```
% Quy Chau
% May 2017
% Objective: capture small sample of instrument and
% generate FM synthesis parameters to emulate instrument
% Function generates parameters for a simple two-oscillator (carrier
% and modulator) synth, such as Ableton's operator or my operator.m
% Input is wav file of sample
% output is paramters A, I, fc, fm, pc, pm, wc, wm, and T

function[fc, fm, A, I, pc, pm, wc, wm, T, fs] = generator(wavfile)


%SECTION I: CREATE AMPLITUDE AND MODULATOR INDEX ENVELOPES

%Read and plot waveform

[y, fs] = audioread(wavfile); %read in file for analysis

figure(1)
plot(y)
title('Input wav file');
xlabel('time (samples)');ylabel('Amplitude');

amax = max(y);
Amax = amax(1);


%Find attack time
maxt = 0;
mint = 0;
i = round(0.02*length(y));
bool = true;
bool2 = true;
while and(i < length(y),bool)
    if and(abs(y(i)) >= 0.05*Amax, bool2)
        mint = i;
        bool2 = false;
    end
    if (abs(y(i)) >= 0.9*Amax)
        maxt = i;
        bool = false;
    end
    i = i + 1;
end
ta = (maxt - mint) / length(y);


%Find rest of amplitude envelope through least squares

ya = y(maxt:end);

Na = 100;
```

```
yamp = zeros(1,Na);
step = floor(length(ya)/Na);
tk = linspace(0,1,Na);
for i = 1:Na
    maxy = max(ya((i-1)*step+1:i*step));
    yamp(i) = maxy(1);
end

pdelay = linspace(0,0.8,9); %Arrays of guesses
psustain = linspace(0,0.8,9);
asustain = linspace(0,1*Amax,11);

a0 = 0.0*Amax; %Default parameters
am = 1*Amax;
s0 = 0.2*Amax;
td = 0.2;
ts = 0.3;
err = inf;

for j = 1:9
    for m = 1:9
        for p = 1:11

            tdj = pdelay(j);
            tsm = psustain(m);
            s0p = asustain(p);

            la = round(ta*length(tk));
            ld = round(tdj*length(tk));
            ls = round(tsm*length(tk));
            lt = length(tk) - la - ld - ls;

            attack = linspace(a0, am, la);
            decay = ((am-s0p)*exp(-(1:ld)*5/ld)+s0p); %exponential
 decay
            sustain = linspace(s0p, s0p, ls);
            release = ((s0p)*exp(-(1:lt)*5/lt));

            A = [attack decay sustain release];

            if (length(A) == length(yamp)) %if time matches up
                if (norm(A - yamp) < err) %if least error so far
                    td = tdj;
                    ts = tsm;
                    s0 = s0p;
                    err = norm(A - yamp);
                end
            end
        end
    end
end

%Time vector, total time
dt = 1/fs;
```

```
t = 0:dt:(length(y)*dt)-dt;
T = t(length(t));

la = round(ta*length(t));
ld = round(td*length(t));
ls = round(ts*length(t));
lt = length(t) - la - ld - ls;

attack = linspace(a0, am, la); %linear
decay = ((am-s0)*exp(-(1:ld)*5/ld)+s0); %exponential decay
sustain = linspace(s0, s0, ls);
release = ((s0)*exp(-(1:lt)*5/lt));

A = [attack decay sustain release];

%Default parameters for simple FM model
pc = 0;
pm = pi/2;
wc = 'sin';
wm = 'cos';


% Find index of modulation envelope through similar means
% Uses bandwidth change over time to calculate desired I

Ny = 20;
stepy = floor(length(y)/Ny);
prevend = 1;
Ienv = zeros(1,Ny);
for i = 1:Ny
    yshort = y(prevend:stepy*i);
    szys = size(yshort);
    if (szys(1) < szys(2))
        yshort = yshort';
    end
    prevend = stepy*i +1;

    [~, ~, maxfshort, ~] = findCM(yshort,fs, 0, 10);

    if isempty(maxfshort)
        Inorm = 0;
    else
        Inorm = maxfshort(1);
    end
    Ienv(i) = Inorm;

end

tv = linspace(0,1,Ny);
yamp2 = Ienv;

posMax = max(Ienv);
Amax2 = posMax(1);
```

```
pattack2 = linspace(0, 0.8, 9);
pdelay2 = linspace(0,0.8,9); %Arrays of guesses
psustain2 = linspace(0,0.8,9);
asustain2 = linspace(0,1*Amax2,11);

a02 = 0.0*Amax2; %Default parameters
am2 = 1*Amax2;
s02 = 0.2*Amax2;
ta2 = 0.1;
td2 = 0.2;
ts2 = 0.3;
err2 = inf;

for koala = 1:9
    for j = 1:9
        for m = 1:9
            for p = 1:11

                takoala2 = pattack2(koala);
                tdj2 = pdelay2(j);
                tsm2 = psustain2(m);
                s0p2 = asustain2(p);

                la2 = round(ta2*length(tv));
                ld2 = round(tdj2*length(tv));
                ls2 = round(tsm2*length(tv));
                lt2 = length(tv) - la2 - ld2 - ls2;

                attack2 = linspace(a02, am2, la2);
                decay2 = ((am2-s0p2)*exp(-(1:ld2)*5/
ld2)+s0p2); %exponential decay
                sustain2 = linspace(s0p2, s0p2, ls2);
                release2 = ((s0p2)*exp(-(1:lt2)*5/lt2));

                A2 = [attack2 decay2 sustain2 release2];

                if (length(A2) == length(yamp2)) %if time matches up
                    if (norm(A2 - yamp2) < err2) %if least error so
 far
                        ta2 = takoala2;
                        td2 = tdj2;
                        ts2 = tsm2;
                        s02 = s0p2;
                        err2 = norm(A2 - yamp2);
                    end
                end
            end
        end
    end
end

la2 = round(ta2*length(t));
ld2 = round(td2*length(t));
ls2 = round(ts2*length(t));
```

```
lt2 = length(t) - la2 - ld2 - ls2;

attack2 = linspace(a02, am2, la2); %linear
decay2 = ((am2-s02)*exp(-(1:ld2)*5/ld2)+s02); %exponential decay
sustain2 = linspace(s02, s02, ls2);
release2 = ((s02)*exp(-(1:lt2)*5/lt2));

A2 = [attack2 decay2 sustain2 release2];


%SECTION II: FIND CARRIER AND MODULATOR FREQUENCIES


%finds frequencies of FFT peaks
[ff, peaks, maxf, Th] = findCM(y,fs, 2, 10);

disp('FFT peak frequencies: ')
disp(ff)
disp('Tristimulus coefficients: ')
disp(Th)

%if zero peaks, look for more peaks with lower minimum peak height
iz = 0;
initialFactor = 10;
while (length(ff) <= 1 && iz < 10)
    factor = 1.1^iz;
    [ff, peaks, maxf, Th] = findCM(y,fs, 0, initialFactor*factor);
    iz = iz + 1;
end

%make sure no harmonics at f ~= 0. If so, remove them.
for ia = 1:length(ff)
    if abs(ff(ia)) <= 10
        ff(ia) = [];
    end
end


% New technique: pick fc, fm until output tristimulus values are
% closest to the original

dt = 1/fs;
t = 0:dt:(length(y)*dt)-dt;
T = t(length(t));

FC = 0;
FM = 0;
therr = 10000;
thp = [0 0 0];
ffp = [];
pp = [];
a = zeros(1, length(t));
ismuth = zeros(1, length(t));
ismuthmax = 0;
```

```
for ii = 1:length(ff)
    for jj = 1:length(ff)

        fc = ff(ii);
        fm = abs(ff(jj) - fc);

        fd = maxf - fc; %peak carrier frequency deviation, i.e.
 bandwidth

        % Use simplified exponential for modulation index
        if (fm == 0)
            Imax = 0;
        else
            Imax = 1*(fd/fm);
        end
        I = Imax.*exp(-5*t/T);

        % Use desired envelope for each test?
        %          if (fm == 0)
        %          I = Imax.*exp(-5*t/T);
        %          else
        %          I = A2/fm;
        %          end

        operator2(fc, fm, A, I, pc, pm, wc, wm, T, fs);

        [ytest, fstest] = audioread('sample_output.wav');

        [fftest, peakstest, maxftest, Thtest] = findCM(ytest,fstest,
 0, 10);

        % If timbral spread is the best found, replace all parameters
        currerr = norm(Thtest - Th);
        if currerr < therr
            therr = currerr;
            FC = fc;
            FM = fm;
            thp = Thtest;
            ffp = fftest;
            pp = peakstest;
            a = A;
            ismuth = I;
            ismuthmax = Imax;

        end

    end
end

%SECTION III: Output

fc = FC;
fm = FM;
```

```
a2max = max(A2/fm);
Itestt = (ismuthmax / a2max(1))*(A2/fm);
if or(ismuthmax == 0 || fm == 0, max(Itestt) < 0.1)
    I = ismuth;
else
    I = Itestt;
end
A = a;
disp('output frequency peaks: ')
disp(ffp)
disp('output tristimulus coefficients: ')
disp(thp)
disp('output tristimulus square error')
disp(therr)

figure(3)
stem(ffp, pp)
xlim([0 1.2*ffp(length(ffp))])
title('Output FFT frequency peaks')
xlabel('frequency (Hz)'); ylabel('magnitude');

if length(t) == length(A) && length(t) == length(I)
    %Plot envelopes if succesful envelope creation.
    figure(4)
    plot(t,A)
    title('Amplitude envelope')
    xlabel('time (s)');ylabel('amplitude');

    disp('amplitude error: ')
    disp(err)


    figure(5)
    plot(t,I)
    title('Index of modulation envelope')
    xlabel('time (s)');ylabel('amplitude');

    disp('index of modulation error: ')
    if (fm ~= 0)
        disp(err2/fm)
    else
        disp('N/A')
    end
end


end

Not enough input arguments.

Error in generator (line 17)
[y, fs] = audioread(wavfile); %read in file for analysis
```

Published with MATLAB® R2016b