap, nfft, Fs1, 'yaxis');

filename = 'mywav.wav';
normrepeatedaudio1 = repeatedAudio1 / (max(abs(repeatedAudio1)));

audiowrite(filename, normrepeatedaudio1, Fs1);
```

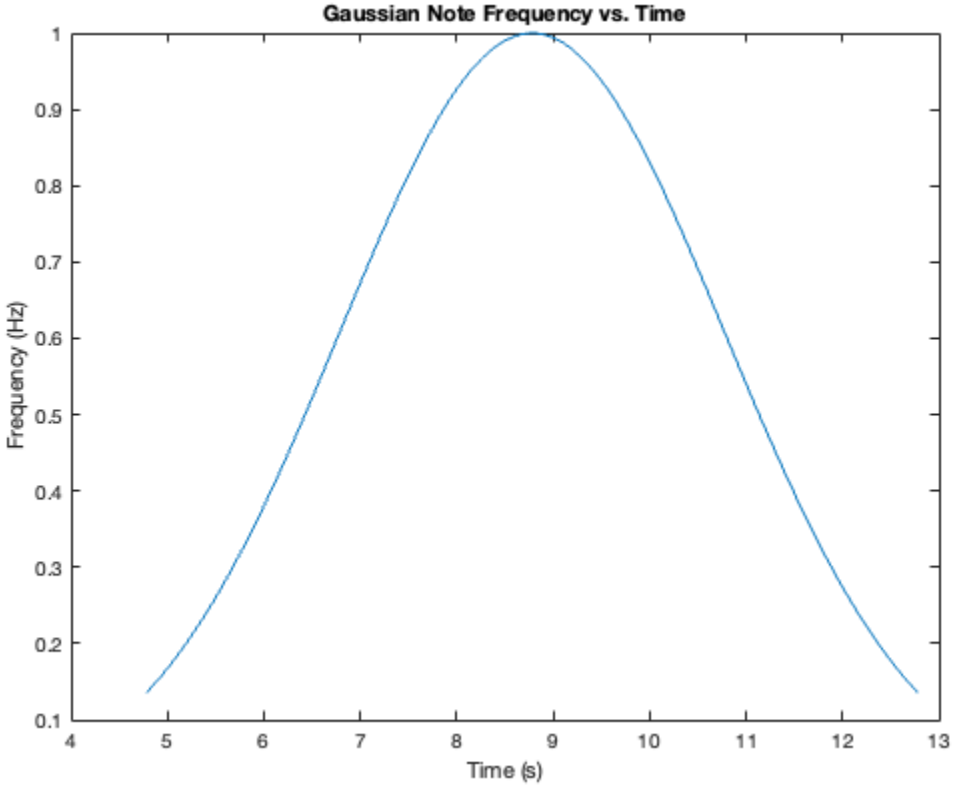

Gaussian Note Frequency vs. Time
Frequency (Hz)
Time (s)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
4
5
6
7
8
9
10
11
12
13

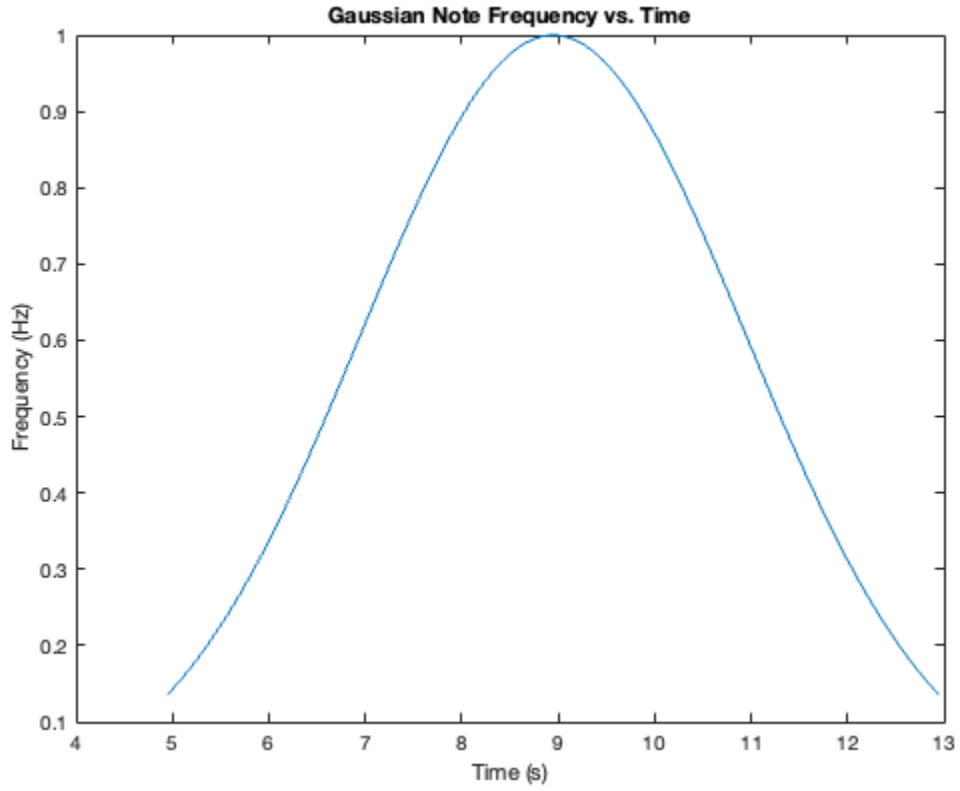

Gaussian Note Frequency vs. Time
Frequency (Hz)
Time (s)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
4
5
6
7
8
9
10
11
12
13

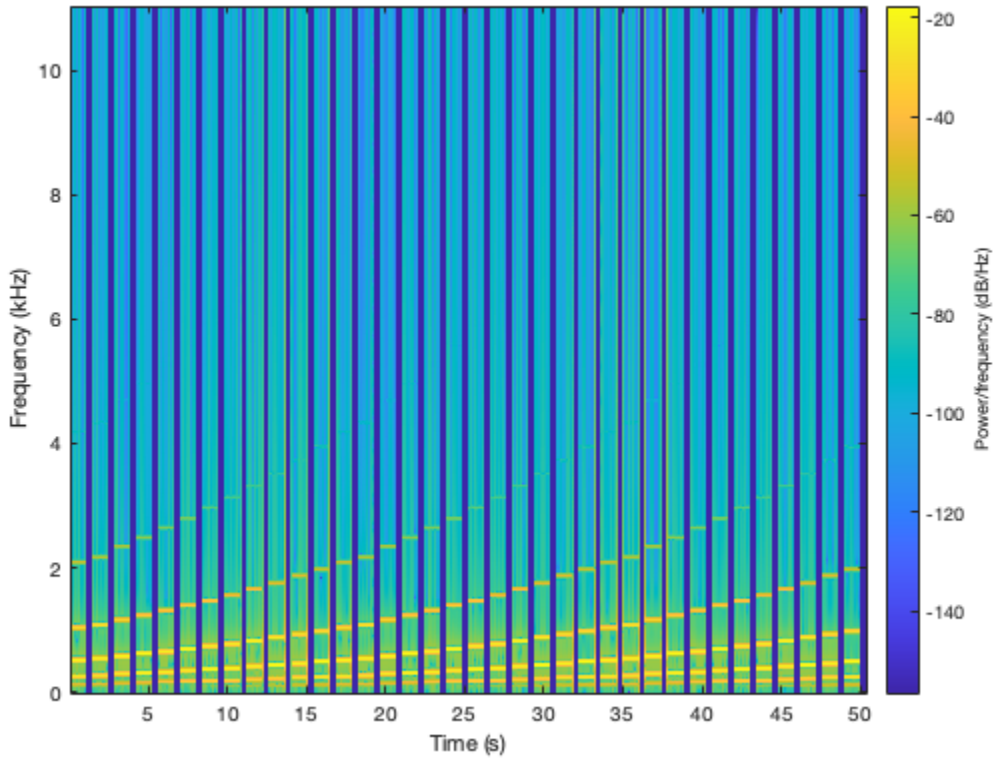

Gaussian Note Frequency vs. Time
Frequency (Hz)
Time (s)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
4
5
6
7
8
9
10
11
12
13


Gaussian Note Frequency vs. Time
Frequency (Hz)
Time (s)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
4
5
6
7
8
9
10
11
12
13


Gaussian Note Frequency vs. Time
Frequency (Hz)
Time (s)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
4
5
6
7
8
9
10
11
12
13


Gaussian Note Frequency vs. Time
Frequency (Hz)
Time (s)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
4
5
6
7
8
9
10
11
12
13


Gaussian Note Frequency vs. Time
Frequency (Hz)
Time (s)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
4
5
6
7
8
9
10
11
12
13


Frequency (kHz)
Time (s)
Power/frequency (dB/Hz)
10
8
6
4
2
0
5
10
15
20
25
30
35
40
45
50
-20
-40
-60
-80
-100
-120
-140

Published with MATLAB® R2024a

```
function [output, ff] = MusicalWeighting(fc, sig)

ff = linspace(log2(fc/16), log2(fc*16), 100);

output = exp( (-(ff - log2(fc)).^2) / (2*sig^2) );

end
```

```
Not enough input arguments.

Error in MusicalWeighting (line 3)
ff = linspace(log2(fc/16), log2(fc*16), 100);
```

Published with MATLAB® R2024a

```
function [weighted_signal] = MusicalWeightingC(f_c, sig)

% Define harmonic frequencies
f5 = f_c*2;
f6 = f5*2;
f3 = f_c/2;
f2 = f3/2;

fs = 8000;

fcenter = 440;

% Array of harmonic frequencies
ff_c = [f2, f3, f_c, f5, f6];

% Generate Gaussian weighting for these frequencies
output = exp( (-(log2(ff_c) - log2(fcenter)).^2) / (2*sig^2) );

% Define a time vector for constructing the signal (1 second duration)
t = 0:1/fs:2;

% Initialize the weighted signal
weighted_signal = zeros(size(t));

% Sum the weighted cosine terms for each frequency over time
for i = 1:length(ff_c)
    weighted_signal = weighted_signal + output(i) * cos(2*pi*ff_c(i)*t);
end

end
```

```
Not enough input arguments.

Error in MusicalWeightingC (line 4)
f5 = f_c*2;
```

Published with MATLAB® R2024a

```
function [weighted_signal] = MusicalWeightingDiscrete(fc, sig)

% Define harmonic frequencies
f6 = fc*2;
f7 = f6*2;
f8 = f7*2;
f9 = f8*2;
f4 = fc/2;
f3 = f4/2;
f2 = f3/2;
f1 = f2/2;

fs = 22050;

% Array of harmonic frequencies
ff_discr = [f1, f2, f3, f4, fc, f6, f7, f8, f9];

% Generate Gaussian weighting for these frequencies
output = exp( (-(log2(ff_discr) - log2(500)).^2) / (2*sig^2) );

% Define a time vector for constructing the signal (1 second duration)
t = 0:1/fs:1;

% Initialize the weighted signal
weighted_signal = zeros(size(t));

% Sum the weighted cosine terms for each frequency over time
for i = 1:length(ff_discr)
    weighted_signal = weighted_signal + output(i) * cos(2*pi*ff_discr(i)*t);
end

end
```

```
Not enough input arguments.

Error in MusicalWeightingDiscrete (line 4)
f6 = fc*2;
```

Published with MATLAB® R2024a

```
function [output, sampling_freq] = NoteHarmonics(base_freq)

% Harmonics
f6 = base_freq*2;
f7 = f6*2;
f8 = f7*2;
f9 = f8*2;
f4 = base_freq/2;
f3 = f4/2;
f2 = f3/2;
f1 = f2/2;

% Sampling Info
sampling_freq = f9 * 10;
Ts = 1/sampling_freq;

% function output
t = [0 : Ts : 2];
output = cos(2*pi*f1*t) + cos(2*pi*f2*t) + cos(2*pi*f3*t) + cos(2*pi*f4*t) +
cos(2*pi*base_freq*t) + cos(2*pi*f6*t) + cos(2*pi*f7*t) + cos(2*pi*f8*t)  +
cos(2*pi*f9*t);
```

```
Not enough input arguments.

Error in NoteHarmonics (line 4)
f6 = base_freq*2;
```

Published with MATLAB® R2024a