Introduction:

Hungarian dance number five instills very distinct emotions when played right on piano. The aim of trying to replicate this particular song on MATLAB was to see how realistic the result would be and if human hear can fully enjoy electronically produced piano music. To make the notes sound similar to a real piano four signal processing effects were used and will be explored in this lab: exponential decay, harmonics, volume variations(ADSR) and echo.

Procedure:

The MATLAB code starts by introducing fs frequency, duration variables and note frequencies. It continues with constructing six note vectors, three per one hand on piano and two duration vectors, one per one hand on piano. The lower1and lower vectors of each hand are written for notes played at the same time, and are created by using zeros vector with the length of the original notes(piano1notes and piano2notes) and adding those additional notes to the specific locations. Code then follows by adjusting piano1 to F major scale, considering natural signs of scale cancellation into account by their index(89). A similar scaling is conducted on piano2. Then code introduces some of the vectors and variables of effects and in a for loop uses them to adjust the sound output. A for loop is introduced that loops through index numbers starting from one to the length of piano’s durations. As each note has a unique duration variable defined, this loop allows each note to be converted into signal and edited separately. Inside the loop, t1 is defined as indices from 0 to note’s duration (1-base period) and sliced for the base period(1/fs). Exponential decay effect was implemented by introducing x variable inside the for loop that takes negative exponential of t1 time vector. Later the amplitude of the cosine signals are multiplied by x variable to implement this trend of exponential decaying over time. ADSR effect was implemented by defining ratios of time separation for each four ADSR segments and introducing those intervals as ati,dti,sti,rli. The output’s amplitude is later adjusted for those intervals using slope equation and a sustain amplitude level of 0.70. Those amplitude intervals are later put together and the resulting vector is multiplied by the cosine signals to give ADSR behavior to the output signal. For each note vector two harmonic frequency is added with the amplitude of 0.50 and 0.25 respectively. This is done by creating additional two vectors per note vector with 2 times and 4 times frequency of the original vector and multiplying by the amplitude. Exponential decay and ADSR is also applied to the harmonic vectors. Inside the for loop notes with frequency 1 and 0 are considered with if statement to give 0 amplitude for rests(1) and empty notes(0) in the note vectors. The output signal is produced by adding each output of the for loop at the end of the last one. Echo effect is constructed after addition of all different outputs and normalization. It is constructed by shifting the original output vector 1000 indices to the right and adding it on top of itself with a smaller amplitude.

Result:

After implementing all the effects, the output sounded very realistic. It did not have the fullness of real piano yet it was still far better than the first electrical sound MATLAB was capable of producing without any effects. In addition to that no ‘popping’ or unexpected disruption was observed. By changing the variables defined for each effect like indices of shifting and amplitude for echo, the ratios of time distribution and sustain amplitude in ADSR, amplitude of harmonics in harmonic effect, and degree of exponential decay; the best combination was achieved and the addition of each element to the quality was observed.

Discussion:

ADSR shaped the output signal for each cosine with the ADSR graphs shape and therefore introduced an increase, decrease, stable and a decrease. This replicated the effect of hitting a piano button and therefore made each note sound more realistic. It also helped eliminating ‘popping’ sound due to fast jumps between amplitudes. Exponential decay gave the effect of notes going slowly silent. Harmonics made sounds similar to the real piano strings that have multiple harmonics. Echo with the amplitude of 0.20 gave the effect of pianos wooden body, which the notes slightly echo from to have a ‘full’ sound. Echo and harmonics had a great impact on making the output more realistic. ADSR had great impact eliminating indecencies in the sound. Exponential decay had the smallest effect as ADSR had a greater effect on overall amplitude which can be observed in the following(Figure-1) graph of a single note’s amplitude distribution.

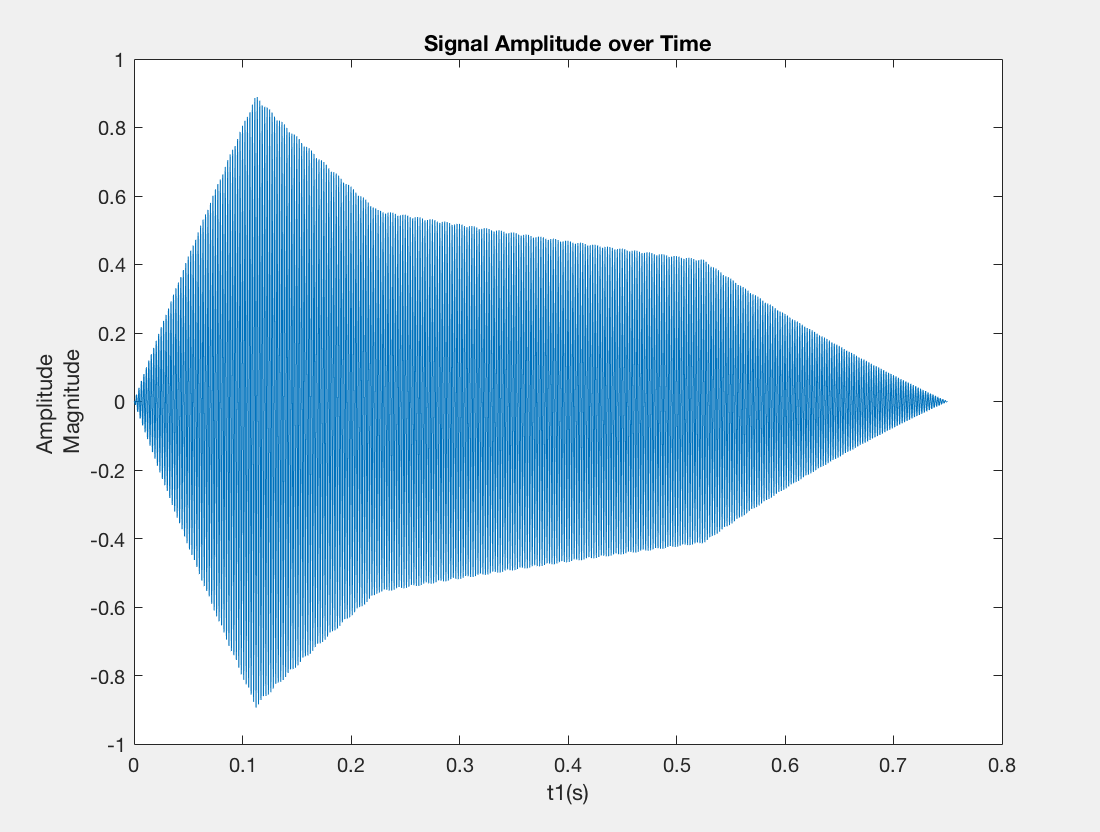


Figure-1

During the implementation of ADSR code, the problem occurred due to addition of different sized vectors. The solution for this was indexing the amplitude vector(A) into slices before adding different intervals of ADSR. Another problem was that due to the notes occurring at the same time and the normalization, the double and triple notes would sound really high volume compared to single notes. The solution was conducted by introducing counting variable that counts number of notes for given duration and adjusting the amplitude accordingly.

The next step in improving quality of songs would be adjusting harmonics specific to the notes. A quick skim through researches conducted on harmonics yielded that the distribution of harmonic sounds differentiates for each note, and as harmonics was one of the most impacting effect this can further improve the quality of the song. Instrument synthesis would be the second step to input that ‘full’ sound of the piano rather than half electronic frequency sounds. In addition to that, overlaying notes can be a viable choice as piano players tend to use that technique and a small overlay between two notes adds to the total harmony of the songs.

The endpoint is chosen as 1-1/fs as t1 starts from zero, and should have total duration of notes duration. Following that, the choice of sin and cosine does not matter due to same frequency input and the high length of the input vector as starting vector would not affect the overall distribution. Vectors of zero amplitudes are implement both for silents and empty notes. As observed in the first lab, if amplitude exceeds the magnitude of 1, the computer makes a popping sound evidencing that the limits have been exceeded.

After looking through graphs showing frequency time and amplitude time distributions of recorded piano notes, a question emerged of ‘intentional randomness’ emerged. It seems as though, no matter how accurate the sound is recorded there is some randomness in the distributions. Even if too many effects are installed to the program, it would still play the same note with exact same distribution each time and there is a chance that human ear can tell that ‘perfectness’ and will not enjoy it. The fact that there are some softwares available to nearly perfectly imitate a piano is not preferred to listening actual piano playing or even the record of actual piano playing supports this chance. This can either mean that there might be an underlaying, dismissed holistic factor that can be achieved by considering song as a whole rather than perfecting each note or that people actually enjoy same randomness. If the latter holds truth, randomness can be implemented after the perfection of each sound to slightly disturb the notes and song while making it more ‘realistic’.

MATLAB CODE:

%clearing variables & screen

clear

%frequency

fs = 8000;

%durations

w = 1;

h = 0.5;

q = 0.25;

e = 0.125;

res = 0.01;

clear piano2notes;

%notes

lD = 220\*2^((13-8)/12); %D4

lDs = 220\*2^((14-8)/12);

lE = 220\*2^((15-8)/12);

lF = 220\*2^((16-8)/12);

lFs = 220\*2^((17-8)/12);

lG = 220\*2^((18-8)/12);

lGs = 220\*2^((19-8)/12);

A = 220\*2^((20-8)/12);

As = 220\*2^((21-8)/12);

B = 220\*2^((22-8)/12);

C = 220\*2^((23-8)/12);

Cs = 220\*2^((24-8)/12);

D = 220\*2^((25-8)/12);

Ds = 220\*2^((26-8)/12);

E = 220\*2^((27-8)/12);

F = 220\*2^((28-8)/12);

Fs = 220\*2^((29-8)/12);

G = 220\*2^((30-8)/12);

Gs = 220\*2^((31-8)/12);

hA = 220\*2^((32-8)/12);

hAs = 220\*2^((33-8)/12);

hB = 220\*2^((34-8)/12);

hBs = 220\*2^((35-8)/12);

listNotes = [lD lDs lE lF lFs lG lGs ...

A As B C Cs D Ds E F Fs G Gs ...

hA hAs hB hBs];

%starts with lE

%notes starts from A2 to E4

G2 = 220\*2^(-14/12);

G2s = 220\*2^(-13/12);

A2 = 220\*2^(-12/12);

A2s = 220\*2^(-11/12);

B2 = 220\*2^(-10/12);

C3 = 220\*2^(-9/12);

C3s = 220\*2^(-8/12);

D3 = 220\*2^(-7/12);

D3s = 220\*2^(-6/12);

E3 = 220\*2^(-5/12);

F3 = 220\*2^(-4/12);

F3s = 220\*2^(-3/12);

G3 = 220\*2^(-2/12);

G3s = 220\*2^(-1/12);

A3 = 220\*2^(-0/12);

A3s = 220\*2^(1/12);

B3 = 220\*2^(2/12);

C4 = 220\*2^(3/12);

C4s = 220\*2^(4/12);

D4 = 220\*2^(5/12);

D4s = 220\*2^(6/12);

E4 = 220\*2^(7/12);

F4 = 220\*2^(8/12);

F4s = 220\*2^(9/12);

G4 = 220\*2^(10/12);

G4s = 220\*2^(11/12);

A4 = 220\*2^(12/12);

A4s = 220\*2^(13/12);

%%%pauses only 1/8 is used in sheet

p = 1; %frequency doesnt matter as amp is 0

% ADD SHIFT FACTOR SIGNATURE KEYS

piano1notes = [A D F D Cs D E ...

D B C D A ... %1

lG lF lE A ...

lD ...

A D F ...

hA F ...

E F G ...

F ...

B C D A B C ... %2

lG lF ...

lG lF lE A...

lD lD p ...

hA hA ...

hB hA ...

p G Fs G ...

hA G Fs hA G p... %3

G G...

hA G ...

p F E F...

G F E G F p...

E E ...

G F E ...

E D Cs D ... % 4

E D Cs E D p ...

hA p p A...

B Cs...

D D Cs D...

E D Cs E D p...

hB hA ...

D p];

%%%for simultaneous notes

piano1lower = zeros(1,size(piano1notes,2));

piano1lower2 = zeros(1,size(piano1notes,2));

piano1lower(42) = C;

piano1lower(43) = C;

piano1lower(44) = D;

piano1lower(45) = C;

piano1lower(56) = B;

piano1lower(57) = B;

piano1lower(58) = C;

piano1lower(59) = B;

piano1lower(70) = lG;

piano1lower(71) = lG;

piano1lower(72) = B;

piano1lower(73) = A;

piano1lower(74) = lG;

piano1lower(75) = lG;

piano1lower(76) = B;

piano1lower(103) = A;

piano1lower(101) = G;

piano1lower(102) = E;

piano1lower2(101) = D;

piano1lower2(102) = Cs;

piano1lower2(103) = lF;

%Piano1durations

piano1dur = [h+q q...

h+q q...

h+q e e...

w...

h+q e e...

w...%1

q q q+e e...

w...

h+q e e...

h+q q...

h+q e e...

w...

e e q e e q...%2

h+q q...

q q q+e e...

h q q...

h h...

h+q q...

q h e e...

e e e e q q... %3

h h...

h+q q...

q h e e ...

e e e e q q ...

h h ...

q h q ...

q h e e ...%4

e e e e q q...

q q q+e e...

h h...

q h e e...

e e e e q q...

h h...

h+q q];

%Piano2notes

piano2notes= [D3 A3 F3 A3... %a

D3 A3 F3 A3...

D3 B3 G3 B3... D3 B3 G3 B3

D3 A3 F3 A3...

D3 B3 G3 B3...

D3 A3 F3 A3...%1

C3s A3 E3 A3...

D3 A3 F3 A3...

D3 A3 F3 A3...

D3 A3 F3 A3...

D3 B3 G3 B3...

D3 A3 F3 A3...

B3 A3...%2

G3 F3...

G3...

D3 A3 D3 p...

Fs D4 A2 D4...

Fs D4 A2 D4...

G3 D4 B3 D4... A2 E4 C4 E4

G3 D4 B3 D4...%3

D3 C4 G3 C4... F3 D4 A2 D4

D3 C4 G3 C4...

F3 C4 A3 C4...

F3 C4 A3 C4...

C3s A3 D3 A3...

C3s A3 D3 A3...

B2 F3 D3 F3...%4

B2 F3 D3 F3...

E3 p p A2...

B2 C3s...

D3 A3 p...

A3 A3 p...

G2 A2...

D3 p];

%Addition of notes at the same time

piano2lower = zeros(1,size(piano2notes,2));

piano2lower2 = zeros(1,size(piano2notes,2));

piano2lower(49) = D3;

piano2lower(50) = C3;

piano2lower(51) = B2;

piano2lower(52) = A2;

piano2lower(53) = A2;

piano2lower(55) = F3;

piano2lower(106) = A2;

piano2lower(113) = F3;

piano2lower(115) = G3;

piano2lower(116) = F3;

piano2lower2(113) = D3;

piano2lower2(115) = D3;

piano2lower2(116) = D3;

%piano2durations

piano2dur = [q q q q...

q q q q...

q q q q...

q q q q...

q q q q...

q q q q...%1

q q q q...

q q q q...

q q q q...

q q q q...

q q q q...

q q q q...

h h... %2

h h...

w...

q q q q...

q q q q...

q q q q...

q q q q...

q q q q...%3

q q q q...

q q q q...

q q q q...

q q q q...

q q q q...

q q q q...

q q q q... %4

q q q q...

q q q+e e...

h h...

q h q...

h q q...

h h...

h+q q];

%%F major scale for piano1 and considering the natural signs.

for c=1:size(piano1notes,2)

if ((piano1notes(c))==(B) && c~=89)

piano1notes(c)=As;

end

if (piano1lower(c))==(B)

piano1lower(c)=As;

end

if (piano1lower2(c))==(B)

piano1lower2(c)=As;

end

end

%B3 to A3s major scale with naturals considered

for o=1:size(piano2notes,2)

if (piano2notes(o) == B3 && o~=110)

piano2notes(o) = A3s;

end

if (piano2lower(o) == B3)

piano2lower(o) = A3s;

end

if (piano2lower2(o) == B3)

piano2lower2(o) = A3s;

end

end

%initalization of note vectors

y1 = [];

y1h1 = [];

y1h2 = [];

yl1 = [];

yl1h1 = [];

yl1h2 = [];

yl2 = [];

yl2h1 = [];

yl2h2 = [];

y2 = [];

y2h1 = [];

y2h2 = [];

y2l1 = [];

y2l1h1 = [];

y2l1h2 = [];

y2l2 = [];

y2l2h1 = [];

y2l2h2 = [];

%harmonic powers

hp1 = 0.50;

hp2 = 0.25;

%ADSR limit ratios

at = 0.15;

dc = 0.30;

st = 0.70;

rl = 1.00;

sustainAmp = 0.70;

for i = 1:(size(piano1dur,2))

clear A;

u = (piano1dur(i)-piano1dur(i)/fs);

t1 = 0:(1/fs):u;

x = exp((-1)\*t1);

% counter for dividing amplitudes of same time notes

%Explained in lab

counter = 1;

maxt = max(t1);

ati = t1(t1<at\*u);

dci = t1(t1>=at\*u & t1<dc\*u);

sti = t1(t1>=dc\*u & t1<st\*u);

rli = t1(t1>=st.\*u);

ATV = [];

DCV = [];

STV = [];

RLV = [];

ATV = [(1/(max(ati)-min(ati))).\*(ati-0)];

DCV = [1.00+((sustainAmp-1.00)./(max(dci)-max(ati)).\*(dci-max(ati)))];

STV = [ones(1,size(sti,2)).\*sustainAmp];

RLV = [0.7+((0-sustainAmp)./(max(t1)-max(sti))).\*(rli-max(sti))];

atl = size(ATV,2);

dcl = size(DCV,2);

stl = size(STV,2);

rlv = size(RLV,2);

A(1:atl) = ATV;

A((atl+1):(atl+dcl)) = DCV;

A((atl+dcl+1):(atl+dcl+stl)) = STV;

A((atl+dcl+stl+1):(atl+dcl+stl+rlv)) = RLV;

if piano1notes(i)==1 %silents

y1 = [y1,0\*t1];

y1h1 = [y1h1,0\*t1];

y1h2 = [y1h2,0\*t1];

else

y1 = [y1,A.\*x.\*cos(2\*pi\*piano1notes(i)\*t1)];

y1h1 = [y1h1,hp1\*A.\*x.\*cos(4\*pi\*piano1notes(i)\*t1)];

y1h2 = [y1h2,hp2\*A.\*x.\*cos(6\*pi\*piano1notes(i)\*t1)];

end

if piano1lower(i)==0 || piano1lower(i)==0

yl1 = [yl1,0\*t1];

yl1h1 = [yl1h1,0\*t1];

yl1h2 = [yl1h2,0\*t1];

else

counter = counter+1;

yl1 = [yl1,A.\*x.\*cos(2\*pi\*piano1lower(i)\*t1)];

yl1h1 = [yl1h1,hp1\*A.\*x.\*cos(4\*pi\*piano1lower(i)\*t1)];

yl1h2 = [yl1h2,hp2\*A.\*x.\*cos(6\*pi\*piano1lower(i)\*t1)];

end

if piano1lower2(i)==1 || piano1lower2(i)==0 %silents and empty notes

yl2 = [yl2,0\*t1];

yl2h1 = [yl2h1,0\*t1];

yl2h2 = [yl2h2,0\*t1];

else

counter = counter+1;

yl2 = [yl2,A.\*x.\*cos(2\*pi\*piano1lower2(i)\*t1)];

yl2h1 = [yl2h1,hp1\*A.\*x.\*cos(4\*pi\*piano1lower2(i)\*t1)];

yl2h2 = [yl2h2,hp2\*A.\*x.\*cos(6\*pi\*piano1lower2(i)\*t1)];

end

counter = counter\*2;

y1(i)=y1(i)./counter;

y1h1(i)=y1h1(i)./counter;

y1h2(i)=y1h2(i)./counter;

yl1(i)=yl1(i)./counter;

yl1h1(i)=yl1h1(i)./counter;

yl1h2(i)=yl1h2(i)./counter;

yl2(i)=yl2(i)./counter;

yl2h1(i)=yl2h1(i)./counter;

yl2h2(i)=yl2h2(i)./counter;

if i==1

plot(t1,y1)

end

end

for i = 1:(size(piano2dur,2))

clear A maxt;

u = (piano2dur(i)-piano2dur(i)/fs);

t1 = 0:(1/fs):u;

x = exp((-1)\*t1);

%if i==1

% g = cos(2\*pi\*notes(i)\*t1);

% t = t1;

% g = g/max(g(:));

%end

% tn = t1/max(t1(:));

counter = 1;

maxt = max(t1);

ati = t1(t1<at\*u);

dci = t1(t1>=at\*u & t1<dc\*u);

sti = t1(t1>=dc\*u & t1<st\*u);

rli = t1(t1>=st.\*u);

ATV = [];

DCV = [];

STV = [];

RLV = [];

ATV = [(1/(max(ati)-min(ati))).\*(ati-1/fs)];

DCV = [1.00+((sustainAmp-1.00)./(max(dci)-max(ati)).\*(dci-max(ati)))];

STV = [ones(1,size(sti,2)).\*sustainAmp];

RLV = [0.7+((0-sustainAmp)./(max(t1)-max(sti))).\*(rli-max(sti))];

atl = size(ATV,2);

dcl = size(DCV,2);

stl = size(STV,2);

rlv = size(RLV,2);

A(1:atl) = ATV;

A((atl+1):(atl+dcl)) = DCV;

A((atl+dcl+1):(atl+dcl+stl)) = STV;

A((atl+dcl+stl+1):(atl+dcl+stl+rlv)) = RLV;

if piano2notes(i)==1

y2 = [y2,0\*t1];

y2h1 = [y2h1,0\*t1];

y2h2 = [y2h2,0\*t1];

else

y2 = [y2,A.\*x.\*cos(2\*pi\*piano2notes(i)\*t1)];

y2h1 = [y2h1,hp1\*A.\*x.\*cos(4\*pi\*piano2notes(i)\*t1)];

y2h2 = [y2h2,hp2\*A.\*x.\*cos(6\*pi\*piano2notes(i)\*t1)];

end

if piano2lower(i)==0 || piano2lower(i)==0

y2l1 = [y2l1,0\*t1];

y2l1h1 = [y2l1h1,0\*t1];

y2l1h2 = [y2l1h2,0\*t1];

else

counter = counter+1;

y2l1 = [y2l1,A.\*x.\*cos(2\*pi\*piano2lower(i)\*t1)];

y2l1h1 = [y2l1h1,hp1\*A.\*x.\*cos(4\*pi\*piano2lower(i)\*t1)];

y2l1h2 = [y2l1h2,hp2\*A.\*x.\*cos(6\*pi\*piano2lower(i)\*t1)];

end

if piano2lower2(i)==1 || piano2lower2(i)==0

y2l2 = [y2l2,0\*t1];

y2l2h1 = [y2l2h1,0\*t1];

y2l2h2 = [y2l2h2,0\*t1];

else

counter = counter+1;

y2l2 = [y2l2,A.\*x.\*cos(2\*pi\*piano2lower2(i)\*t1)];

y2l2h1 = [y2l2h1,hp1\*A.\*x.\*cos(4\*pi\*piano2lower2(i)\*t1)];

y2l2h2 = [y2l2h2,hp2\*A.\*x.\*cos(6\*pi\*piano2lower2(i)\*t1)];

end

counter = counter\*2;

y2(i)=y2(i)./counter;

y2h1(i)=y2h1(i)./counter;

y2h2(i)=y2h2(i)./counter;

y2l1(i)=y2l1(i)./counter;

y2l1h1(i)=y2l1h1(i)./counter;

y2l1h2(i)=y2l1h1(i)./counter;

y2l2(i)=y2l2(i)./counter;

y2l2h1(i)=y2l2h1(i)./counter;

y2l2h2(i)=y2l2h2(i)./counter;

end

y1tot = y1 + y1h1+ y1h2 + yl1 + yl1h1 + yl1h2 + yl2 + yl2h1 + yl2h2;

y2tot = y2 + y2h1+ y2h2 + y2l1 + y2l1h1 + y2l1h2 + y2l2 + y2l2h1 + y2l2h2;

y = (y2tot + y1tot)./2;

%y = y./max(y(:));

yecho(1:1000) = zeros(1,1000);

ylen = size(y,2);

yecho(1001:1000+ylen)=y; %echo

y(ylen+1:1000+ylen)=zeros(1,1000);

yf = y+yecho.\*0.2;

yf = yf./max(yf(:)); %normalization

sound(yf,fs)

filename = 'Yagcioglu\_HungarianDance.wav';

audiowrite(filename,yf,fs);