Chromium 2

For solo guitar

Alexis Kirke

Edited by Nigel Morgan

This study score has been downloaded from the website archive of composer Nigel Morgan. The PDF file is solely for personal study, repertoire research or educational reference. It is not intended for use in public performance except in educational situations when an extract is required for illustration purposes.

Performance scores and parts are available from Tonality Systems Press in two formats: as standard printed and bound paper copies, and as PDF electronic masters carrying a special electronic license for an unlimited number of performances over an agreed period. For more information please e-mail **Tonality Systems Press**.

ISMN 979-0-57043-137-3 (Full Score)

Chromium 2

For solo guitar

Alexis Kirke

Alexis Kirke writes . . .

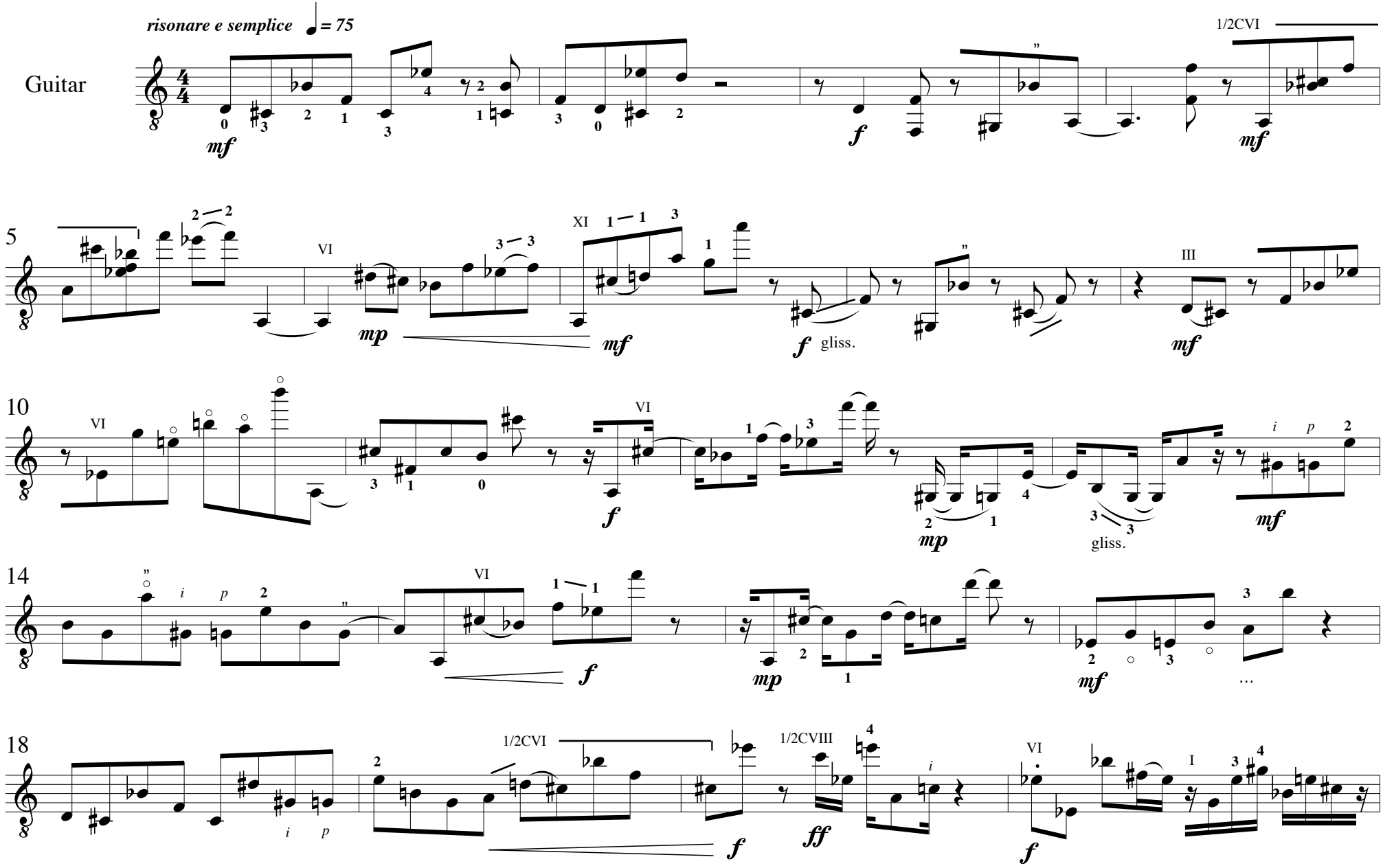
Chromium 2 was written in 2007, inspired by Nigel Morgan's piece for solo guitar *Dreaming Aloud*. it is based on short randomly generated motifs, which I then compose into the structure I desire. An important element in composing *Chromium 2* was producing motifs which would allow the piece to cohere as a solo guitar performance, and generating motifs which were appropriate to the four short sections of the piece. The piece was written using the mathematical programming language Matlab, with code developed by the composer.

Nigel Morgan writes . . .

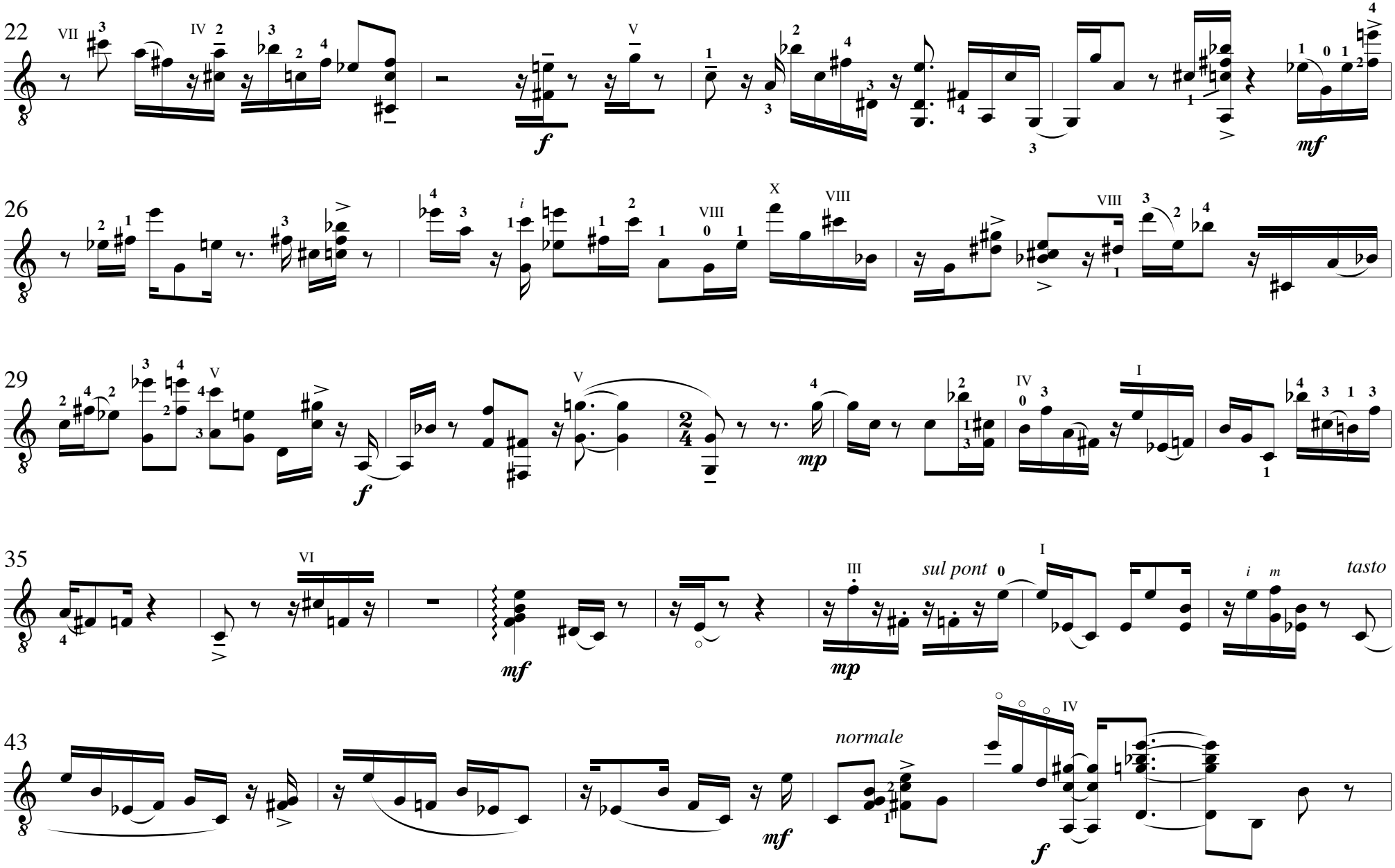
When I joined Future Music Lab I was keen to offer researchers the opportunity to illustrate something of their work through collaboration with a composer and performer. The solo guitar is well-known as an instrument that composers find difficult to write for. I wondered how mathematicians and computer scientists might approach it. In studying my work *Dreaming Aloud*, written using symbolic computation, Alexis was quick to pick up certain givens and constraints that composers use when writing for guitar and express these mathematically. His score *Chromium 2* very cleverly exploits his observations and results in a piece of considerable ingenuity and beauty.

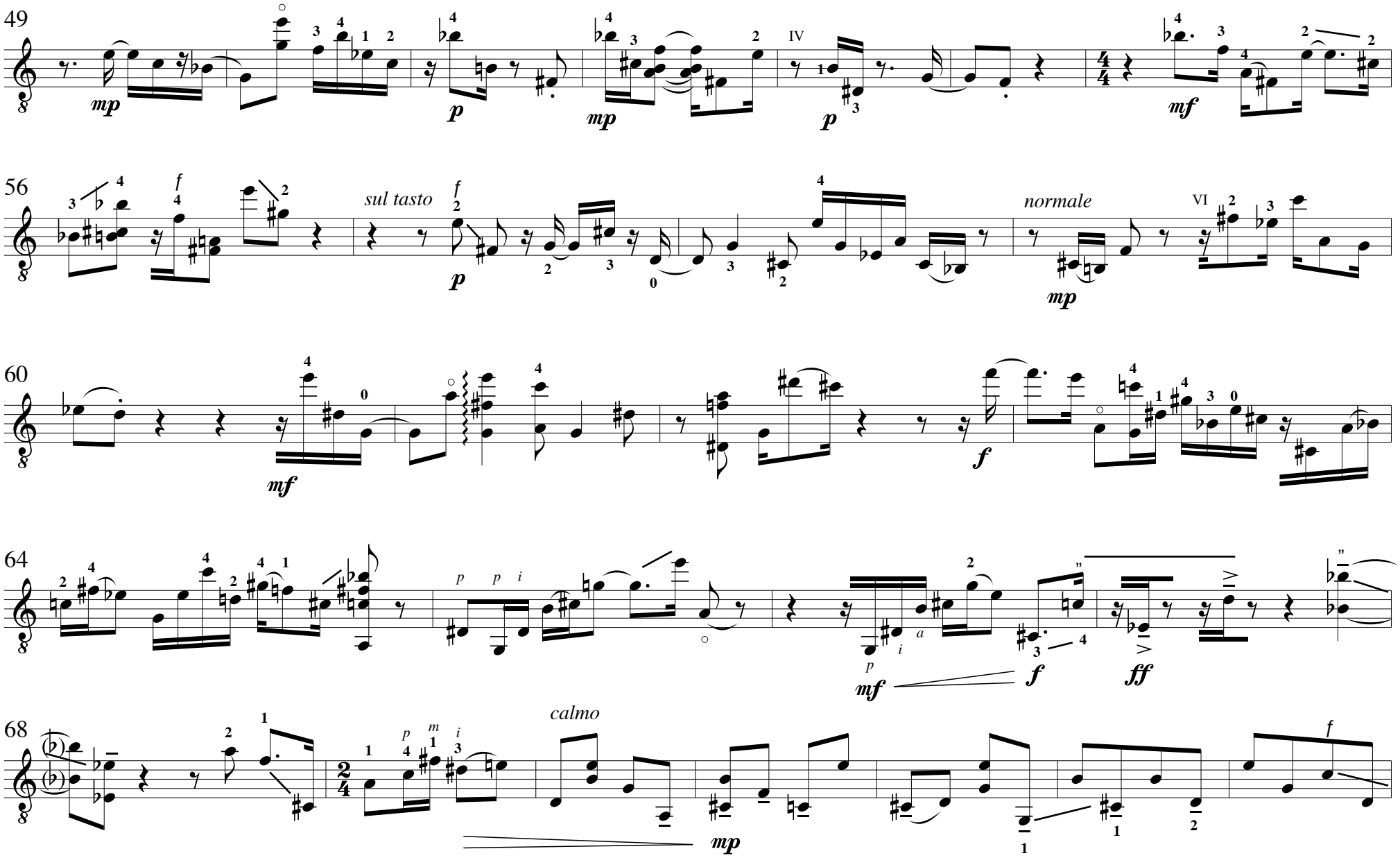
Chromium 2

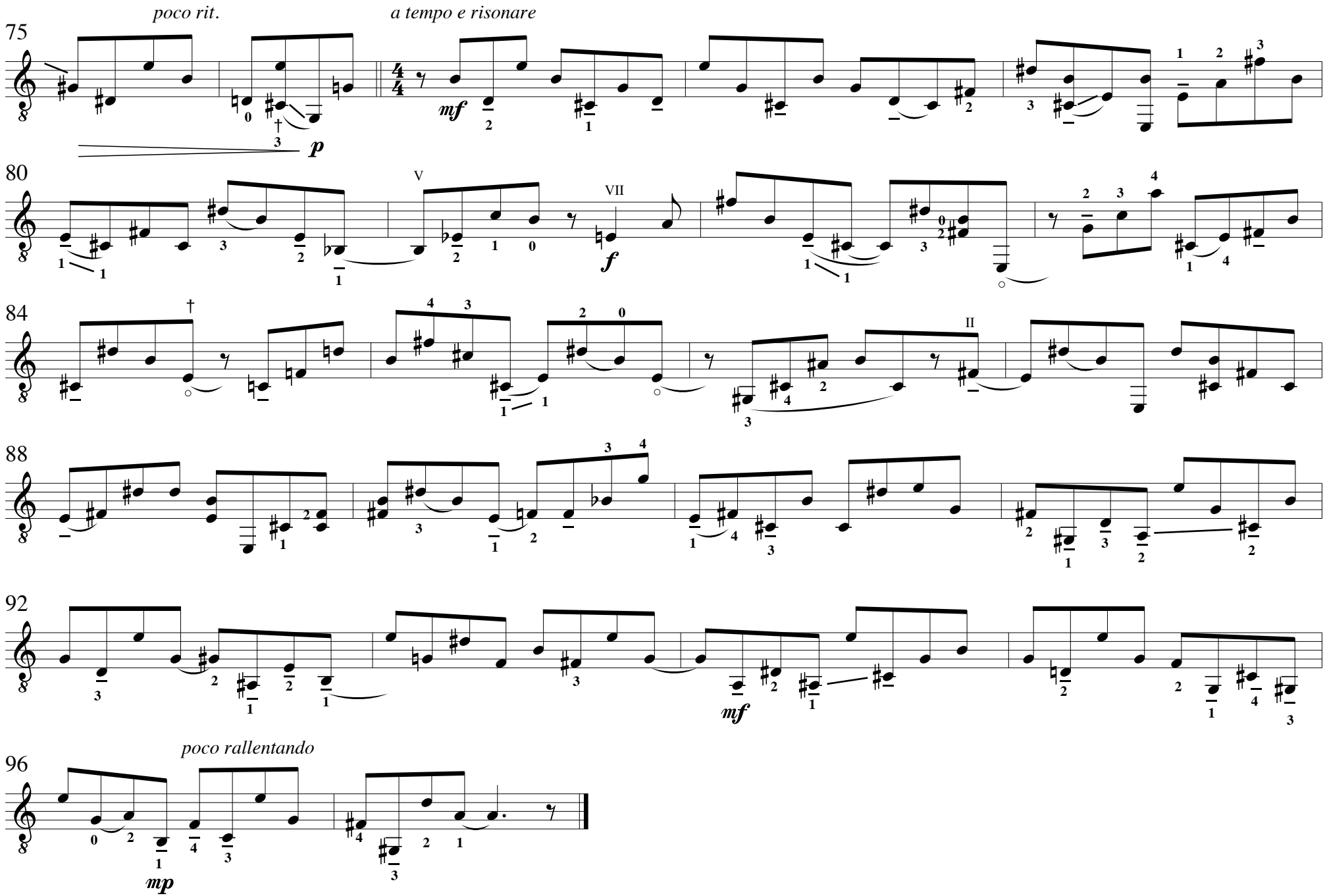
Alexis Kirke



© Copyright Tonality Systems Press 2009


22
26
29
35
43
sul pont
normale
tasto


49
mp
p
mp
IV
p
mf
56
f
sul tasto
f
p
normale
mp
VI
60
mf
f
64
p
mf
f
ff
68
calmo
mp
f


poco rit.
a tempo e risonare
75
mf
p
80
V
VII
f
84
II
88
92
mf
poco rallentando
96
mp

```
%chromium 2
% 6 November 2007
%solo guitar

clear all;
stringsList = [40,45,50,55,59,64];      %MIDI values for open guitar strings

%Compose Section A
N = {};
for i = 1:2
        N{i} = buildNoteSet(0.5);
end

M = {};

noteMatrix = N{1};    %Start the piece by playing the first motif unchanged.

for i = [1,2,2,1]
   if not(i==0)
      for j = 1:3
         M{j} = transformNoteSet(N{abs(i)},'rando');

         noteMatrix = buildChordMatrix(noteMatrix,M{j});
      end
   end
end
```

Generate motifs for Sections A and B. Build 2 random guitar-type motifs, with notes of duration 0.5 beats. buildNoteSet() is designed to compose guitar motifs and is listed and annotated further on in this document.

Now generate a section of form abba, where a = motif 1 repeated in 3 random transformations, and b = motif 2 repeated in 3 random transformations. buildChordMatrix() transforms a motif . It is designed to transform motifs in a guitar-like way. It is listed and annotated further on in this document. (NB buildChordMatrix() is a simple concatenation function which is used to sequentially build up MIDI files from the generated sections.)

```
% Compose Section B
M = {};

for i = [2,1,2,1]
    if not(i==0)
        for j = 1:3
            if rand > 0.75
                M{j} = transformNoteSet(N{abs(i)},'verti');
            else
                M{j} = transformNoteSet(N{abs(i)},'horiz');
            end
            M{j}(:,1) = M{j}(:,1) + round(rand*4)/4;
            noteMatrix = buildChordMatrix(noteMatrix,M{j});
        end
    end
end
```

Section B is of form baba, where a and b are motifs 1 and 2 respectively. Each motif is repeated 3 times in different random guitar transformations. 25% percent of the time the transformation is 'verti' (i.e. in fourths) and 75% of the time the transformation is 'horiz' (i.e. up and down the frets). The transformed motif is also shifted backwards in time up to 1 beat.

```
%Compose Section C
N = {};

for i = 1:2
    N{i} = buildNoteSet(0.25);
end
```

A new pair of motifs (we will call them '3' and '4') are composed for section C. The notes are half the duration of those in Sections A and B (0.25 beats), so on average section C will tend to be twice as fast as Sections A and B.

```
M = {};

for i = -2:2
    if not(i==0)
        for j = 1:24
            M{j} = transformNoteSet(N{abs(i)},'rando');
            noteMatrix = buildChordMatrix(noteMatrix,M{j});
        end
    end
end
```

Section C is of form dccd, where each c and d are motifs 3 and 4, repeated 24 times in different random guitar transformations, using transformNoteSet().

```
%Compose Section D
N = {};
for i = 1:2
    N{i} = buildNoteSet(0.5);
end

M = {};
for i = [1,2,2,1]
%i = 1;
    if not(i==0)
        for j = 1:9
          if rand > 0.5
              M{j} = transformNoteSet(N{abs(i)},'verti');
          else
              M{j} = transformNoteSet(N{abs(i)},'swap');
          end
          noteMatrix = buildChordMatrix(noteMatrix,M{j});
        end
```

Section D's motifs (5 and 6) have note durations of length 0.5 beats. Section D is of the form effe, where each e and f are motifs 5 and 6 (repspectively), repeated 9 times in different random guitar transformations using transformNoteSet(). 50% of the time the transformation is a vertical transformation up or down the frets. 50% of the time it involves swapping round in time up to 3 notes in the motif. The simplicity of the transformations is what leads to the effect of the final section.

```
    end
end


%now convert open notes
    for j = 1:6
    x = find(noteMatrix(:,4)==stringsList(j));
        if not(isempty(x))
            for k = 1:length(x)
                noteMatrix(x(k),2)=noteMatrix(x(k),2)*4;
            end
        end
    end
```

Now the whole piece is analysed to find where open strings could be played by the performer – and such strings could be left resonating while other strings are barred. To emphasise this effect, the duration of such open string note is multiplied by 4.

function noteSet = buildNoteSet(baseDuration)

```
stringsList = [40,45,50,55,59,64];      %MIDI values for open guitar strings
strings = stringsList;

openString = 1+round(rand*5);
strings(openString) = 0;        %mark as taken
freeString = 0;
while freeString == 0  %find a string apart from the open string
    coreString = 1+round(rand*5);
    freeString = strings(coreString);
end
strings(coreString) = 0;    %mark as taken
```

Each motif will contain at least one opportunity for an open string.

Having chosen an open string, now choose the "core" string and a "core note" on that string around which the motif pattern will be based. The purpose of having a core note, is to prevent the notes in the motif being too spread apart on the fret board – this increases playability and increases the chordability of a motif.

```
coreNote = stringsList(coreString)+2+round(rand*11);  %within 12 semitones

noteSet = [stringsList(openString), coreNote];
stringCount = 3;

while sum(strings) %until all strings taken

    freeString = 0;
    while freeString == 0  %find a string that's free
        nextString = 1+ round(rand*5);
        freeString = strings(nextString);
    end

    strings(nextString) = 0; %to show it's taken
    coreStringDist = abs(nextString-coreString);
    localDist = -2 + round(rand*3);
    %29/6 = average distance between strings in semitones
    totalSemitones = localDist + round(coreStringDist * (29/6));
    noteSet = [noteSet stringsList(coreString) + totalSemitones];

    stringCount = stringCount + 1;
end

localBeat = 0;
noteMatrix = [];
channel = 1; velocity = 127;
for i = 1:length(noteSet)
    duration = baseDuration;
    noteRow = [localBeat duration channel noteSet(i) velocity 0 0];
    noteMatrix = [noteMatrix; noteRow];
    localBeat = localBeat + baseDuration;
```

Select a random string from strings not yet used in the motif.

Select a random note on that string but ensure its fretboard distance from the core note is not too far, thus increasing the playability of the motif, and its chordability.

Convert the note list into a MIDI file section.

```
end
noteSet = noteMatrix;
function noteMatrix = transformNoteSet(noteMatrix,trans)

i = 0;
j = 0;
if strcmp(trans,'rando')
   i = round(1+rand*6);
   j = round(1+rand*7);
end

if strcmp(trans,'verti') || (i==1)|| (i==2)  || (i==3)
   transl = -6 + round(rand*12);
   for ll = 3:size(noteMatrix,1)
      noteMatrix(ll,4) = noteMatrix(ll,4) + transl;
   end
end

if strcmp(trans,'horiz') || (i==4) || (i == 5)
   transl = 6*(-2 + round(rand*4));
   for ll = 1:(size(noteMatrix,1))
      noteMatrix(ll,4) = noteMatrix(ll,4) + transl;
   end
end


if strcmp(trans,'swap') || (i==6) || (i == 7)
   n = round(1+rand*2);  %swap up to 3 notes
   for i = 1:n
      x = round(1+rand*5); %noteMatrix = noteMatrix(size(x,1):-1:1,:)
      y = round(1+rand*5);
      while x==y
```

This function can be called asking for a specific motif transformation type, or it can be told to randomly pick a transformation.

The Matlab || operator is a logical "OR" function. It is used here as a simple way to creative a probability distribution when the function is asked to randomly pick a motif transformation. For example, the first type 'verti' has 3 numbers OR'ed whereas 'horiz' has only 2 numbers OR'ed – this makes horizontal transformations slightly more likely than vertical. A more elegant programming solution could have been used, but this was sufficient for the composer's needs.

A 'horizontal' transformation means transposing the motif chord randomly up or down the strings (i.e. approximately in fourths). A 'vertical' transformation means moving the motif chord up and down the frets randomly. In both cases the algorithm attempts to keep as much of the chord shape as possible.

```
        y = round(1+rand*5);
      end
      temp = noteMatrix(x,1);
      noteMatrix(x,1) = noteMatrix(y,1);
      noteMatrix(y,1) = temp;
   end
end


if strcmp(trans,'clear 0.25') || (j==1)
   for i = 1:size(noteMatrix,1)
      if rand < 0.25
         noteMatrix(i,5) = 0;    %silent note = rest
      end
   end
end

if strcmp(trans,'clear 0.5') || (j==2)|| (j==7) || (j==6)
   for i = 1:size(noteMatrix,1)
      if rand < 0.5
         noteMatrix(i,5) = 0;    %silent note = rest
      end
   end
end

if strcmp(trans,'clear 0.75') || (j==3)
   for i = 1:size(noteMatrix,1)
      if rand < 0.75
         noteMatrix(i,5) = 0;    %silent note = rest
      end
   end
end
```

The ‘clear’ transformations randomly delete notes from the motif. ‘clear 0.25’ deletes approximate 25%, ‘clear 0.5’ deletes approximate 50%,‘clear 0.75’ deletes approximate 75%.

```
if strcmp(trans,'chord 0.25') || (j==4)
    for i = 2:size(noteMatrix,1)
        if rand < 0.25
            noteMatrix(i,1) = noteMatrix(i-1,1);    %play at same time
        end
    end
end

if strcmp(trans,'chord 0.5') || (j==5)
    for i = 2:size(noteMatrix,1)
        if rand < 0.5
            noteMatrix(i,1) = noteMatrix(i-1,1);    %play at same time
        end
    end
end

if strcmp(trans,'chord 0.75') || (j==6)
    for i = 2:size(noteMatrix,1)
        if rand < 0.75
            noteMatrix(i,1) = noteMatrix(i-1,1);    %play at same time
        end
    end
end
```

The ‘chord’ transformations randomly combine notes in the motif so they play at the same time. ‘chord 0.25’ combines approximate 25%, ‘chord 0.5’ combines approximate 50%,‘ chord 0.75’ combines approximate 75%.