1-1-2007

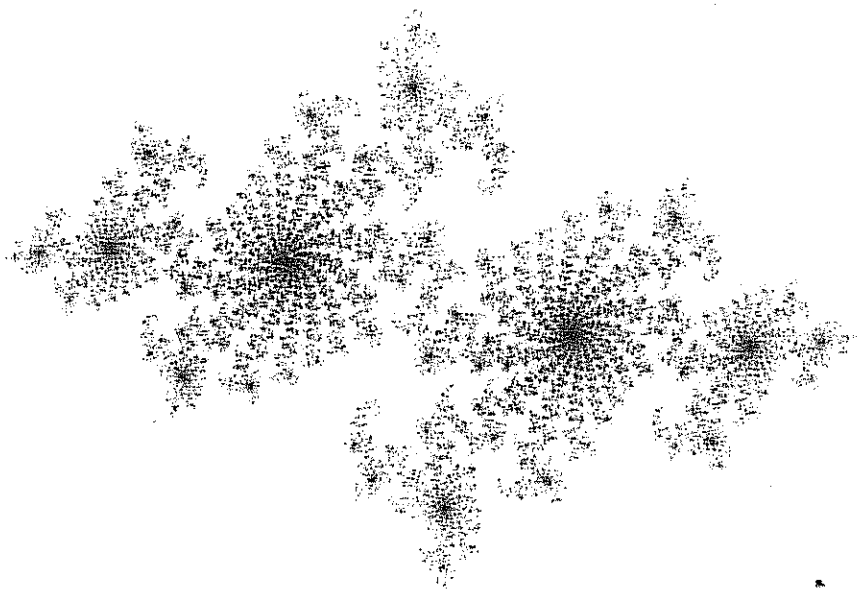
Composition and Analysis of Music Using Mathematica

Marianella P. Machado
Eastern Kentucky University, marianella.machado@eku.edu

Christopher W. Kulp
Lycoming College

Dirk Schlingman
Eastern Kentucky University, dirk.schlingman@eku.edu

Mathematica®
in EDUCATION and RESEARCH



VOLUME 12 NUMBER 1

2007

MATHEMATICA IN EDUCATION AND RESEARCH

Mathematica in Education and Research is published by iJournals.net.

ISSN1096-3324 (Print)

ISSN 1815-2627 (Electronic)

AIMS AND SCOPE

Mathematica in Education and Research publishes articles, notes, and special features on the use of *Mathematica* in all aspects of education and research. The scope of topics covered is wide-ranging and includes the use of *Mathematica* and *webMathematica* in the teaching of technical subjects in all of the engineering, scientific, and mathematical disciplines as well as in the carrying out of basic and applied research in these areas. The intended audience for the articles is broad in its interests and includes students, professors, professionals and specialists as well those without an extensive background in the subject of the article. The address of the editorial office is:

MiER
PO Box 6546
St Lucia
Queensland 4067
Australia.
email: mier@ijournals.net.

SUBSCRIPTION INFORMATION

Mathematica in Education and Research is published four times annually in January, April, July, and October. Annual subscription rates, in US Dollars, are: Individuals: $75 (print & electronic), $45 (electronic); Institutions: $160 (print & electronic). For print subscriptions additional shipping and handling charges, in US Dollars, apply: USA $20, all other countries $35. Subscriptions can be placed online at www.ijournals.net. All major credit cards are accepted.

Electronic subscriptions are accessible from the downloads page of the iJournals website. This page is located on the main menu and is visible following successful login to the website.

Mathematica
IN EDUCATION
AND RESEARCH

CONTENTS

VOLUME 12, NUMBER 1, 2007

Articles

Columns

Reviews

Composition and analysis of music using Mathematica

Christopher W. Kulp, Marianella Machado, and Dirk Schlingmann

Abstract

In this paper we demonstrate how to create and analyze musical compositions using *Mathematica*. In §1, we begin by demonstrating how to create a musical composition for an orchestra based on the butterfly curve using traditional means. In §2, we then show how a computer generated piece based on any curve can be composed using *Mathematica*. Finally, in §3 we show how *Mathematica* can be used to analyze musical compositions using the methods from nonlinear time series analysis.

1. The composition of an orchestral piece using the butterfly curve

1.1 The relationship between music and mathematics

In the composition of a musical work, it is common to find that musical transformations are closely related to geometric transformations [1]. For example, in musical terms, a geometric reflection occurs when notes are symmetric about some line in a staff. In this paper, we study the relationship between music and mathematics through the use of *Mathematica*. This approach is based on the premise that the compositional process of a musical work could be generated from a graph of a mathematical expression by using geometric transformations as patterns for musical transformations.

In this attempt, the graphical expression of a parametric curve executed through *Mathematica* is taken as a starting point for determining the compositional features and creating form in an acoustical piece. The following section contains a description of the compositional process of an orchestral work based on a butterfly curve [2].

```
In[1]:= r[t_] := {Sin[t] (e^Cos[t] - 2 Cos[4 t] + Sin[t / 12]^5),
          Cos[t] (e^Cos[t] - 2 Cos[4 t] + Sin[t / 12]^5)};
        ParametricPlot[r[t], {t, 0, 48}];
```

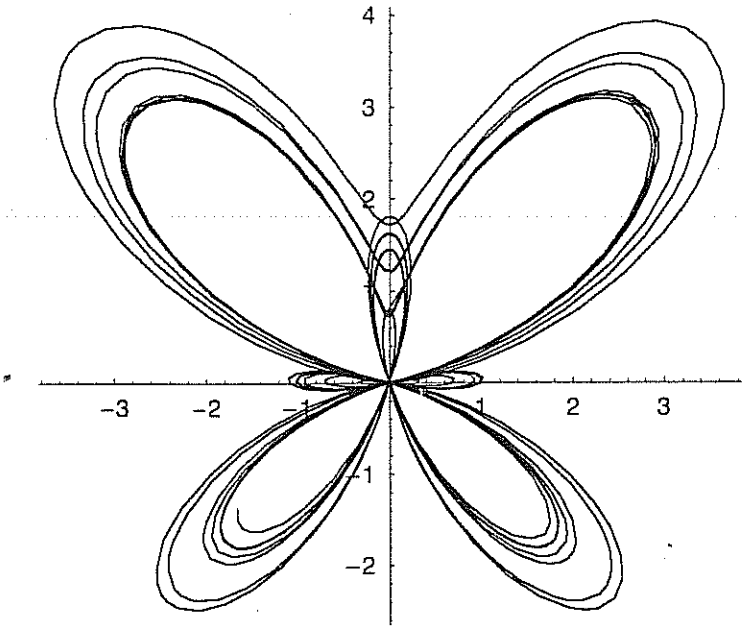


Fig. 1 The butterfly curve.

The creative process for this composition will be done in two steps. Step one refers to the musical parameters that would be equivalent to the butterfly curve and how they function in order to generate a theme as the basis of the composition. For example, the x and y axes correspond to duration and pitch, respectively. The second step takes elements from the first step and creates a complete musical form for an orchestral piece. For more information, the interested reader is directed to [3].

1.2 The musical parameters and their equivalent parameters in a butterfly curve

In this musical representation of the butterfly curve, the horizontal axis refers to duration and the vertical one, to pitch. A measure is created as follows:

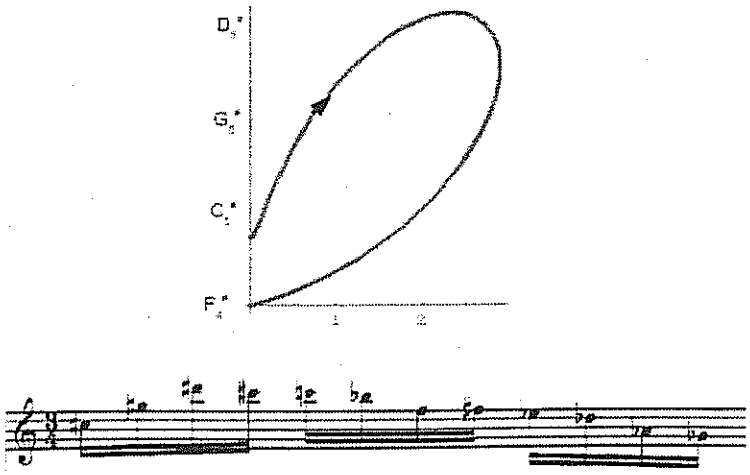


Fig. 2 Creation of a measure of music using one section of the butterfly curve.

Notice that in Fig. 2 the path of the butterfly curve and the notes in the piece are related. As the parameter, t, of the curve increases, the curve follows the direction of the arrow. Notice how the pitch of the notes increases then decreases as suggested by the curve. Further, note that the x axis crosses the y axis at $y = F_4^{\#}$. The scale in Fig. 2 was chosen to accommodate the register for a treble clef instrument, e.g. a violin.

As the graph below shows, a similar procedure is used to create three measures:

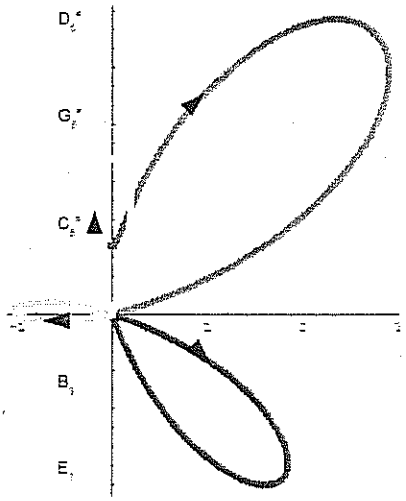


Fig. 3 Four lobes of the butterfly curve, with arrows denoting the progression through the curve as its parameter, t, increases.

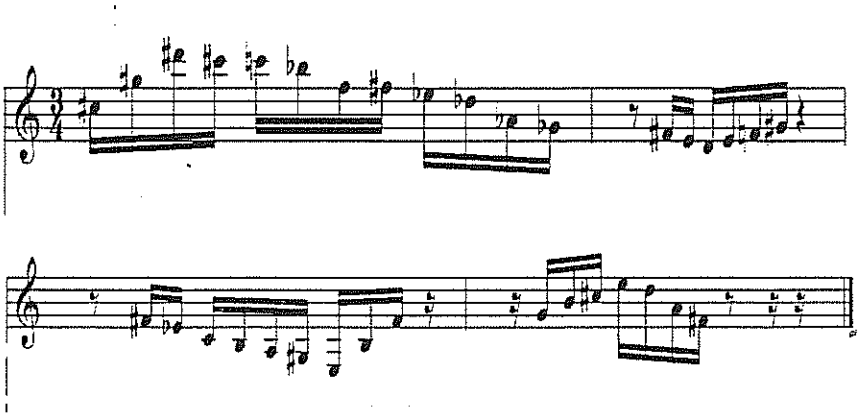


Fig. 4 Four measures as created by the butterfly curve in Fig. 3.

In Fig. 3, as in Fig. 2, the first measure is formed by the lobe in Quadrant I. As the curve's parameter increases, more lobes are formed. The second measure is formed by the lobe that straddles the negative x axis. Continuing along the curve, the third measure is created by the lobe in Quadrant IV. Finally, the fourth measure is created by the lobe that straddles the positive y axis. A similar process will be used to compose the 16-measure theme of the piece. For more details about the composition of the orchestral piece, the interested reader is referred to [3].

1.3 The compositional process that unfolds over time, following the butterfly curve

The path of the butterfly curve draws different curve lengths and shapes in each quadrant. Therefore, some measures will have more notes than others. Also, its visual representation has a mirror-like structure. This implies that some measures will be mirror images of others. Following the procedure above, one can develop the melodic theme as well as create the harmonic plan of the piece.

The musical discourse consists of four distinctive parts. The first part consists of the 16 measures shown above. The second part consists of 32 measures where we double the length of the tones in the original 16 measures. We can continue this process to create a 64-measure section and a 128-measure section. Up to this point, there are 240 measures. However, these 240 measures are not the composition itself, but rather, they provide the building blocks of the composition.

We create the musical piece using the butterfly curve again. This time we use a looser interpretation of the butterfly curve for a musical purpose. In particular, the elements of the y axis correspond to melody, harmony, and rhythm, while the x axis corresponds to the dynamics (or intensity) and tempo. The butterfly curve dictates the order and volume in which the measures are played. The instrumentation is chosen by using a third axis. Different regions of the third axis correspond to different families of instruments.

2. Using *Mathematica* to create computer generated musical compositions

With *Mathematica* it is easy to create sound whose amplitude is given by a function f. For example, `Play[Sin[440*2*π*t], {t, 0, 1}]` plays a pure tone with a frequency of 440 Hz. Hence tools like *Mathematica* can be used to create new music. Let us demonstrate how random points in a three-dimensional space or points on a parametric curve can be turned into musical compositions.

2.1 Musical representation of random points in three-dimensional space

In a simplified way, a traditional musical composition is a collection of finite tones (frequencies) that are played at certain times for certain durations. A three-dimensional point $\{x, y, z\}$ could be interpreted as a tone that is played at time x (measured in seconds), with frequency y (measured in Hz), for a duration z (measured in seconds). In this fashion, a list of three-dimensional points represents a musical composition.

Here we look at a list of two three-dimensional points, {{0.655309, 217.411, 0.740096}, {3.53015, 620.606, 1.65959}}. The corresponding composition consists of two tones. The first tone is played after 0.655309 seconds, with frequency 217.411 Hz, for 0.740096 seconds. The second tone is played after 3.53015 seconds, with frequency 620.606 Hz, for 1.65959 seconds.

To turn these points into a musical composition, we first create a function, `composition[t]`:

```
In[6]:= composition[t_] :=
          Piecewise[{{0, t < 0.655309}, {Sin[217.411 * 2 * π * t],
              (t ≥ 0.655309) && (t < 0.655309 + 0.740096)}}] +
           Piecewise[{{0, t < 3.53015}, {Sin[620.606 * 2 * π * t],
              (t ≥ 3.53015) && (t < 3.53015 + 1.65959)}}];

        Play[Evaluate[composition[t]],
         {t, 0, Max[{0.655309 + 0.740096, 3.53015 + 1.65959}]}];
```

Of course, we might need to scale the coordinates of the three-dimensional points to reasonable values so that the times (when tones are played), frequencies, and durations are in acceptable ranges. For example, we do not want to include negative durations or frequencies outside the range of human hearing. 000000000000000

Now we bring all these individual steps together into a program called **createMusic**, which takes any finite list of three-dimensional points as input and turns it into a composition. As a side effect, a three-dimensional graph of the scaled three-dimensional points will also be displayed.

```
In[8]:= Clear[createMusic];
        Options[createMusic] = {TimeRange → {0.0, 30.},
          FrequencyRange → {164.81, 1244.5},
          DurationRange → {0.25, 3.0}};

        createMusic::usage = "createMusic creates a
           musical composition from a list of triplets";

        TimeRange::usage =
         "TimeRange is an option for the createMusic function.
           The input for TimeRange consists of a list of
           two elements, {x,y}; where x is the time (in
           seconds) when the composition begins, and y is
           the time (in seconds) when the composition ends";

        FrequencyRange::usage =
         "FrequencyRange is an option for the createMusic
           function. The input for FrequencyRange consists
           of a list of two elements, {x,y}; where x is
           the minimum frequency (in Hertz) played, and y
           is the maximum frequency (in Hertz) played";

        DurationRange::usage =
         "DurationRange is an option for the createMusic function.
           The input for DurationRange consists of a list of
           two elements, {x,y}; where x is the shortest time (
           in seconds)for which a tone played, where x is the
           longest time (in seconds)for which a tone played.";

        createMusic::"err1" = "Invalid time range.";

        createMusic::"err2" =
         "Invalid frequency range. Min and max frequencies must be
           between 100 and 20000 Hz with min freq < max freq.";

        createMusic::"err3" =
         "Invalid tone duration range. Minimum tone duration must
           be positive and less than maximum tone duration.";

        createMusic[data_, opts___] :=
         Module[{ldata, events, default, whenmin, whenmax,
           freqmin, freqmax, durmin, durmax, xmin, xmax, ymin,
```

```
 ymax, zmin, zmax, whenscaled, whenslope, freqscaled,
 freqslope, durscaled, durslope, scaleddata, sorteddata,
 when, freq, dur, composition, compositionLength, t},
ldata = N[data];
events = Length[ldata];

{{whenmin, whenmax},
  {freqmin, freqmax}, {durmin, durmax}} =
 {TimeRange, FrequencyRange, DurationRange} /. {opts} /.
  Options[createMusic];

If[whenmax ≤ whenmin || whenmin < 0,
 Message[createMusic::"err1"]; Return[$Failed] ];

If[freqmin < 100 || freqmax > 20000 || freqmin ≥ freqmax,
 Message[createMusic::"err2"]; Return[$Failed]];

If[durmin < 0 || durmax ≤ durmin,
 Message[createMusic::"err3"]; Return[$Failed]];

{{xmin, xmax}, {ymin, ymax}, {zmin, zmax}} =
 {Min[#], Max[#]} & /@ Transpose[ldata];

If[xmin == xmax, whenslope = 0.0,
 whenslope = (whenmax - whenmin)/(xmax - xmin)];
If[ymin == ymax, freqslope = 0.0,
 freqslope = (freqmax - freqmin)/(ymax - ymin)];
If[zmin == zmax, durslope = 0.0,
 durslope = (durmax - durmin)/(zmax - zmin)];

sorteddata = Sort[({whenslope, freqslope, durslope}
       (# - {xmin, ymin, zmin}) +
      {whenmin, freqmin, durmin} &) /@ ldata];

{when, freq, dur} = Transpose[sorteddata];

composition[t_] = Sum_{i=1}^{Length[sorteddata]} Piecewise[

   {{0, t < when[[i]]}, {Sin[freq[[i]] * 2 * π * t],
     (t ≥ when[[i]]) && (t < when[[i]] + dur[[i]])}}];

compositionLength = Max[when + dur];

Show[
 Graphics3D[{
```

```
    {Black, AbsoluteThickness[0.8],
     Line[{{1, 0.01, 1} #, {1, 0.01, 0} #}] & /@ sorteddata},
    {Red, PointSize[0.02], Point[{1, 0.01, 1} #] & /@
      sorteddata},
    {Blue, AbsoluteThickness[1.0],
     Line[({1, 0.01, 1} # &) /@ sorteddata]},
    Play[Evaluate[composition[t]], {t, when[[1]],
       compositionLength}, DisplayFunction → Identity,
      SampleRate → 2 * Max[freq][[1]]} ],
  ViewPoint → {1.300, -2.400, 2.000},
  BoxRatios → {3, 1.5, 0.75}, ImageSize → 400,
  AxesLabel → {"when (s)", "freq × 10^-2 Hz", "dur (s)"}];
]
```

In[18]:=
```
Clear[data];

data =
 Table[{Random[Real, {-30, 30}], Random[Real, {-50, 50}],
   Random[Real, {-20, 20}]}, {i, 1, 100}];

createMusic[data, TimeRange → {0, 10},
 DurationRange → {0.1, 0.2}, FrequencyRange → {100, 1000}]
```

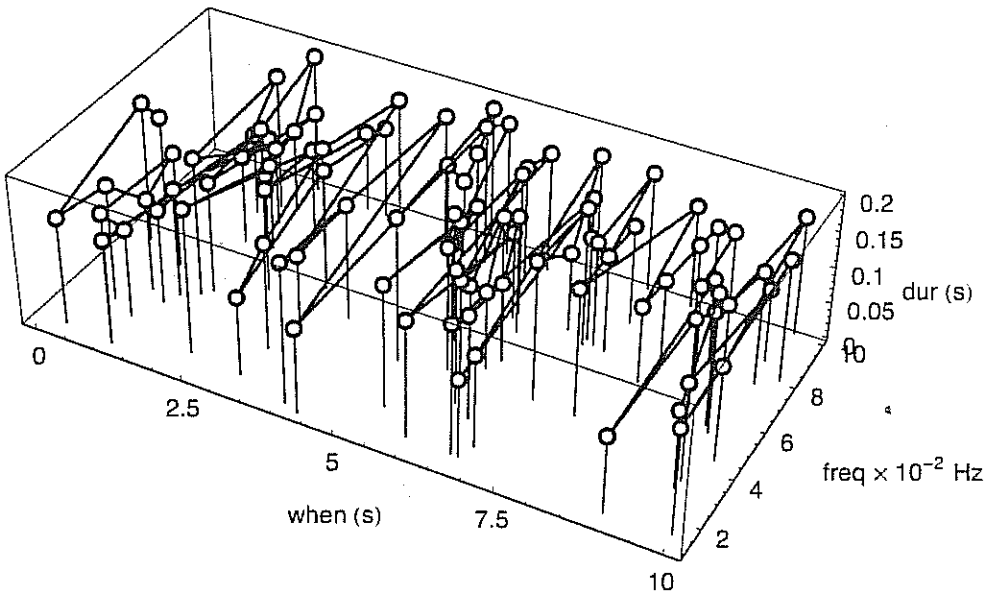


Fig. 5 A plot of the musical variables created from random data.

2.2 Discrete musical representation of two-dimensional parametric curves

We can also experiment with data that originates from two-dimensional parametric curves. Instead of picking random points in a three-dimensional space as in §2.1, we choose points `{t, r[t][[1]], r[t][[2]]}` where `r[t]={r[t][[1]], r[t][[2]]}` represents any two-dimensional parametric curve, `t` represents the time (when the event occurs or when the tone is played), `r[t][[1]]` represents the frequency, and `r[t][[2]]` represents the duration. Again, the ranges for `r[t][[1]]` and `r[t][[2]]` might need scaling to get reasonable frequencies and durations.

For example, here is a parametric representation of the butterfly curve, [2]. For its two-dimensional parametric plot, see Fig. 1.

```
In[21]:= Clear[r];
         r[t_] := {Sin[t] (e^Cos[t] - 2 Cos[4 t] + Sin[t / 12]^5),
           Cos[t] (e^Cos[t] - 2 Cos[4 t] + Sin[t / 12]^5)};
```

Now we create data that is used in §3 and that is based on the parametric curve `{t, r[t][[1]], r[t][[2]]}` where `r[t]` represents the butterfly curve from above. Instead of creating random times when tones should be played, we would like a tone to be played every quarter second. A quarter second represents a 1/16 note in traditional music scoring. If we would like to use 3/4-measures, then one 3/4-measure equals 3 seconds and has 12 (1/16) notes. To create a composition of sixteen measures, we need $48 = 3 \times 16$ seconds or $192 = 16 \times 12$ (1/16) notes.

```
In[23]:= Clear[data, i, j];
         data = Table[{j = i 0.25, r[j][[1]], r[j][[2]]}, {i, 1, 192}];
```

Here is a three-dimensional plot of the data points and the parametric curve `Prepend[r[t], t]={t, r[t][[1]], r[t][[2]]}`.

```
In[25]:= Clear[p3DPlot];
         p3DPlot = ParametricPlot3D[Prepend[r[t], t],
           {t, 0, 48}, AspectRatio → Automatic,
           ViewPoint → {4.0, -0.5, 1.0}, PlotPoints → 300,
           BoxRatios → {3, 0.7, 0.7}, DisplayFunction → Identity];

         Clear[points];
         points = Table[{Red, PointSize[0.02], Point[data[[i]]]},
           {i, 1, Length[data]}];

         Clear[pointsPlot];
         pointsPlot =
          Show[Graphics3D[points], ViewPoint → {4.0, -0.5, 1.0},
           BoxRatios → {3, 0.7, 0.7}, DisplayFunction → Identity];

         Show[{pointsPlot, p3DPlot},
          DisplayFunction → $DisplayFunction, ImageSize → 188];
```

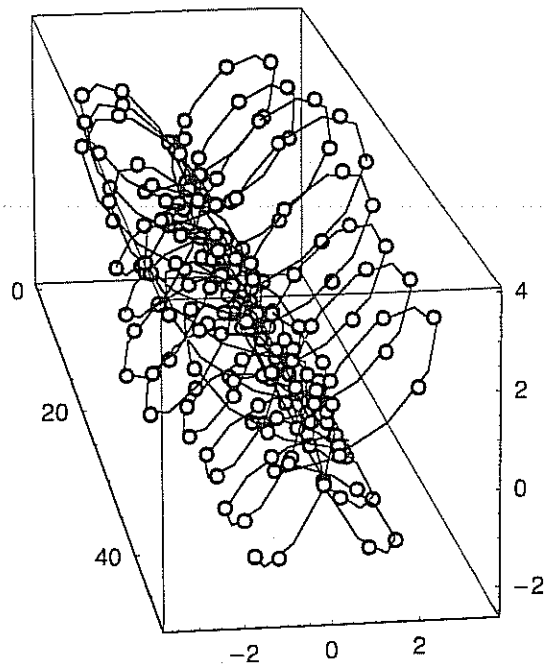


Fig. 6 A three-dimensional rendering of the butterfly curve. The red dots denote locations on the curve which are used to generate the composition.

Here is the musical representation of the data.

```
In[32]:= createMusic[data]
         (* Pick the default option settings. *)
```

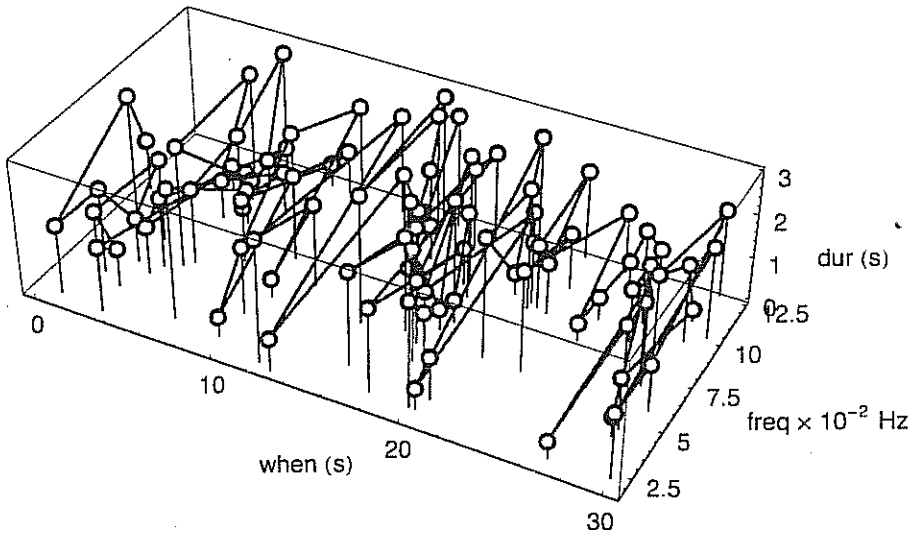


Fig. 7 A plot of the musical variables used in the composition generated from Fig. 6.

2.3 Continuous musical representation of parametric functions

We could also turn the individual components of the parametric function r that created the butterfly curve into continuous and overlapping melodies as demonstrated below. The possibilities are endless for creating interesting and intriguing sounds and music. Explore and have fun.

```
In[33]:= Clear[f, g, t];
         f[t_] = r[t][[1]];
         g[t_] = r[t][[2]];
         Clear[freqmin, freqmax, xmin, xmax, xfreqsqualed,
          ymin, ymax, yfreqsqualed, scaledPlot];
         xmin = -4.0;
         xmax = 4.0;
         ymin = -3.0;
         ymax = 4.0;
         freqmin = 164.81;
         freqmax = 1244.5;
         xfreqscaled[x_] = (freqmax - freqmin)/(xmax - xmin) * (x - xmin) + freqmin;
         yfreqscaled[y_] = (freqmax - freqmin)/(ymax - ymin) * (y - ymin) + freqmin;

         Play[Sin[2.0 * Pi * xfreqscaled[f[t]] * t] +
           Sin[2.0 * Pi * yfreqscaled[g[t]] * t], {t, 0.0, 2.0 * Pi}];
```

3. Using time series to find patterns in musical compositions

In this section, we study compositions in order to find some patterns which may not be apparent by simply listening to the piece. In order to analyze the composition mathematically, we can convert it into a time series. A time series is simply a list of numbers which are the measured value of some physical quantity at some regular time interval, Δt. Time series are typically written in the form:

$$S = \{s_1,\ s_2,\ \ldots,\ s_i,\ \ldots,\ s_n\} \tag{1}$$

where, the number s_i is the value of the measured quantity at a time of $i\Delta t$ after the initial measurement (which, without loss of generality, we assume to be at $t = 0$).

The methods of time series analysis provide a powerful tool in extracting information from the measurements of a physical system [4, 5]. In this section, the system of interest will be musical compositions. Using time series analysis on musical compositions is not new. In references [6] and [7], analysis was done on digital recordings of various compositions. *Mathematica* can easily import digital recordings. However, the time series that *Mathematica* generates to represent the recording may not be useful in some analyses. In the following sections, we will discuss how to generate a time series from a composition as well as demonstrate two different analyses that can be done with compositions.

3.1 Generating time series from music

Mathematica can import digitally recorded music, in the form of .wav files, and convert them into a time series. The time series produced by *Mathematica* contains information about the amplitude of the sound measured at the sampling rate at which the music was recorded. For many applications, this is

sufficient. However, the amplitude information is performer-dependent, meaning that the performer has a large influence on the value of the amplitude at any give time. Different performers may play the same piece at different amplitudes leading to two different time series for the same piece! This can be a problem if one is searching for patterns in compositions written by one composer, or if one is comparing compositions by various composers. In order to do such comparisons, we need a time series representation of the piece which is performer-independent. As we will see, we can produce such a time series directly from the piece's sheet music.

We demonstrate how to generate a time series from sheet music by using the example illustrated in Fig. 8.

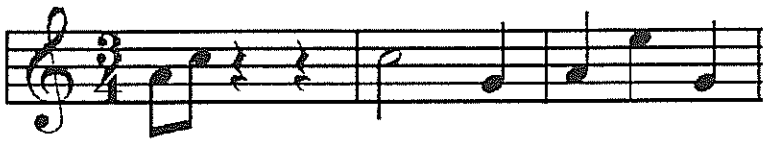


Fig. 8 Three measures of a song from which we will generate a time series.

We begin by noting that our time interval between elements of the time series will be, $\Delta t = 1/16$ *note*. A quarter-note, for example, will be represented by four sequential elements in the time series. Because this composition is in 3/4 time, each measure will produce 12 elements of the time series. We chose 1/16 note for our interval because this is the shortest note that is most commonly used. Converting to a time interval of a 1/32 note is straightforward. The elements of the time series are the frequency of each note. The first measure contains: one A_4 (440 Hz) 1/8 note which makes up two sequential elements of the time series, one C_5 (523.35 Hz) 1/8 note which makes up two elements, and two 1/4 rests (0 Hz) each consisting of four elements. The time series for the first measure is:

```
In[47]:= Measure1 = {440, 440, 523.35, 523.35, 0, 0, 0, 0, 0, 0, 0, 0};
```

It is clear that for longer compositions, this process can be quite tedious. We can use *Mathematica* to speed up this process. Let's begin by identifying the notes that appear in the composition and their corresponding frequencies. The other notes that appear are: G_4 (392 Hz) and E_5 (659.25 Hz). We assign each note its frequency value:

```
In[48]:= A4 = 440; C5 = 523.35; G4 = 392; E5 = 659.25;
```

Next, we write each measure in *Mathematica*. Each measure is represented as a list of ordered pairs. The first element of each ordered pair is the note, and the second element is the number of 1/16-notes for which the note is played:

```
In[49]:= measure = Table[Null, {i, 1, 3}];
         measure[[1]] = {{A4, 2}, {C5, 2}, {0, 4}, {0, 4}};
         measure[[2]] = {{C5, 8}, {G4, 4}};
         measure[[3]] = {{A4, 4}, {E5, 4}, {G4, 4}};
```

Next, we expand each measure into a variable called **ExpandedMeasure**. For example, **measure[[1]]** should be expanded into **ExpandedMeasure[[1]]** which should be equal to **Measure1**.

```
In[53]:= padNotes[{note_, len_}] := PadRight[{}, len, note];
         ExpandedMeasure = Flatten /@ Map[padNotes, measure, {2}];
```

Next, we form the time series for the whole song:

```
In[55]:= piece = Flatten[ExpandedMeasure];
         ListPlot[piece,
          FrameLabel → {"n", " Frequency"}, PlotJoined → True];
```

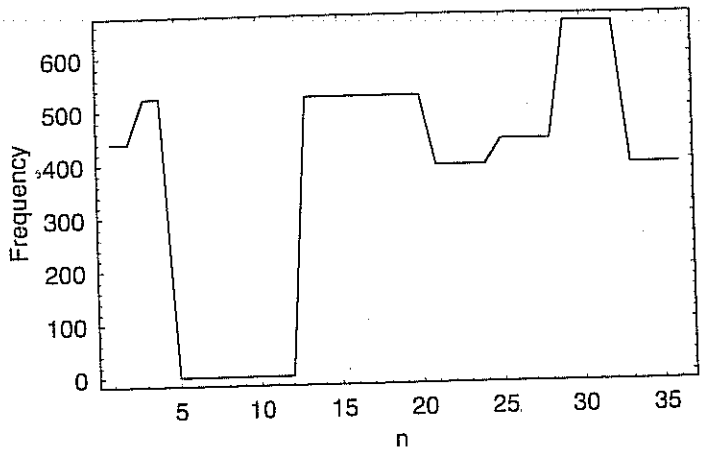


Fig. 9 The time series generated from Fig. 8.

Using this technique, we can then begin to analyze and compare the time series generated from various musical compositions. We note that using this technique we can not distinguish between, say, an A_4 1/2-note and two consecutive A_4 1/4-notes. Including such information would complicate the time series. However, including such information is an interesting problem for future research.

3.2 Visualization of compositions

In this section, we will produce visualizations of compositions produced from the butterfly curve in §1 and 2 using a method similar to that used in [6] and [7].

Consider the time series in eqn (1). To embed a time series, we need to create a coordinate system from the time series. This is done by forming a list of ordered pairs from the time series,

$$\{\{s_1, s_{1+\tau}\}, \{s_2, s_{2+\tau}\}, \ \ldots, \ \{s_{n-\tau}, s_n\}\}. \tag{2}$$

How do we choose which two elements of a time series to pair-up as coordinates (*i.e.* how do we choose τ, also known as the *delay*)? One method to determine τ is to use the first minimum of the average mutual information [8]:

$$MI(\tau) = \sum_{s_i s_{i+\tau}} p(s_i, s_{i+\tau}) \log_2\left(\frac{p(s_i, s_{i+\tau})}{p(s_i)p(s_{i+\tau})}\right), \tag{3}$$

where $p(s_i)$ is the probability that the element s_i appears in S, and the joint probability, $p(s_i, s_{i+\tau})$ is the probability that the pair $\{s_i, s_{i+\tau}\}$ appears in the time series. The logarithm is taken to base two to give units of bits. One way of interpreting eqn (3) is that it tells us how much information (in bits) on the average is known about the measurement $s_{i+\tau}$, when a measurement of s_i is made. Coordinates in $\mathbb{R}^n$ are composed of independent quantities; hence, we want the coordinates for our visualization to also contain quantities which are as independent as possible but are also closely related in time. By

convention, the first value of τ for which there is a local minimum of $MI(\tau)$ is the delay used for the embedding.

There is one more note that needs to be made about computing the average mutual information. Typically one symbolizes the time series before performing calculations. For example, in this article we symbolize a data set by choosing some number which serves as a threshold. If an element has a value greater than the threshold, that element is assigned the value (or symbol) of 1; otherwise, it is assigned a value (or symbol) of 0. For this paper, we will choose the threshold which maximizes the resulting binary series Shannon Entropy [9, 10]. Symbolizing time series is done for many reasons, such as speeding up calculations for larger series. For more information on symbolization, the interested reader is referred to [11].

We begin by looking at the visualization of the 16-measure theme created from the butterfly curve without using *Mathematica* (§1). The time series representation of this piece is created from the treble staff of the piece using the second method discussed in §3.1.

```
In[57]:= songtrad = Flatten[Import["bftradtrebel.xls"]];
         ListPlot[songtrad, FrameLabel → {"i", "frequency"}];
```

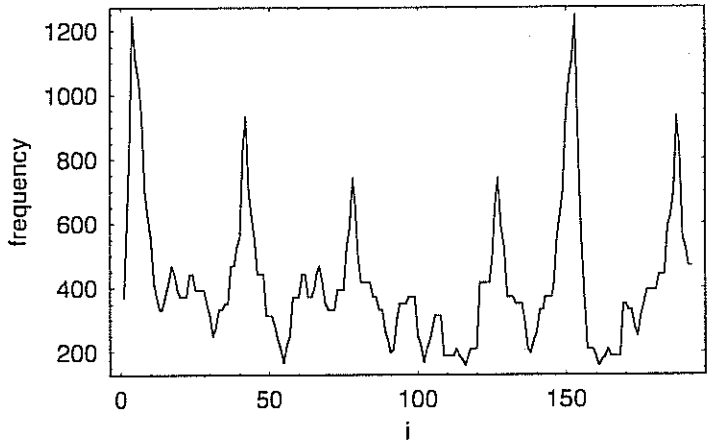


Fig. 10 The time series generated from the part of the composition discussed in §1.

Next, we will compute the mutual information for this series to find the proper lag for the embedding. We begin this process by binning up the series *songtrad* into a binary series, using the threshold which maximizes the Shannon Entropy of the binary series.

```
In[59]:= threshold = 381;
         s = Table[IntegerPart[If[songtrad[[i]] > threshold, 1, 0]],
           {i, 1, Length[songtrad]}];

         (* probability that element x appears in the time series *)
         p[x_] := Count[s, x] / Length[s];

         Do[U = Table[{s[[i]], s[[i + τ]]}, {i, 1, Length[s] - τ}];
              pj[x_, y_] := Count[U, {x , y}] / Length[U];
          (* pj computes the joint probability *)

          (* MIτ is the mutual information for delay τ *)
```

```
    MIτ = Sum[If[pj[i, j] == 0, 0,
      pj[i, j] * Log[2, pj[i, j] / (p[i] * p[j])]],
     {i, 0, 1}, {j, 0, 1}], {τ, 0, 20}]

mutualinfo = Table[N[MIτ, 3], {τ, 0, 20}];

ListPlot[mutualinfo,
 FrameLabel → {"Time Delay + 1", "Mutual Information"},
 PlotStyle → PointSize[0.02]];
```

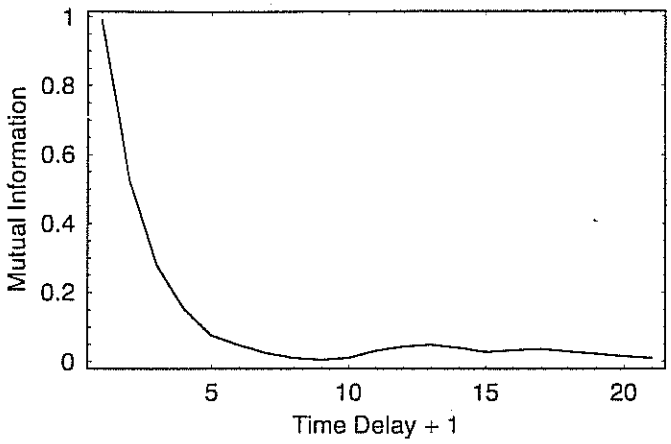


Fig. 11 A plot of the mutual information as a function of time delay from the time series generated from the composition discussed in §1.

It is difficult to tell where the first minimum of the mutual information occurs in Fig. 11. However, we can use the below calculation to show that the first minimum occurs at $\tau = 8$. Using this time lag, we can create our visualization.

```
In[65]:= First[Flatten[Position[mutualinfo, Min[mutualinfo]]]] - 1;

In[66]:= τ = 8;
         phase2Dt = Table[{songtrad[[i]], songtrad[[i + τ]]},
           {i, 1, Length[songtrad] - τ}];

         ListPlot[phase2Dt, FrameLabel → {"si", "si+8"}];
```

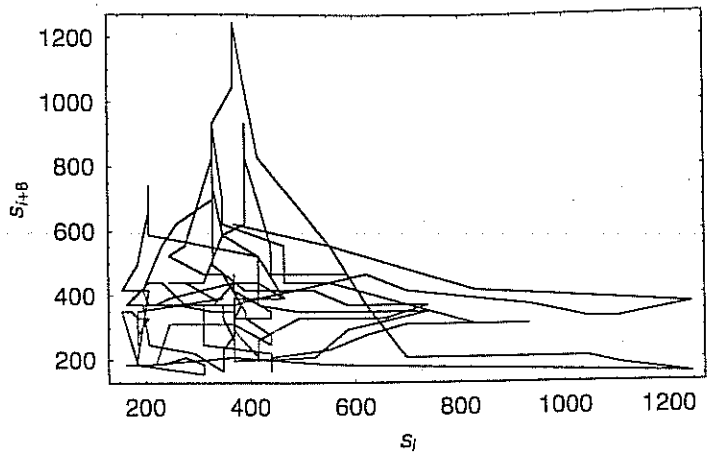


Fig. 12 A visualization of the composition from Section 1.

Next, we will repeat the process for the piece composed using *Mathematica* in Section 2.

```
In[69]:= songmath = Flatten[Import["bflymath.xls"]];
         ListPlot[songmath, FrameLabel → {"i", "frequency"}];
```

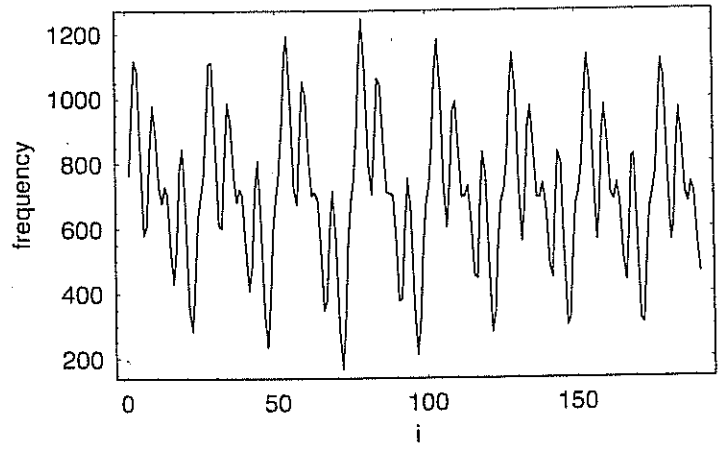


Fig. 13 The time series generated from the composition developed in Fig. 7.

Using a similar process as above, we can compute the mutual information and create a visualization. We found that the first minimum of the mutual information in this case is at $\tau = 3$. Note that the threshold we used for converting this data into a binary series is 707. Although the first minimum occurs at $\tau = 3$, we find that a more interesting visualization is made when we choose $\tau = 5$, the second minimum of the mutual information.

```
In[71]:= τ = 5;
         phase2Dt = Table[{songmath[[i]], songmath[[i + τ]]},
           {i, 1, Length[songmath] - τ}];

         ListPlot[phase2Dt, PlotStyle → PointSize[0.02],
          FrameLabel → {"s_i", "s_i+5"}];
```

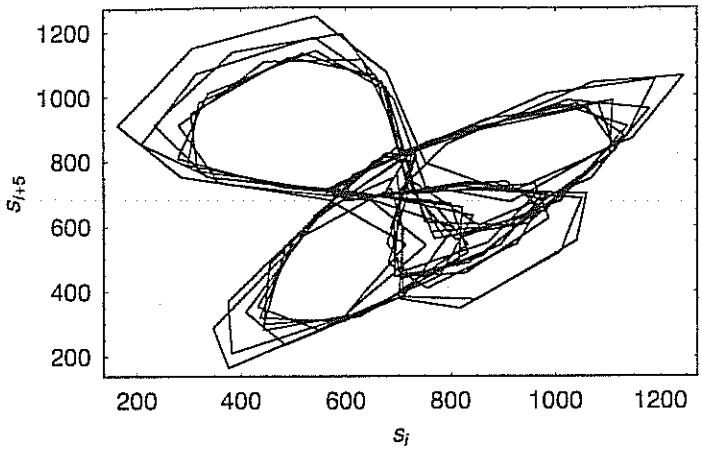


Fig. 14 A visualization of the composition from Fig. 7.

It is important to note that using the first minimum of the mutual information is simply a guide for choosing the delay. Choosing different delays is equivalent to choosing different coordinate systems for the visualization. Since the goal here is the creation of a visualization and not an approximation of the system's phase space, we can be flexible with our choice of delay.

3.3 Transfer entropy and piano music

Once a composition has been reduced to a time series, there are many calculations which can be performed to extract information about the structure of the composition. For example, piano music contains two staves. One staff is known as the *treble staff* (top staff) and is usually played by the right hand. The other is known as the *bass staff* (bottom staff) and is usually played by the left hand. Together, these are known as the *grand staff*. Usually, the melody is provided by the treble staff and the harmony is provided by the bass staff.

A natural question to ask is, "Can we demonstrate whether the notes played by one hand influence the notes played by the other?" This question can be answered by computing something known as the transfer entropy [12]:

$$T(B \to T) = \sum_{t_{n+1}, t_n, b_n} p(t_{i+1}, t_i, b_i) \log_2 \frac{p(t_{i+1}, t_i, b_i)p(t_i)}{p(t_i, b_i)p(t_{i+1}, t_i)}, \tag{4}$$

$$T(T \to B) = \sum_{b_{n+1}, t_n, b_n} p(b_{i+1}, t_i, b_i) \log_2 \frac{p(b_{i+1}, t_i, b_i)p(b_i)}{p(t_i, b_i)p(b_{i+1}, b_i)}. \tag{5}$$

The transfer entropy computes how much information is exchanged between two time series B and T and in which direction that information is exchanged. The first (second) equation gives the number of bits of predictability given to series $T(B)$, by series $B(T)$. Hence, if $T(B \to T)$ is large, then the series B greatly influences (or drives) series T. When computing the transfer entropy between time series, we must symbolize the series just like in the mutual information calculation. For more information on the transfer entropy, the interested reader is referred to [12].

We will compute the transfer entropies for part of Beethoven's *Fur Elise* (sheet music freely available at [13]). We generate the time series from the treble (T) and bass (B) staves as described in §2. We

then symbolize each series into a binary one using the threshold which maximizes the Shannon Entropy.

```
In[74]:= T = IntegerPart[Flatten[Import["furelisebinarytreb.xls"]]];
         B = IntegerPart[Flatten[Import["furelisebinarybass.xls"]]];
         TT = Table[{T[[i + 1]], T[[i]]}, {i, 1, Length[T] - 1}];
         BB = Table[{B[[i + 1]], B[[i]]}, {i, 1, Length[B] - 1}];
         TB = Table[{T[[i]], B[[i]]}, {i, 1, Length[T]}];
         TTB = Table[{T[[i + 1]], T[[i]], B[[i]]}, {i, 1, Length[T] - 1}];
         BTB = Table[{B[[i + 1]], T[[i]], B[[i]]}, {i, 1, Length[T] - 1}];
```

Next, we generate the probability functions:

```
In[81]:= Pt[z_] := Count[T, z] / Length[T];
         Pb[z_] := Count[B, z] / Length[B];
         Ptt[u_, v_] := Count[TT, {u, v}] / Length[TT];
         Pbb[u_, v_] := Count[BB, {u, v}] / Length[BB];
         Ptb[u_, v_] := Count[TB, {u, v}] / Length[TB];
         Pttb[u_, v_, w_] := Count[TTB, {u, v, w}] / Length[TTB];
         Pbtb[u_, v_, w_] := Count[BTB, {u, v, w}] / Length[BTB];
```

where **Pt** = $p(t_i)$, **Pb** = $p(b_i)$, **Ptt** = $p(t_{i+1}, t_i)$, **Pbb** = $p(b_{i+1}, b_i)$, **Ptb** = $p(t_i, b_i)$, **Pttb** = $p(t_{i+1}, t_i, b_i)$, and **Pbtb** = $p(b_{i+1}, t_i, b_i)$.

Next, we compute the transfer entropy in each direction.

```
In[88]:= Needs["Graphics`Graphics`"];

         (* T(B→T) *)
         Tbt = Sum[If[Pttb[i, j, k] == 0, 0, Pttb[i, j, k] *
             Log[2, (Pttb[i, j, k] * Pt[j]) / (Ptb[j, k] * Ptt[i, j])]],
           {i, 0, 1}, {j, 0, 1}, {k, 0, 1}];

         (* T(T→B) *)
         Ttb = Sum[If[Pbtb[i, j, k] == 0, 0, Pbtb[i, j, k] *
             Log[2, (Pbtb[i, j, k] * Pb[k]) / (Ptb[j, k] * Pbb[i, k])]],
           {i, 0, 1}, {j, 0, 1}, {k, 0, 1}];

         BarChart[{Tbt, Ttb}, BarLabels → {"T(B→T)", "T(T→B)"}];
```

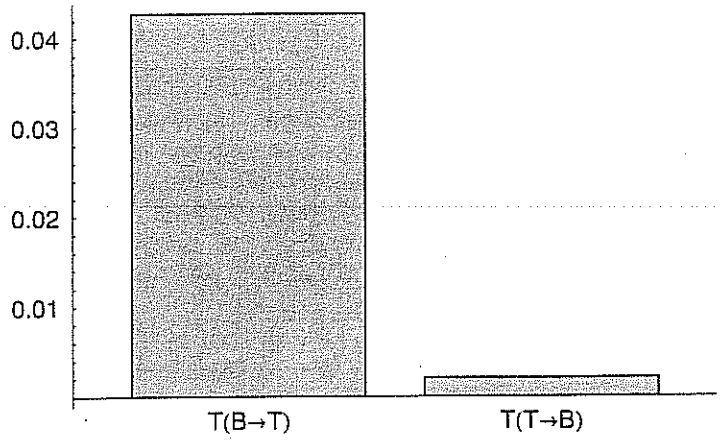


Fig. 15 A plot of the transfer entropies between the time series generated from the bass (B) and treble (T) staff of the first part of *Fur Elise*.

From the transfer entropy calculation, we can see that the bass influences the treble much more strongly than the treble influences the bass. In terms of predictability, the bass gives almost ten times as many bits to the treble as the treble does to the bass. While further studies are required, the transfer entropy may be one way to demonstrate how a composer's musical style changes with time. It may also serve as a means of identifying the composer or type of piano music.

As we can see, there are many different analyses that can be done to a composition once it has been converted to a time series. One of the future goals of this research is to identify means of classifying classical music by its composer. While we believe that we have a basic tool for converting the composition into a time series, we must continue to work to find the method of analysis which will serve as the classifier.

4. Conclusions

We have shown that a musical composition can be created using a curve. The composition can be created using either traditional means, as demonstrated in §1, or by using Mathematica, as demonstrated in §2. Further, musical compositions can be analyzed using the methods of nonlinear time series analysis in the *Mathematica* computing environment. Using these analytical methods, we can go beyond visualization and begin to study the structure of the music itself (e.g. the transfer entropy calculation of *Fur Elise*).

5. Acknowledgments

CWK would like to thank the College of Arts and Sciences of Eastern Kentucky University for support. CWKs part of this work was partially funded by the Junior Faculty Summer Research Award given annually by EKUs College of Arts and Sciences. Further, he would like to thank Prof. E. R. Tracy of The College of William and Mary for his helpful comments and suggestions. MM would like to thank Dr. Kathy Hill for her helpful suggestions. All of the authors would like to thank the reviewers for their many useful suggestions and feedback.

References

[1] T. Garland and C. Kahn, Math and Music, Dale Seymour Publishing, New Jersey, 1995.

[2] P. Bourke, "Butterfly Curve." Available online at astronomy.swin.edu.au/~pbourke/curves /butterfly/.

[3] Available online at people.eku.edu/machadom/.

[4] H. Kantz and T. Schreiber, Nonlinear Time Series Analysis, 2nd edn, Cambridge, London, 2004.

[5] H. D. I. Abarbanel, Analysis of Observed Chaotic Data, Springer, New York, 1996.

[6] G. Monro and J. Pressing, *Computer Music Journal* **22** (1998) 20.

[7] J. D. Reiss and M. B. Sandler, *Proc. of the 6th Intl. Conference on Digital Audio Effects* (DAFX-03), London, United Kingdom, September 8 - 11, 2003.

[8] A. M. Fraser and H. L. Swinney, *Phys. Rev.* A **33** (1986) 1134.

[9] C. E. Shannon, *The Bell System Technical Journal*, **27** (1948) 379.

[10] E. R. Tracy, private communication.

[11] C. S. Daw, C. E. A. Finney, and E. R. Tracy, *Review of Scientific Instruments* **74** (2003) 915.

[12] T. Schreiber, *Phys. Rev. Lett.* **85** (200) 461.

[13] Available online at www.sheetmusic1.com/NEW.GREAT.MUSIC.HTML.

About the authors

Christopher Kulp is an Assistant Professor of Physics at Eastern Kentucky University. Marianella Machado is a professional composer and Assistant Professor of Foreign Languages and Humanities at Eastern Kentucky University. Dirk Schlingmann is a Professor of Mathematics and Chair of the Department of Mathematics and Statistics at Eastern Kentucky University.

Christopher Kulp
Department of Physics and Astronomy
Moore 351
Eastern Kentucky University
Richmond, KY 40475
chris.kulp@eku.edu

Marianella Machado
Department of Foreign Languages and Humanities
Case Annex 368
Eastern Kentucky University
Richmond, KY 40475
marianella.machado@eku.edu

Dirk Schlingmann
Department of Mathematics and Statistics
Wallace 312
Eastern Kentucky University
Richmond, KY 40475
dirk.schlingmann@eku.edu