ECE280 - Lab 1: Music Synthesis

Ace Abdulrahman (aka40)

February 10, 2022

I have adhered to the Duke Community Standard in completing this assignment.

Ace Abdulrahman **[aka40]**

Contents

1 Introduction

- Motivation

  The chosen song is the theme song from the popular TV series "The Office." Played at the opening of every episode, the song is easily recognizable to the show's wide audience,and was chosen for its "catchiness," reputability, and lack of complexity.

- Effects

  I. Echo

  II. Harmonics

  III. Attack, Decline, Sustain, Release Envelopes (ADSR)

2 Procedure

- Explanation

  A sheet music for the theme song was found on the website Musescore.com. The notes for the treble and base clefs were handwritten on paper. In MATLAB, a function that outputs notes based on measure, amplitude, octave, and pitch passed as arguments was created (the function also implements harmonics and ADSR effects). Afterwards, variables representing all notes in the sheet music were declared at the top of the script. Three arrays were used to store the notes in the treble clef and two for the base clef. After normalizing the arrays to avoid clipping, all 5 arrays were added together to make one big array called SongArr, which also gets normailized. The echo effect is then created, before further normalization. The soundsc function is then used to create the sound output with 8000 sampling frequency.

- Effects

  I. Echo

  The echo effect was implemented by creating an array of 900 zeros using the zeros function and padding it to the right of one copy of SongArr (named one) and to the left of another copy (named two). To reduce the amplitude in the echoed sound, the SongArr array is multiplied by 0.4 in the two array. The one and two arrays are then added together and stored in a variable named SA.

  II. Harmonics

  In MATLAB, harmonics were created inside the m function using fourier series by summing up waves of different amplitudes at different frequencies for each note. The waves created were multiplied by high amplitude at low frequencies and low amplitude at higher frequencies as they are being summed together. This pattern mimics that of piano keys.

  III. ADSR Envelopes

  ADSR is created inside the m function. Only attack, decline, and sustenance phases were implemented. To create the ADSR envelope in MATLAB, the linspace function was used to create linearly space values between the starting amplitude and ending amplitude for each phase. The duration of each phase was determined by multiplying the length of the note with the proportion of time each note is desired to occupy. These arrays passed by the linspace function were then combined to make a larger array with variable name ADSR in the code, which was then element wise multiplied with the output of the function.

3 Results

The sound output highly resembles the piano used in the original theme. The notes lengths and overall beat also match the original theme. However, there is no sound similar to the saxophone in the sound output as is used in original theme. The echo effect is audible and is not part of the original theme. Overall, however, it sounds incredibly similar to the original piece and is easily recognizable.

4 Discussion

- Discuss how each effect individually affects the sound.

  A) Multiple Notes:

  Allowing for multiple sounds simultaneously, multiple notes gives the sound output more subtle details. It adds depth and intricacy to the sound output.

  B) Exponential Decay:

  Exponential decay allows for smoother transitions between notes rather than abrupt ones. It makes the notes in the sound output feel more connected, as if the previous note is picking up from where the last note left.

  C) Harmonics:

  Harmonics allow for the sound output to mimic the sound of an actual musical instrument. It adds realism to the sound output.

  D) Instrument Synthesis:

  The amplitude and wave modulation in instrument synthesis also allow for mimicking real instruments. They add realism.

  E) Reverb/Echo:

  Reverb/Echo makes the sound output sound as if it's being played in an empty room.

- What challenges did you encounter, if any, when implementing each effect? How did you resolve the challenges?

  The biggest challenge was implementing ADSR without ruining the sound of the piano from harmonics. It was solved by trial and error. The solution was raising the starting amplitude in the attack phase such that it's not starting at zero as well as making it rapid; shortening the decay phase; lengthening the sustenance phase; and removing the release phase. These changes preserved the echo and harmonics effects.

- Which effects had the greatest impact on the sound? Which had the least impact? Why?

  The harmonics definitely had the greatest impact on the quality of the song. Before implementing them, the song was vague and was not immediately recognizable as it didn't match the sound of any instrument. It sounded like the output was generated using beat box. After adding the harmonics, the sound output was more lively due to mimicking the piano.

- Assuming you had more time, how would you make the song better? Why do you think this would improve your song?

  I would start by adding another pattern of harmonics that mimics the saxophone. In the original theme, there is a clear horn instrument being used. Implementing this effect would increase the resemblance and make the output sound less lackluster. I would also modulate the amplitude of the base notes to emphasize certain sections.

I. Why is the endpoint $1 - \frac{1}{f_s}$ instead of 1?

This is done to ensure that the 1 is not included when incrementing by the step size from 0.

II. Does it matter if you use sin or cos?

Yes, it does matter whether sin or cos is used. Cosine starts with high amplitude in standard form where as sin starts at 0 in standard form. sin(0)=0=cos(pi/2) , sin(pi/2)=1=cos(0)

III. Why do you need to use vectors of zero amplitude?

0 amplitude is equivalent to no sound being generated which allows for silence/delays. It also allows for padding of the arrays when creating reverbs/delays/echoes.

IV. What happens when the amplitude signal exceeds the range between -1 and 1?

The result is an undesirable, because the sound command only accepts amplitudes between -1 and 1. It results in what is known as clipping.

5 MATLAB^{TM} code

(a) FUNCTION NAME: m

This function is used to create all the notes. It takes in four arguments; amplitude, let, measure, and octave to determine the length, volume, octave and type of note; in addition to adding the harmonics and ADSR effects.

```
%% m by Ace Abdulrahman for ECE 280 Lab 1
% The m function takes in 4 parameters Amplitude, let, measure, and octave
% Amplitude: used to determine the amplitude of the sound
% let: used for letter notation to determine the n value that affects pitch
% measure: used to determine the measure by taking the inverse of value passed
% (4 passed as measure would create a note of measure 1/4)
% octave: used to determine the octave of the note.

function [output] = m(Amplitude,let, measure,octave)
%% Declaring variables
fs = 8000;          % Sampling frequency
n=0;                % Declaring n for switch
tpm = 1.45;         % Tempo variable, some beat per minute values create round off error so
meas = [0 : 1/fs: tpm/measure-1/fs]; % meas used for the note measure

%% ADSR Effect
A=linspace(0.8,1,0.02*length(meas));        % Rapid attack
D=linspace(1,0.8,0.2*length(meas));         % Decline
S=linspace(0.8,0.8,0.78*length(meas));      % SUSTAINANCE
% R=linspace(1,0,0.98*length(meas));        % No release

ADSR=[A D S];                               % No R

%% Switch flow to determine n value used in note
switch let
    case 'A'
        n=0;
    case 'B'
        n=2;
    case 'C'
        n=3;
```

```
    case 'D'
        n=5;
    case 'E'
        n=7;
    case 'F#'
        n=9;
    case 'G'
        n=10;
    otherwise
        n=0;
end

%% Applying Harmonics to Output
output =  1 * cos(2 * pi * 1 * 220 * 2^( (octave-3) + n/12 ) * meas) + ...
        0.1 * cos(2 * pi * 2 * 220 * 2^( (octave-3) + n/12 ) * meas) + ...
       0.35 * cos(2 * pi * 3 * 220 * 2^( (octave-3) + n/12 ) * meas) + ...
       0.06 * cos(2 * pi * 4 * 220 * 2^( (octave-3) + n/12 ) * meas) + ...
       0.05 * cos(2 * pi * 5 * 220 * 2^( (octave-3) + n/12 ) * meas) + ...
       0.04 * cos(2 * pi * 6 * 220 * 2^( (octave-3) + n/12 ) * meas) + ...
       0.00 * cos(2 * pi * 7 * 220 * 2^( (octave-3) + n/12 ) * meas) + ...
       0.02 * cos(2 * pi * 8 * 220 * 2^( (octave-3) + n/12 ) * meas) + ...
       0.01 * cos(2 * pi * 9 * 220 * 2^( (octave-3) + n/12 ) * meas) + ...
       0.01 * cos(2 * pi * 10 * 220 * 2^( (octave-3) + n/12 ) * meas)+ ...
       0.00 * cos(2 * pi * 11 * 220 * 2^( (octave-3) + n/12 ) * meas)+ ...
       0.01 * cos(2 * pi * 12 * 220 * 2^( (octave-3) + n/12 ) * meas)+ ...
       0.015 * cos(2 * pi *13 * 220 * 2^( (octave-3) + n/12 ) * meas) + ...
       0.00 * cos(2 * pi * 14 * 220 * 2^( (octave-3) + n/12 ) * meas) + ...
       0.0 * cos(2 * pi *15 * 220 * 2^( (octave-3) + n/12 ) * meas) + ...
       0.0 * cos(2 * pi *16 * 220 * 2^( (octave-3) + n/12 ) * meas) + ...
       0.0 * cos(2 * pi *17 * 220 * 2^( (octave-3) + n/12 ) * meas) + ...
       0.0 * cos(2 * pi *18 * 220 * 2^( (octave-3) + n/12 ) * meas) + ...
       0.0 * cos(2 * pi *19 * 220 * 2^( (octave-3) + n/12 ) * meas) + ...
       0.0 * cos(2 * pi *20 * 220 * 2^( (octave-3) + n/12 ) * meas);

%% Applying ADSR to output
output= output.*ADSR;
end
```

Office Script

```
%% ECE 280 Lab 1 - Music Synthesis
%    "The Office" Theme Song
%    Author: Ace Abdulrahman
%    Emulated Sheet Music: https://musescore.com/user/22944241/scores/4875944

%% Declaring Note Variables
C4_2=m(1,"C",2, 4);
A3_1=m(1,"A",1, 3);
A3_2=m(1,"A",2, 3);
A3_4=m(1,"A",4, 3);
A3_8=m(1,"A",8, 3);
A4_8=m(1,"A",8, 4);
B3_1=m(1,"B",1, 3);
B3_2=m(1,"B",2, 3);
B3_4=m(1,"B",4, 3);
B4_4=m(1,"B",4, 4);
B3_8=m(1,"B",8, 3);
C3_1=m(1,"C",1, 3);
C3_2=m(1,"C",2, 3);
C3_4=m(1,"C",4, 3);
C4_4=m(1,"C",4, 4);
C4_8=m(1,"C",8, 4);
C3_8=m(1,"C",8, 3);
D3_1=m(1,"D",1, 3);
D3_2=m(1,"D",2, 3);
D3_4=m(1,"D",4, 3);
D3_8=m(1,"D",8, 3);
E3_1=m(1,"E",1, 3);
E3_2=m(1,"E",2, 3);
E3_4=m(1,"E",4, 3);
E4_4=m(1,"E",4, 4);
E3_8=m(1,"E",8, 3);
FS3_1=m(1,"F#",1, 3);
FS3_2=m(1,"F#",2, 3);
FS3_4=m(1,"F#",4, 3);
FS3_8=m(1,"F#",8, 3);
F3_1=m(1,"F",1, 3);
F3_2=m(1,"F",2, 3);
F3_4=m(1,"F",4, 3);
F3_8=m(1,"F",8, 3);
G3_1=m(1,"G",1, 3);
G3_2=m(1,"G",2, 3);
G3_4=m(1,"G",4, 3);
G3_8=m(1,"G",8, 3);
G4_1=m(1,"G",1, 4);

C2_1=m(1,"C",1, 2);
C2_2=m(1,"C",2, 2);
C2_4=m(1,"C",4, 2);
C2_8=m(1,"C",8, 2);
D2_1=m(1,"D",1, 2);
D2_2=m(1,"D",2, 2);
```

```
D2_4=m(1,"D",4, 2);
D2_8=m(1,"D",8, 2);
E2_1=m(1,"E",1, 2);
E2_2=m(1,"E",2, 2);
E2_4=m(1,"E",4, 2);
E2_8=m(1,"E",8, 2);
F2_1=m(1,"F#",1, 2);
F2_2=m(1,"F#",2, 2);
F2_4=m(1,"F#",4, 2);
F2_8=m(1,"F#",8, 2);
G2_1=m(1,"G",1, 2);
G2_2=m(1,"G",2, 2);
G2_4=m(1,"G",4, 2);
G4_1=m(1,"G",1, 4);
G2_8=m(1,"G",8, 2);

E2_34= m(1,"E",4/3, 2);
E3_34= m(1,"E",4/3, 3);
B2_4=m(1,"B",4, 2);
B4_8=m(1,"B",8, 4);
B2_8=m(1,"B",8, 2);
D4_8=m(1,"D",8, 4);
E4_8=m(1,"E",8, 4);
G4_78=m(1,"G",8/7, 4);
R1=m(0.00,"C",1, 3)

%% Treble Clef
Song=[D3_8, G2_8, D3_2, D3_8, G2_8,              D3_8, F2_8, D3_2, D3_8, F2_8,
...
      E3_8, G2_8, E3_2, E3_8, G2_8,              E3_4,  C3_4, C3_8, B3_8, A3_8, B3_8, ...
      D3_8, G2_8, D3_4, G3_4, D3_8, G2_8,        D3_8, F2_8, D3_8, G3_8, FS3_8, G3_8,FS3
      ...
      E3_8, G2_8, B3_2, B3_8, E3_8,              E3_4, C3_4, C3_8, B3_8, A3_8, B3_8,
...
      D3_8, G2_8, D3_4, G3_4,D3_8, G2_8,         D3_8, F2_8, D3_8, G3_8, FS3_8, G3_8,A4_8
      E3_8, G2_8, B3_4, D3_8,E3_8 D3_8, E3_8,    E3_4, C3_4, C3_8, B3_8, A3_8, B3_8,...
      ...
      D3_8, G2_8, D3_4, G3_2,                    G3_2, FS3_8,G3_8,FS3_8, D3_8,...
      E3_34,E3_8,G2_8,                           E3_4, C3_4, C3_8, B3_8, A3_8, B3_8,...
      D3_8, G2_8, D3_4, G3_2,...
      ...
      G3_2, FS3_8,G3_8,A4_8, FS3_8,              E3_2, D3_8,E3_8,D3_8, B3_8,...
      E3_4, C3_4, C3_8, B3_8, A3_8, B3_8,        G2_1,...
      D3_8, G3_8, D3_8, G3_8, B4_8, G3_8,B4_8, D4_8,     D4_8, G4_78];


Song2=[B3_8,G2_8, B3_2, B3_8, G2_8,              B3_8, F2_8, B3_2, B3_8, F2_8,
...
      B3_8, G2_8, B3_2, B3_8, G2_8,              C3_4,  C3_4, C3_8, B3_8, A3_8, B3_8, ...
      D3_8, G2_8, D3_4, G3_4, B3_8, G2_8,        B3_8, F2_8, B3_8, G3_8, FS3_8, G3_8,D3_8
      ...
      B3_8, G2_8, B3_2, B3_8, G2_8,              C3_4, C3_4, C3_8, B3_8, A3_8, B3_8,
...
      B3_8, G2_8, B3_4, G3_4, B3_8, G2_8,        B3_8, F2_8, B3_8, G3_8, FS3_8, G3_8,D3_8
```

```
      B3_8, G2_8, B3_4, D3_8, E3_8, D3_8,B3_8,  C3_4, E2_4, C3_8, B3_8, A3_8, B3_8,...
      ...
      B3_8, G2_8, B3_4, D3_2,                   D3_2, D3_8, D3_8, D3_8, G2_8,...
      E2_34,B3_8,G2_8,                          C3_4, E3_4, E2_8, B3_8, E2_8, B3_8,...
      B3_8, G2_8, B2_4, G2_2,...
      ...
      G2_2, D3_8, G3_8, D3_8, FS3_8,            E2_2, D3_8,E3_8,D3_8, G2_8,...
      C3_4, E2_4, E2_8, B3_8, E2_8, C3_8,       G2_1,...
      D3_8, G3_8, D3_8, G3_8, B4_8, G3_8,B4_8, D4_8,     D4_8, G4_78];


Song3=[D3_8, G2_8, D3_2, D3_8, G2_8,            D3_8, F2_8, D3_2, D3_8, F2_8,
...
      E3_8, G2_8, E3_2, E3_8, G2_8,             E3_4,  C3_4, C3_8, B3_8, A3_8, B3_8, ...
      D3_8, G2_8, G2_4, G3_4, D3_8, G2_8,       D3_8, F2_8, D3_8, G3_8, FS3_8, G3_8, B3
      ...
      E3_8, G2_8, B3_2, B3_8, E3_8,             E3_4, C3_4, C3_8, B3_8, A3_8, B3_8,
...
      D3_8, G2_8, G2_4, G3_4, D3_8,G2_8,        D3_8, F2_8, D3_8, G3_8, FS3_8, G3_8,B3_8
      E3_8, G2_8, B3_4, D3_8,E3_8 D3_8, G2_8,   E2_4, C3_4, C3_8, B3_8, A3_8, B3_8...
      ...
      D3_8, G2_8, D3_4, G3_2,                   G3_2, FS3_8,G3_8,FS3_8, D3_8,...
      E3_34,E3_8,G2_8,                          E3_4, C3_4, C3_8, B3_8, A3_8, B3_8,...
      D3_8, G2_8, D3_4, G3_2,...
      ...
      G3_2, B3_8,G3_8,B3_8, FS3_8,              E3_2, D3_8,E3_8,D3_8, B3_8,...
      E2_4, C3_4, C3_8, B3_8, A3_8, B3_8,       G2_1,...
      D3_8, G3_8, D3_8, G3_8, B4_8, G3_8,B4_8, D4_8,     D4_8, G4_78];



%% Base Clef
Base=[G3_1,                             B3_1,...
      G4_1,                             C4_2, C4_2,...
      G3_4, G3_4, G3_4, G3_4,           B3_4, B3_4, B3_4, B3_4,...
      ...
      E4_4, E4_4, E4_4, E4_4,           C4_2, C4_2,...
      G3_4, G3_4, G3_4, G3_4,           B4_4, B4_4, B4_4, B4_4,...
      E4_4, E4_4, E4_4, E4_4,           C4_4, C4_4,C3_8,G3_8,C4_8,G3_8,...
      ...
      G2_8, G3_8, G2_8, D3_8, G2_8, G3_8, D4_8,G3_8,       B2_8, B3_8, B2_8, F3_8, B2_8
      E2_8, E3_8, E2_8, B2_8, E2_8, E3_8, B3_8,E3_8,       C3_8, C4_8, C3_8, G3_8, C3_8
      G2_8, G3_8, G2_8, D3_8, G2_8, G3_8, D4_8,G3_8,       ...
      ...
      B2_8, B3_8, B2_8, F3_8, B2_8, B3_8, E4_8, B3_8,      E2_8, E3_8, E2_8, B2_8, E2_8
      C3_8, C4_8, C3_8, G3_8, C3_8, C4_8, G3_8,C4_8,       G3_1,...
      G3_1,                                                R1];

Base2=[G3_1,                            B3_1,...
      G4_1,                             C4_2, C4_2,...
      G3_4, G3_4, G3_4, G3_4,           B3_4, B3_4, B3_4, B3_4,...
      ...
      E4_4, E4_4, E4_4, E4_4,           G3_2, G3_2,...
      G3_4, G3_4, G3_4, G3_4,           B4_4, B4_4, B4_4, B4_4,...
```

```
E4_4, E4_4, E4_4, E4_4,        C3_4, C3_4,C3_8,G3_8,C4_8,G3_8,....
...
G2_8, G3_8, G2_8, D3_8, G2_8, G3_8, D4_8,G3_8,        B2_8, B3_8, B2_8, F3_8, B2_8
E2_8, E3_8, E2_8, B2_8, E2_8, E3_8, B3_8,E3_8,        C3_8, C4_8, C3_8, G3_8, C3_8
G2_8, G3_8, G2_8, D3_8, G2_8, G3_8, D4_8,G3_8,...
...
B2_8, B3_8, B2_8, F3_8, B2_8, B3_8, E4_8, B3_8,       E2_8, E3_8, E2_8, B2_8, E2_8
C3_8, C4_8, C3_8, G3_8, C3_8, C4_8, G3_8,C4_8,        G2_1,...
G2_1,                                                 R1];


%% Normalizing Arrays
Song = Song / max(abs(Song))
Song2 = Song2 / max(abs(Song2))
Song3 = Song3 / max(abs(Song3))
Base = Base / (max(abs(Base))*18);
Base2 = Base2 / (max(abs(Base2))*20);


SongArr =(Song+Song2+Song3+Base+Base2);
SongArr = SongArr / max(abs(SongArr));

%% Echo Effect
z3= zeros(1,900);                    % array of 900 zeros
one= [SongArr z3];                   % right padding
two= [z3 .4*SongArr];                % left padding and reduction of amplitude
SA = one + two;                      % Echo
SA = SA/ max(abs(SA));               % Further normalizing

%% Playing Song
soundsc(SA, 8000);

%% Creating .wav file
audiowrite("Ace_Office_Theme_280.wav", SA,8000)
```

6 Extension

How does using sine wave functions rather than a cosine wave functions affect the sound output? Despite them being mathematically different, the sinusoidal pattern is present in both; therefore, I hypothesize there will no noticeable different in the sound output. To test this, a copy of the m function and the main script were created where all cosines are replaced with sines. Subject were randomly chosen to note if the presence of any differences. Results are discussed below.

6 subjects were randomly chosen in the Hudson building (time did not allow for a larger sample) and surveyed whether they notice any difference between the two versions of the song. 3 out of the 6 subjects did not notice any difference; 2 out of 6 mentioned noticing a very slight difference but nothing they could pinpoint; and one subject gave a detailed analysis in the difference of notes between the two. While the sample size is small, 50% of the subjects noticed a difference. I also noticed that the amplitude is higher and the echo is more pronounced in the sine version. The hypothesis is thus rejected.

Figure 1 and figure 2 display plots using the plot function in MATLAB with the SA array passed as an argument for both versions. The amplitude in figure 2 is visibly larger, supporting the conclusion that the amplitude is higher in the sound output using sine.

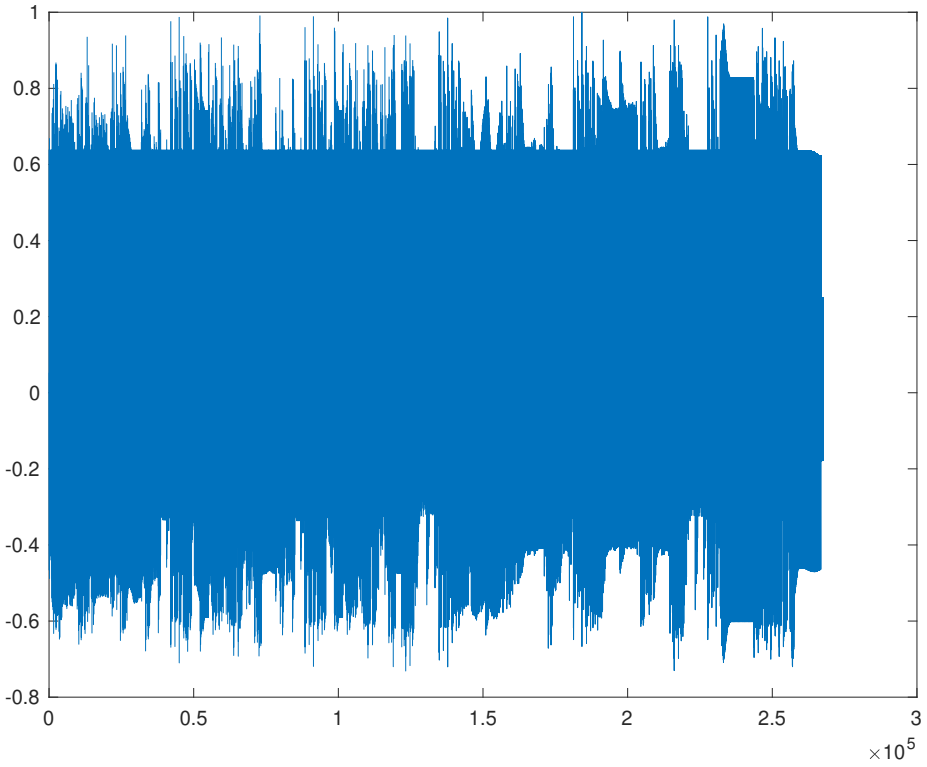

Figure 1: Plot of SA array with cosine used

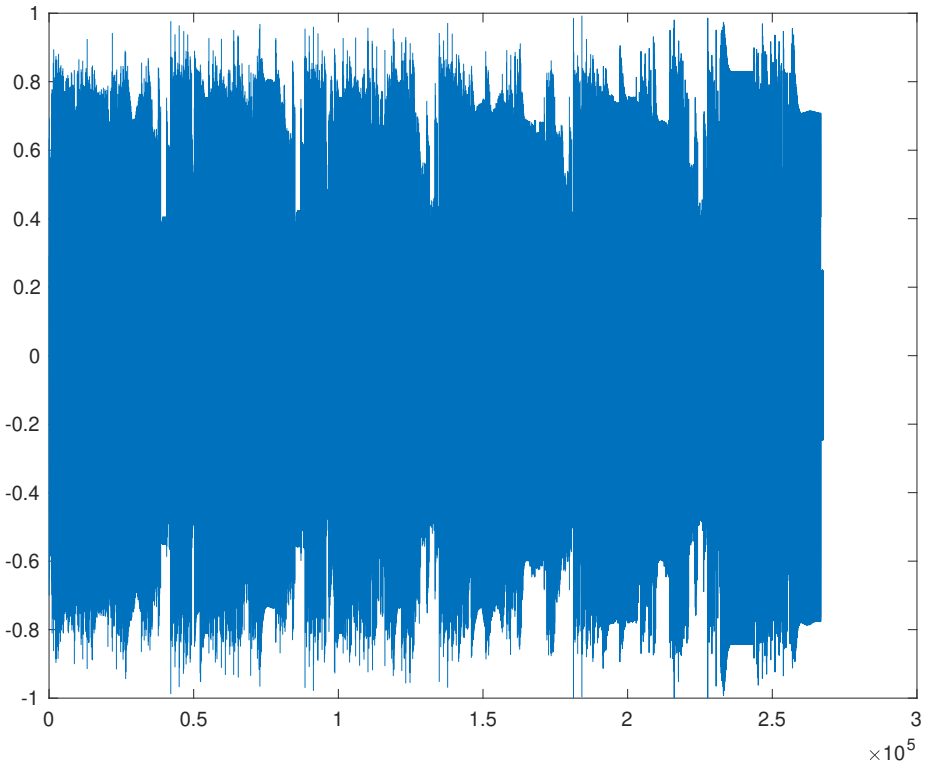

Figure 2: Plot of SA array with sine used