Please refer to these links
Links in reports are access denied , so please use these links. Ignore them

Same frequency
https://drive.google.com/file/d/1rzC9p6KdXiGsa97UW1dz-sepqffZ2zie/view?usp=sharing

Song A part 1
https://drive.google.com/file/d/1hMEc56Ps8m3qk2MPCVfEQ0UxiUxEe6gj/view?usp=sharing

Song A part 2
https://drive.google.com/file/d/1chRpzM0NOmG44HrdCLTkhSXTkkdaT9rP/view?usp=sharing

Song b
https://drive.google.com/file/d/1uV-ZlgXv7N7rcCMNnInw41IIctBrdan9/view?usp=sharing

EE229
Computing Assignment -2
Dikshit Singla
23b3974

Introduction

The goal of this assignment was to create two songs given information and values. Also, some interesting sinusoidal waveform with fixed frequency but different amplitudes and phases. Then analyze its timbre and richness. The implementation was carried out in Scilab using the provided skeletal framework.

Concatenation of Sinusoids:

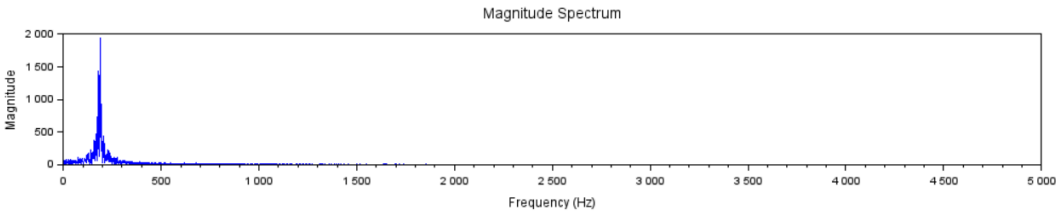
By concatenating sinusoids of different frequencies and durations, we simulate a sequence of notes, as in a musical piece. This allows for studying how different frequency transitions and durations influence the perception of sound.

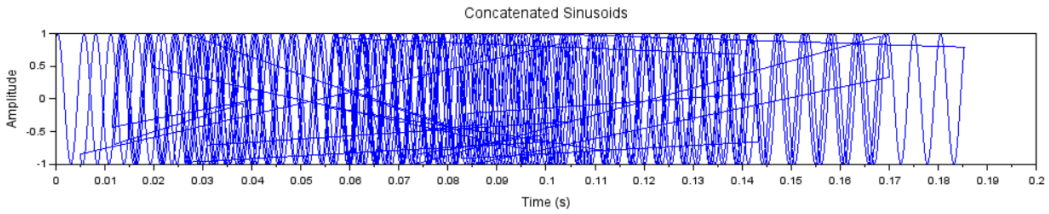
A sinusoid and skeleton code was given to us and it was stated that turn your sinusoid to a more interesting waveform with the same fundamental frequency (Fo) but different waveshape by adding in more harmonics of various amplitudes and phases.

So, I fixed the frequency to 185 hz and put amplitude, time duration, and phase as rand() function and made specification as rand() ranges from 0 to 1 only. I added 20 harmonics and I got some random sound which is just noise as compared to nice sound I got in songs A, and B. And magnitude spectra and waveform -
Files-
https://drive.google.com/drive/folders/1HApR7dbU6tvqCBwbPkF1U4XwJ01HqwsL?usp=drive_link

"**Amplitudes**: 0.3873779, 0.3435337, 0.2615761, 0.5253563, 0.2256303, 0.0485566, 0.3911574, 0.4829179, 0.1205996, 0.8494102, 0.6488563, 0.7485507, 0.8544211, 0.9262344, 0.816011, 0.124934, 0.5465335, 0.0037173, 0.2552206, 0.6117004"

"**phi**:5.7949181, 2.3625523, 3.1375047, 3.3779848, 3.9421289, 4.2247822, 5.2152427, 1.4029505, 1.7940782, 3.3031087, 6.2349248, 2.5786563, 0.4037868, 3.5608139, 0.357468, 4.5736703, 6.2111848, 3.7074393, 3.9281717, 4.2624854"





Sampling Frequency- 10,000hz
Sample Code for one harmonic-

```
F0 = 196;
a = rand();      // Random amplitude between 0 and 1
phi = rand()*2*%pi;  // Random phase between 0 and 2*pi
T = 200;    //(ms)
[y7,t8] = get_continuous_sinusoid(a,F0,phi,T);
```

1. **Waveform (Time Domain)**: This plot shows the amplitude of the sound wave over time. From the waveform, we can observe the overall shape and envelope of the signal, which can indicate its duration and intensity variations.

Code to generate it-

```
clf();
xsetech([0.1, 0.55, 0.8, 0.35]);// Define plot area for first plot (left, bottom, width, height)
plot(t_combined, z, 'b');
xlabel('Time (s)');
ylabel('Amplitude');
title('Concatenated Sinusoids');
```

2. **Magnitude Spectrum (Frequency Domain)**: This plot represents the distribution of signal energy across different frequencies. Peaks in the spectrum correspond to the fundamental frequency and its harmonics.

Code to generate this graph-

```
n = length(z);              // Length of the concatenated signal
Z_fft = fft(z);            // Perform FFT
frequencies = (0:n-1) * 10000 / n;  // Frequency axis (assuming sample rate 10 kHz)
magnitude_spectrum = abs(Z_fft);    // Magnitude of FFT values

// Plot 2: Magnitude Spectrum (Frequency Domain)
xsetech([0.1, 0.1, 0.8, 0.35]);  // Define plot area for second plot (left, bottom, width, height)
```

```
plot(frequencies(1:n/2), magnitude_spectrum(1:n/2));
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Magnitude Spectrum');
```

FFT Calculation:

- `fft(z)`: This computes the Fast Fourier Transform of the concatenated signal `z`.
- `frequencies`: This array defines the frequency axis, based on the length of the signal `n` and assuming a sample rate of 10,000 Hz (since we used `playsnd(z, 10000)`).
- `magnitude_spectrum = abs(Z_fft)`: This gives the magnitude of the FFT results.

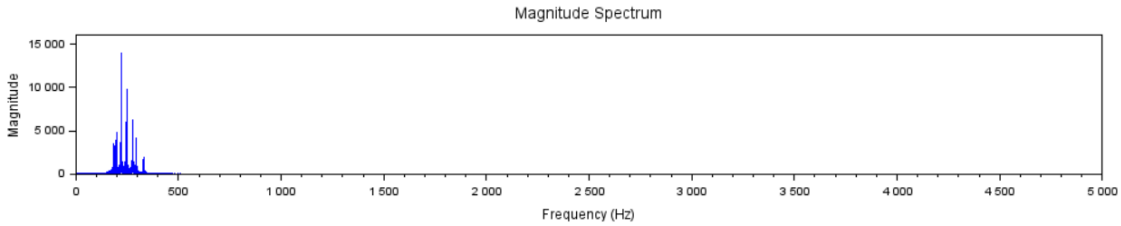
To generate the final output these are code snippets-

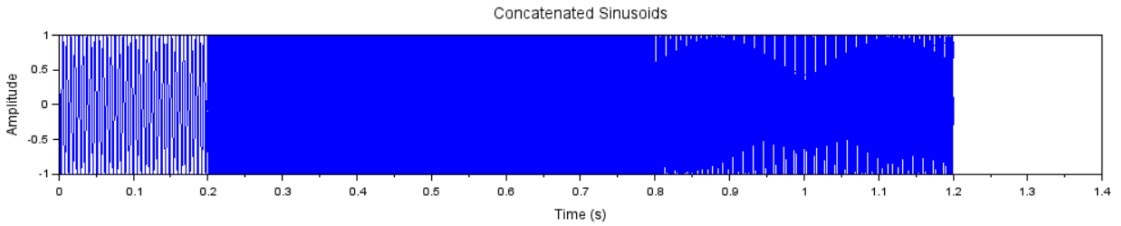
```
// Play the combined waveform
playsnd(z, 10000);

// Save the combined sound to output.wav
wavwrite(z, 10000, "output.wav");
```

- **Insights**
  Harmonics and Timbre: The presence of multiple peaks in the magnitude spectrum indicates harmonics, which contribute to the timbre or "color" of the sound. A richer timbre is typically associated with more harmonics at integer multiples of the fundamental frequency.

SONG A

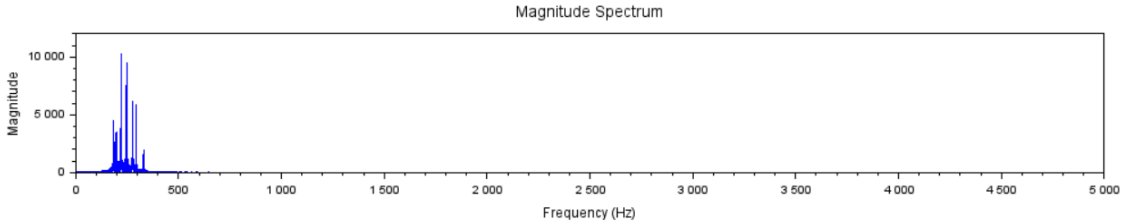
Below plot is obtained by keeping some of the amplitude and phi values as rand() function and some numerical values by wish
Files-
https://drive.google.com/drive/folders/1FsYcMHRm3Kj5it9QctsTp1eRSDwdwIFX?usp=drive_link

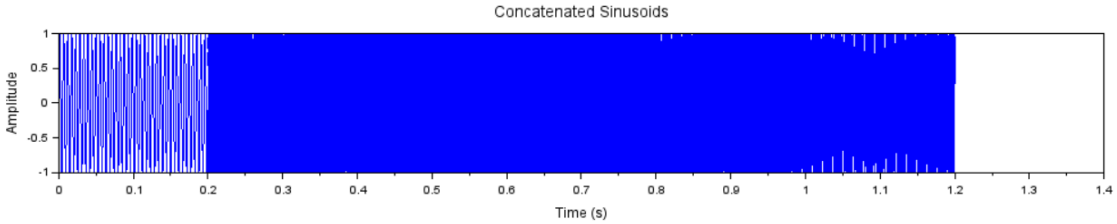




Below plot is obtained by keeping all amplitude and phi as rand() function along with respective frequency and time duration values.
Files-
https://drive.google.com/drive/folders/14D5DDG04lRs2Xo9UcjZr2HS16tdy9o6U?usp=drive_link

Amplitudes:
0.3760119, 0.2615761, 0.2638578, 0.537623, 0.2256303, 0.7608433, 0.672395, 0.3911574, 0.587872, 0.2232865, 0.1205996, 0.8607515, 0.5257061, 0.6488563, 0.050042, 0.4104059, 0.8544211, 0.8279083, 0.5667211, 0.816011, 0.5595937, 0.7279222, 0.5465335, 0.7395657, 0.5900573, 0.2552206, 0.1157417, 0.6783956, 0.025871, 0.3916873, 0.5064435, 0.2893728"





Little difference is there between the graphs

SONG B

Files-

https://drive.google.com/drive/folders/1ZcC4CnkiXhjSh0BFCQ2Xj4WdhHakSuDd?usp=drive_link

Below plot is obtained by keeping all amplitude and phi as rand() function along with respective frequency and time duration values.

amplitude =rand()

Value of amplitudes-

"0.2893728"
"0.6212882"
"0.7064868"
"0.2870401"
"0.0881335"
"0.7227253"
"0.2427822"
"0.9677053"
"0.5232976"
"0.5617307"

"0.7794547"
"0.9808542"
"0.4256872"
"0.9229532"
"0.4678218"
"0.0366117"
"0.8325452"
"0.1871112"
"0.8433565"
"0.8532815"
"0.1867539"
"0.7489608"
"0.2124056"
"0.2628148"
"0.9110545"
"0.8102653"
"0.4139087"
"0.6912788"
"0.357265"
"0.5477634"
"0.9561172"
"0.0143259"
"0.1304993"
"0.6561381"
"0.5283124"
"0.7876622"
"0.7883861"
"0.2659857"
"0.8875248"
"0.8525161"
"0.9152874"
"0.2367841"
"0.1202527"
"0.3161073"
"0.5715175"
"0.824862"
"0.2791808"
"0.9071155"
"0.1175613"
"0.7263671"
"0.3948993"
"0.706149"
"0.4132936"
"0.4952356"

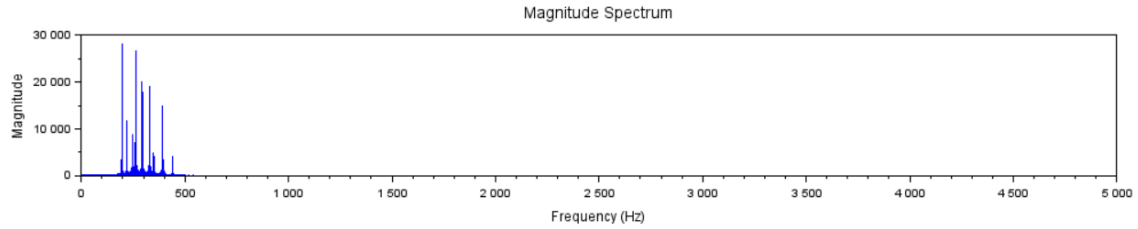
"0.8626222"
"0.2512136"
"0.3921976"
"0.3361603"
"0.2039064"
"0.0181815"
"0.0105835"
"0.2725595"
"0.2033702"

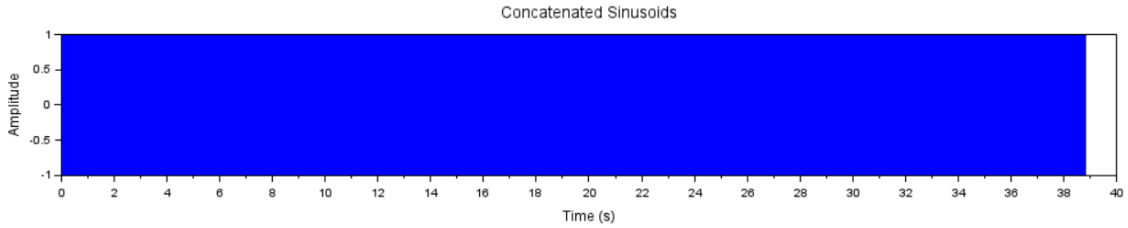
Phi =rand()*2*pi
phi=
"0.2113249"
"4.7503636"
"0.0002211"
"2.0755063"
"0.6653811"
"3.9483021"
"0.8497452"
"4.3085751"
"0.8782165"
"0.4296067"
"0.5608486"
"4.1617114"
"0.7263507"
"1.2473027"
"0.5442573"
"1.4581689"
"0.2312237"
"1.3600788"
"0.8833888"
"4.0998632"
"0.3076091"
"5.8619708"
"0.2146008"
"1.9643876"
"0.3616361"
"1.8361143"
"0.5664249"
"3.0325618"
"0.3321719"
"3.72913"
"0.5015342"

"2.7448645"
"0.2693125"
"3.9745827"
"0.4051954"
"5.7709221"
"0.0437334"
"3.0275585"
"0.2639556"
"2.6063304"
"0.2806498"
"0.8042845"
"0.7783129"
"1.3314261"
"0.1121355"
"4.3083148"
"0.1531217"
"4.3799146"
"0.8415518"
"2.5522454"
"0.4094825"
"5.519229"
"0.113836"
"1.2555926"
"0.5618661"
"3.7046775"
"0.685398"
"5.595946"
"0.5042213"
"2.1951033"
"0.3873779"
"5.7949181"
"0.9488184"
"2.1584861"
"0.3760119"
"4.612449"
"0.2615761"
"3.1375047"
"0.2638578"
"3.300911"
"0.537623"
"0.7539354"
"0.2256303"
"3.9421289"
"0.7608433"

"0.3050902"
"0.672395"
"1.267427"
"0.3911574"
"5.2152427"
"0.587872"
"3.0342628"
"0.2232865"
"5.2784321"
"0.1205996"
"1.7940782"
"0.8607515"
"5.3370015"
"0.5257061"
"6.2399632"
"0.6488563"
"6.2349248"
"0.050042"
"4.7032825"
"0.4104059"
"3.8230206"
"0.8544211"
"0.4037868"
"0.8279083"
"5.8197022"
"0.5667211"
"3.5887286"
"0.816011"
"0.357468"
"0.5595937"
"0.7849837"
"0.7279222"
"1.6824903"
"0.5465335"
"6.2111848"
"0.7395657"
"0.0233566"
"0.5900573"
"1.9455678"
"0.2552206"
"3.9281717"
"0.1157417"
"3.843427"
"0.6783956"

"2.0860774"
"0.025871"
"3.2512143"
"0.3916873"
"1.5164709"
"0.5064435"
"2.6616214"





Harmonics and Sound Quality: By adding more harmonics, the sound becomes richer, resembling musical instrument tones. A simple sine wave (just the fundamental) sounds pure but not as full, while additional harmonics create a fuller sound.

Practical Application:

This type of waveform synthesis has applications in music production, digital sound design, and signal processing. By controlling the harmonic content, we can model the sound of musical instruments, generate sound effects, or analyze complex waveforms.

These theoretical insights provide a comprehensive understanding of the relationship between frequency, harmonics, and timbre, giving a foundation for the practical experiments you've conducted with sinusoidal waveforms.