Signals and system

Project

Automatic Music Composition

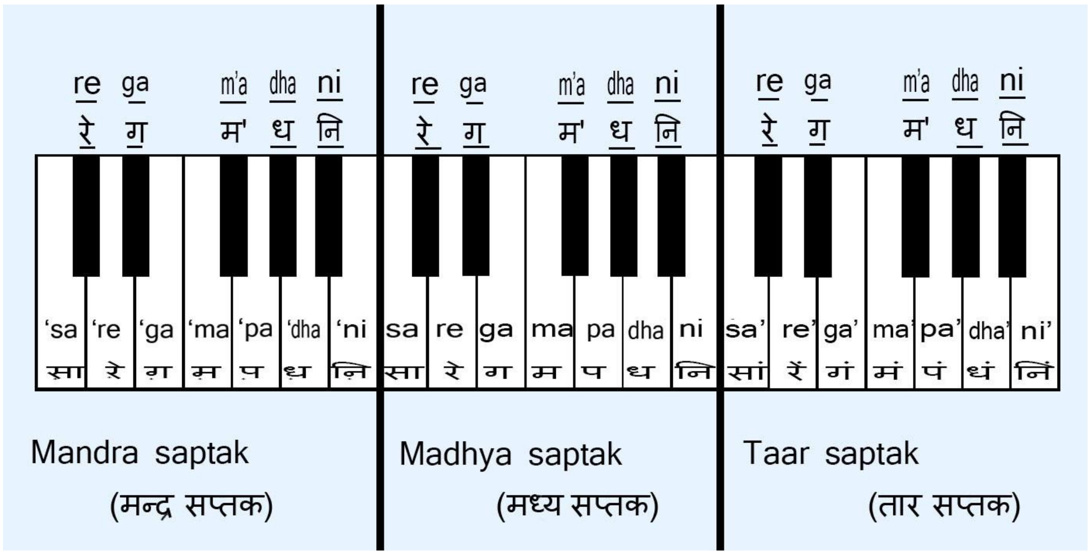
Synopsis

This program will generate an automatic random music melody using harmonics of the sine waveform. A note is generated by playing a particular frequency. Playing a series of different note frequencies, notes are generated one after another in rhythm, this is how music is generated. Variety and repetition are mixed to produce good rhythm music.

Scilab will generate the desired output and then the amplifier will amplify it. The final output is heard through the speakers. The quality of the notes depends on the number of harmonics and the change of envelope. Different types of instruments have distinct numbers of harmonics and envelopes.

Phase modulation of fundamental sine waves will generate a string or bell like sound. Also, for thinner sound, we can add harmonics to the fundamental sine wave.

The generated sound depends upon the envelope. If the envelope is sustained then we get wind type of instrument; if it decays fast then we get string type of instrument, etc. Envelope has 4 main phases:- Attack, Decay, Sustain, Release. The figure below represents various phases of the envelope.


re ga m'a dha ni re ga m'a dha ni re ga m'a dha ni
रे ग म' ध नि रे ग म' ध नि रे ग म' ध नि
'sa 're 'ga 'ma 'pa 'dha 'ni sa re ga ma pa dha ni sa' re' ga' ma' pa' dha' ni'
सा रे ग म प ध नि सा रे ग म प ध नि सां रें गं मं पं धं निं
Mandra saptak
(मन्द्र सप्तक)
Madhya saptak
(मध्य सप्तक)
Taar saptak
(तार सप्तक)

3 saptaks / Octaves can be represented as seen above.

We made a function named tone and defined 3 variables s, n and d to produce a tone. ‘s’ denotes saptak (1,2,3). ‘n’ denotes different tones like n = 0 means Sa, 1 means Komal Re, 2 means Shuddh Re etc. d denotes duration in second.

The ‘s’ and ‘n’ are generated by our program as required by the user i.e it depends on the scale and instrument

Frequency of mandra saptak “sa” is fixed and equal to 130.813 Hz. We multiplied it by N(Nodes) to get the particular frequency of the input parameter.

Different equations of fundamental sine waves represents a different type of musical instrument.

For example,

```
S=[sin(sin(2*%pi*f1*t) + a.*sin(2*%pi*f1*t) + sin(sin(8*%pi*(f1)*t)))];
```

is for bell like sound

```
S=2*(sin(2*%pi*f1*t)+sin(4*%pi*f1*t)/2+sin(3*%pi*f1*t)/2);
```

for Triumph like sound

```
S1=sin(2*%pi*f1*t)+sin(4*%pi*f1*t)/2+sin(3*%pi*f1*t)/2;
```

This equation is used to generate a Synth Wave

The program will prompt users to generate Happy or Sad type music or User given value of Tempo (beats per minute). For varieties, Users can select a single Scale out of multiple scales i.e C,C minor, D,E etc .

On Prompting happy or sad, the program will automatically choose high or low tempo and scale respectively.User also gets freedom to Choose the scale, tempo and instrument.

Techniques used

We have used 2 functions to implement the main objective: 1) tone and 2) composition

1) Functions for tone (Strings and bell):-

In this function, we have 3 parameters (s, n and d). The function noteG is used to call functions and store the values. s denotes saptak (1,2 or 3). n ranges from 0 to 11. d denotes duration in second. We have to change frequency for every saptak and rag. For that we calculate N. Which further multiplies with our main frequency f (130.815 Hz) and changes it as we require.

```
if s == 1
  N = 2^(n/12);
elseif s == 2
      N = 2*2^(n/12);
  else N = 4*2^(n/12);
```

After that, we generate a variable named ‘t’ for sampling purpose and for managing our time duration d.

t = 0:1/22050:d ; (22,050 is default sampling rate of scilab)

We have asked the user to input tempo (Beats per minute). From that tempo, we found d1(duration of bits). After that, we calculate d2 (double duration), d3(3*d1/2), d4 (4d1).

We generate envelope from the help of some internal functions like linspace, round, etc.

After we generate a note by putting the obtained value of s,f1,t in sine waveform.

```
 S=2*(sin(2*%pi*f1*t)+sin(4*%pi*f1*t)/2+sin(3*%pi*f1*t)/2); //Triumph like
```

(Phase modulated signal)

We multiply it with **s** to generate the final tone.

```
tone = 2*a.*S;
```

2) Function composition:-

The main aim is to generate an automated tune as required by the user. The methods required to accomplish this task is implemented in the Composition function. The user is required to choose an instrument between a string, bell and triumph.

Also the tempo is provided by the user having units of notes or beats per minute. d1 is the reference duration of the note. It decides the tempo of the note and vice versa. As mentioned above, the user will provide the required tempo. Then d1 will be calculated by the formula *d1 = 60/Tempo* . On increasing d1, the speed of the music will increase. Then we have generated other notes of duration d0, d2, d3, d4. They are generated by multiplying the reference duration by integer multiple of 2, i.e. d1/2, 2*d1, d1*3/2, 4*d1 etc. in order to keep the rhythm proper.

Silence is quite often used in music composition for a required amount of duration. inter is used to generate the silence of d1 seconds. 22050 is the sampling rate of the scilab. Default rate at which the samples are sent is different depending on the programming languages. C_scale, D_scale, E_scale, etc. are written using the general knowledge of music scales as required in our program.

Then depending on the instrument chosen by the user, the execution of the 'if' loop will take place. First loop for string instrument whereas second loop for the bell instrument chosen. Total time of composition is same. temp is taken as an empty array initially. j varies from 1 to 2. Hence the melody will repeat twice.

Here y = grand(1,"uin",1,8) means in variable y, any random tone is taken of d1 duration from C_scale. The numbers 1 and 8 are the pointers in the C_scale. Then stored under variable 'random'.

random = noteF(1,C_scale(Y),d2);

Harmonics and an envelope of string instrument are stored under note F so note F is used. Then composition will store random and temp will store composition. In short, the variable ‘composition’ will get updated and a melody will be generated at the end of the loop. In the second loop, the loop will run only two times because the duration d2 is taken. The duration of all the measures should be the same. Hence by taking d2 the duration gets doubled and the number of notes gets halved.

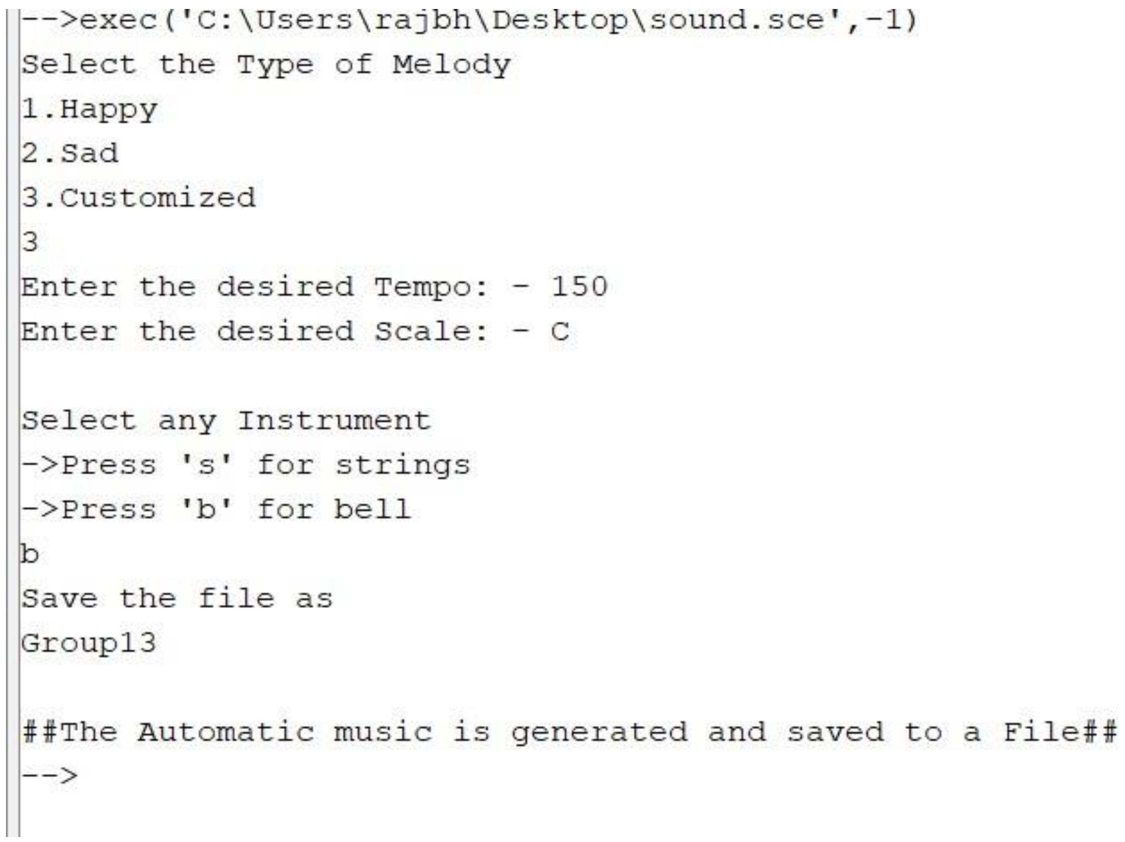
In the second loop, again an empty array temp2 is taken. A random2 variable will store 4 notes of d2 duration from C_scale and Note F. In a similar way composition2 and temp2 will be updated. In the ‘final’ variable the silence produced at the starting of this function is used and combined with temp2 and then updated. At last ‘ran’ variable will produce a melody having d4 duration. melody1 and melody2 are generated by us by combining the melodies of composition and composition2 . This whole process of assigning is done to combine all the notes to generate a beautiful melody. Now this all melodies will be combined by using repetition and variety in suitable proportion to produce the music like a string instrument.

The above process is used for the input instrument ‘string’. If the user has chosen the bell instrument then the below mentioned loop will get executed. Overall the time duration and process is similar.

Music composition is done in terms of measures .The length of the measure can be any integral multiple of the reference note duration. But the duration in each measure should be the same. By putting all the measures in a phrase and combination of phrases will generate a melody. Variety and repetition are mixed such that a music having good rhythm is generated.

User interface

The images shown below are our user interface. We have written our code in Scilab. User will select the type of music he wants to listen to. We have given him 3 choices namely: Happy, Sad, Customized. We have also provided him with two instruments, strings or bell. According to the type of melody and instrument selected, our program will compose an automated music. The music generated will be saved in the file having .wav file format. The name of the file will be given by the user.

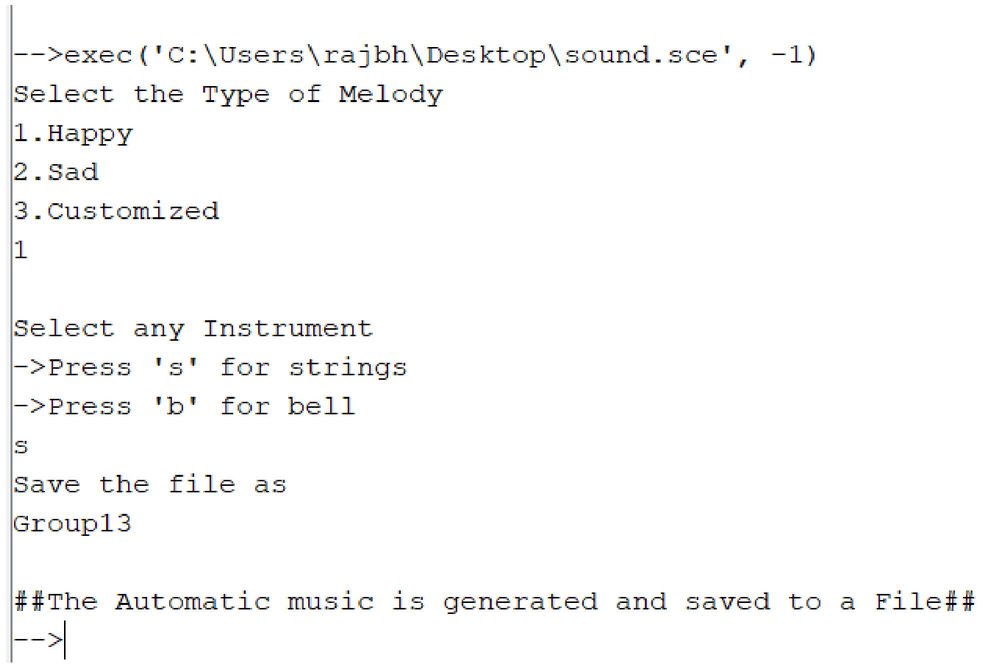


```
-->exec('C:\Users\rajbh\Desktop\sound.sce',-1)
Select the Type of Melody
1.Happy
2.Sad
3.Customized
3
Enter the desired Tempo: - 150
Enter the desired Scale: - C

Select any Instrument
->Press 's' for strings
->Press 'b' for bell
b
Save the file as
Group13

##The Automatic music is generated and saved to a File##
-->
```

For example, if the user wants to customise his melody he will enter '3'. Thus, the user is free to choose the tempo and scale. The tempo should be in the unit of beats per minute. The scales available are C major, D minor, E, etc. Thus, a customized melody will be produced.



```
-->exec('C:\Users\rajbh\Desktop\sound.sce', -1)
Select the Type of Melody
1.Happy
2.Sad
3.Customized
1

Select any Instrument
->Press 's' for strings
->Press 'b' for bell
s
Save the file as
Group13

##The Automatic music is generated and saved to a File##
-->
```

If the user doesn't want to manually select the tempo and scale. Then he can choose a happy or sad melody by pressing 1 or 2 respectively. Thus the program will automatically select the scale and tempo depending on the happy or sad melody. The tempo will be high for happy and low for sad. Thus, quickly user will get automated music.

Programs

We have implemented our code in scilab version 5.5.2. Below is the code we used and wrote.

```
funcprot(0);
clear;
function bell=noteG(s, n, d) ////// BELL
//s denotes saptak (1,2,3).
// n ranges from 0 to 11
// n = 0 means Sa, 1 means Komal Re, 2 means Shuddh Re etc.
// d denotes duration. Calibrated in seconds
 if s == 1
    N = 2^(n/12);
```

```
elseif s == 2
        N = 2*2^(n/12);
    else N = 4*2^(n/12);
end
f = 130.815;
f1 = f*N;
t = 0:1/22050:d;

T = length(t);
T1 = round(0.02*T);
T2 = round(0.04*T);
T3 = round(0.88*T);
L1 = linspace(0,1,T1);
L2 = linspace(1,1,T2);
L3 = linspace(1,0.9,T2);
L4 = linspace(0.9, 0.45,T3);
L5 = linspace(0.45,0,T1);
a = [L1 L2 L3 L4 L5 ];
A = length(a);
if T > A then
    diff = T-A;
    for i = 1:diff
        a = [a 0];
    end

elseif T < A then
        diff = A-T;
        for i = 1:diff
            t = {t 0};
        end



end
if instrument =  then
```

```
end
//S = sin((2*%pi*f1*t + a.*sin(4*%pi*f1*t) )); //Bell
//S=2*(sin(2*%pi*f1*t)+sin(4*%pi*f1*t)/2+sin(3*%pi*f1*t)/2); //Triumph like
S=[sin(sin(2*%pi*f1*t) + a.*sin(2*%pi*f1*t) + sin(sin(8*%pi*(f1)*t)))];
bell = 2*a.*S;
endfunction

function strings=noteF(s, n, d) ///// SYNTH
//s denotes saptak (1,2,3).
// n ranges from 0 to 11
// n = 0 means Sa, 1 means Komal Re, 2 means Shuddh Re etc.
// d denotes duration. Calibrated in seconds
 if s == 1
    N = 2^(n/12);
elseif s == 2
        N = 2*2^(n/12);
    else N = 4*2^(n/12);
end
f = 130.815;
f1 = f*N;
t = 0:1/22050:d;

T = length(t);
T1 = round(0.02*T);
T2 = round(0.04*T);
T3 = round(0.88*T);
L1 = linspace(0,1,T1);
L2 = linspace(1,1,T2);
L3 = linspace(1,0.9,T2);
L4 = linspace(0.9, 0.45,T3);
L5 = linspace(0.45,0,T1);
a = [L1 L2 L3 L4 L5 ];
A = length(a);
if T > A then
```

```
    diff = T-A;
    for i = 1:diff
        a = [a 0];
    end

elseif T < A then
        diff = A-T;
        for i = 1:diff
            t = {t 0};
        end



end
//S1 = sin((2*%pi*f1*t + a.*sin(4*%pi*f1*t) ));
S1=sin(2*%pi*f1*t)+sin(4*%pi*f1*t)/2+sin(3*%pi*f1*t)/2; //SYNTH
strings = 10*a.*S1;

endfunction

function COMPOSITION=composed(instrument, d1,Scale)
d0 = (d1)/2;
d2 = 2*d1;
d3 = d1*(3/2);
d4 = 4*d1;
inter=0*[0:1/22050:d1];

if Scale == "C" then
    scale=[ 0,2,4,5,7,9,11,12];
elseif Scale == "D" then
    scale=[2,4,6,7,9,11,13,14];
elseif Scale == "E" then
    scale=[4,6,8,9,11,13,15,16];
elseif Scale == "Dm" then
    scale=[2,4,5,7,9,10,12,14]
```

```
end


if instrument == "s" then
        temp=[];
        for j = 1:2
            for i = 0:3
        y = grand(1,"uin",1,8);
        random = noteF(2,scale(y),d1);
        composition = [temp random];
        temp = composition;
    end
    for i = 0:1
        Y = grand(1,"uin",4,8);
        random = noteF(1,scale(Y),d2);
        composition = [temp random];
        temp = composition;
    end
        end

temp2=[];
    for j = 1:1
        for i = 0:3
        Y1 = grand(1,"uin",1,8);
        random2  =  noteF(2,scale(Y1),d1);
        composition2  =  [temp2  random2];
        temp2 = composition2;
        end
    final = [inter temp2 ];
    temp2 = final;
end
num = grand(1,"uin",1,8);
ran = noteF(2,scale(num),d4)
```

```
melody1 = [composition composition ];
melody2 = [composition2 composition2 ran ];
music = [melody1 melody2 melody1];

   elseif instrument == "b" then
   temp=[];
for j = 1:2
   for i = 0:3
      y = grand(1,"uin",1,8);
      random = noteG(2,scale(y),d1);
      composition = [temp random];
      temp = composition;
   end
   for i = 0:1
      Y = grand(1,"uin",4,8);
      random = noteG(1,scale(Y),d2);
      composition = [temp random];
      temp = composition;
   end
end

temp2=[];
for j = 1:1
   for i = 0:3
      Y1 = grand(1,"uin",1,8);
      random2 = noteG(2,scale(Y1),d1);
      composition2 = [temp2 random2];
      temp2 = composition2;
   end
   final = [ temp2 random2];
   temp2 = final;
end
num = grand(1,"uin",1,8);
ran = noteG(2,scale(num),d4)
```

```
melody1 = [composition composition ];
melody2 = [composition2 composition2 ran ];
music = [melody1 melody2 melody1 ];
end

COMPOSITION = music;

endfunction

////// MAIN FUNCTION///////

printf("Select the Type of Melody\n" + "1.Happy\n" + "2.Sad\n" +
"3.Customized\n");
genre = input("","string");

if genre == "1" then
    x = 180;
    Scale = "C"
elseif genre == "2" then
    x = 120;
    Scale = "Dm"
elseif genre == "3" then
    x=input("Enter the desired Tempo: - ");
    Scale = input("Enter the desired Scale: - ","string");
end

d1 =60/x;       /// Tempo

printf("\n");
printf("Select any Instrument\n" + "->Press "s" for strings\n" + "->Press "b" for
bell\n");
Instrument = input("","string");

FINAL = composed(Instrument,d1,Scale)
```

```
printf("Save the file as\n");
name = input("","string");
wavwrite(FINAL,name);
printf("\n");
printf("##The Automatic music is generated and saved to a File##");
//sound(FINAL);
```

Results

In the results, User will get a saved file which has been generated by the above code. If the user selected low tempo or sad type then the duration of the music file would be longer because the composition will generate music of low speed. Similarly, if User selected a happy type or High tempo then the duration of the file would be smaller because composition would be at slow speed. Selecting the type of Instrument will not make any difference other than difference in sound. However, Selecting different Scales will choose different notes of different emotions to alter the User's ideology of music theory. The code will generate a music melody as per input. The notes of the music produced will be displayed along with the composition of music. This music will be saved in a .wav file format.

Conclusion

Overall, This whole project of AUTOMATIC MUSIC COMPOSITION is about understanding the concept of music and how signals take a huge part in the role of making music. First of all, We can understand how a particular note is produced and similarly produce a series of notes resulting in music. All of us, actually did research and found out how practically the sound is produced through a sine wave. We all did learn much through this project, processed our brains to work on the Algorithms to generate automatic randomized composition of music. However, there was a limitation that we were not able to play chords/bass notes along with the melodies. Overall, everything was great, each and every one of us enjoyed doing the project and are fully satisfied with the output.