

Analysing Music for a String Performer

Bill Findlay & Cordelia Hall



Department of Computing Science
University of Glasgow
Glasgow G12 8QQ
Scotland

TR-1999-40
July 1999

For more information about Technical Reports
and Research at the

Department of Computing Science
University of Glasgow,

send mail to

reports @dcs.gla.ac.uk

or consult our Worldwide Web pages at

<http://www.dcs.gla.ac.uk/>

Technical Reports Coordinator
Department of Computing Science
University of Glasgow
Glasgow
G12 8QQ

phone +44 0141 330 4256
fax +44 0141 330 4913

Analysing Music For A String Performer

Bill Findlay and Cordelia Hall
Department of Computing Science
University of Glasgow
17 Lilybark Gardens
Glasgow, G12 8QQ
{wf,cvh}@dcs.gla.ac.uk

Abstract

String players know that bowing properly is the hardest skill they have to learn. In particular, a piece of music must be analysed very carefully if it is to tell them what part of the bow to use for each note. Such detailed knowledge is required if the player is to practice properly. This paper describes an algorithm that calculates the length and location of each bow stroke, given minimal guidance by the player. The algorithm has been tested on six substantial pieces of music, and seemed to work well in practice.

1 Introduction

String instruments such as the violin are hard to play at all, and very hard to play well. In general, the left hand selects the pitch to be played and the right hand contributes the sound of the note. Once the musician has learned to place the left hand fingers properly, it is easy to tell if the left hand is performing adequately. If it isn't, there are some straightforward things to practice, such as a clean shift from one position to another.

The right hand is much harder to control. Becoming good at bowing requires that the player think very carefully about the nature of a good sound and the particular movements that achieve that sound. After working hard to mentally articulate the motions to be used when playing at speed, the string player must practice so that these are used during performance. The problem is that there is virtually no way to denote these motions, which makes it hard to communicate them to others. Often a student will return home after a lesson in which the music being studied has been marked up with upbow (<) and downbow (I) markings describing the direction of the bow, but without

any way of recalling the interesting part of the lesson, which is *how* the teacher moved the bow.

A *bowing* is a sequence of symbols indicating movement away from the tip (an upbow) or away from the frog (a downbow). It cannot do more than represent changes in bow direction. It is up to the player to try out a variety of bowings, and experiment with various strokes. All of the hard won experience needed to develop a sequence of strokes that express a piece well is encapsulated so laconically in the available notation that it is hard to believe musicians do not forget much of what they achieve during practice. In fact, we suspect that many players do not consciously decide upon the strokes they use; they just hope that practice will solve the problem somehow.

The fact that bowing decisions are poorly articulated mentally means that both performers and students develop their bowing technique slowly. The student is usually unaware of the variety of choices available, and tends to develop restrictive habits that make expressive playing difficult. In particular, it is hard to move smoothly between the tip and the frog, as most people have trouble integrating the hand, wrist and arm movements that make up a 'good bow arm'. Carl Flesch observes [2], p. 93, that 'Diminished bow expenditure is a comfortable, yet in general an inferior means of securing a lesser degree of tone power because the carrying power of the tone is noticeably diminished thereby'. As the student progresses, he or she will encounter more encouragement to use as much bow as possible, since that produces a loud smooth sound that carries well in today's concert halls.

Professional players are required to be able to follow the bowings given their section heads in orchestras, and may even have to adjust to these bowings in real time. Some music editions are unreliable, and musicians sightreading from them must learn to look ahead and protect themselves. For example, the edition we used for the Bach double violin concerto in D minor had a bowing for the solo first violin part that simply ran out of bow on the second line. These days, musicians have something like one rehearsal before performing, and need as much help as they can get from the editors of the music they read. And soloists will often wish to create their own approach to their repertoire, which requires that they explore a large variety of bowings. So professional string players must learn to be in the habit of analysing.

There are some very successful schools of string teaching that bring

as much thought and planning to bear upon this problem as they do on understanding the components of a piece of music. The goal of the BowTech project is to support this work, so that musicians who aren't able to analyse their technique will receive some help.

In addition, this work can be applied to at least two other areas. One is that of editing string music, so that musicians can rely on their editions much more than they can afford to do today. Another goal is to process MIDI files so that a programmable synthesizer can produce sounds that are more like those of a real string instrument.

1.1 Related work

Much work has been done on modelling the behaviour of the violin and bow [5]. Askenfelt's work on understanding the bowing strategies of two professional violinists [1] concluded that the bow bridge distance was more important than the bow velocity. The conclusion of Hall, who was trained as a professional violinist, is that it can be useful to play near the bridge when a shorter bow is required, since that results in a sound with a well defined 'core' which also carries well, but that this cannot be the basis of a bowing technique because it requires too much preparation for the stroke (the bow has to be moved near the bridge, the speed reduced and the correct timbre has to be established by the right hand and arm). And more important, this technique does not take advantage of the powerful, free sound that results from using lots of bow at the midpoint between the bridge and the fingerboard. For this reason, our algorithm encourages the use of lots of bow.

There have been a number of well-known teachers, such as Dorothy DeLay at Juilliard, who have emphasised exercises aimed at making the player as conscious as possible of the portion of bow to be used. However, this is the first work we are aware of that uses the resources of a computer to do so.

2 The previous algorithm

This paper is based in part on our previous work [3], which used information provided by the musician to calculate whether a bowing resulted in a disaster (running out of bow at either the frog or the tip).

A variety of variables affect the sound produced by the bow. For example, the distance from the bridge affects the intensity of the sound, as do the weight or pressure brought to bear by the right arm. We have selected the amount of bow used as the only parameter which will be simulated. The reason for this is that we have found this parameter to be critically important in the development of a string player.

For the sake of simplicity, we assume that traditional time signatures are used by the pieces we analyse. We also want to be able to define a speed at which the simulation takes place. One of the algorithm's inputs is a metronome speed which is divided into 60. If the metronome speed is lower than 60, then the duration of the note(s) played will be longer, simulating a slower speed. If the metronome speed is higher, then the duration of the note will be shorter.

The basic ideas are simple. First, we propagate global and local information to each note. For example, a sequence of notes may appear in a section marked 'piano', so each note in that section is marked with a P. A bowing mark may appear, so we propagate the effect of that mark until either a rest or a new bowing mark appears. The result of these first passes is that each note now contains information about the context in which it appears at the time the music is being played.

Then the algorithm processes each note in turn, maintaining several variables that describe the state of the bow. The most important of these is the *bow position*, which records the point on the bow at which motion stopped after playing the current note. The length of the stroke assigned to the current note is a standard length, multiplied by several factors. These include the note's duration, a factor describing the dynamic and another describing the metronome marking of the music. The resulting value is then added to the current bow position if the bow is an upbow, and subtracted if it is a downbow. The tip of the bow is represented by zero, and the frog by one, so all bow positions are a fraction between those two boundaries. If the bow position ever crosses these boundaries, then the bowing is considered a 'disaster'.

The first algorithm [3] simulated the behaviour of the bow in this way, and stopped when a bowing was rejected. The main defect of this algorithm was that it did not help the user in dealing with difficult situations. It couldn't automatically deal with problems such as calculating the bow length for a mixture of long and short notes.

The program would flag these as disastrous because the long bows eventually consume the bow available, when the player would perform these passages by allocating less bow to the long notes and more to the short ones.

Another significant problem was that the program would get stuck at either the frog or the tip after negotiating a complex passage, causing a long sequence of sixteenth or eighth notes that followed to be played at one end of the bow. The player would gradually move to the middle of the bow, where the sequence would sound better and be more under the player's control.

3 The new algorithm

The new algorithm also maintains and propagates information about the state of the bow. However, it is designed to be independent of the user. In order to do this, it has to deal with the problems described above. The new algorithm divides up the music into two kinds of blocks. The first kind, called a *simple block*, is composed of notes with the same duration; the second kind, a *complex block*, contains only notes of mixed durations or very long notes.

For simple blocks, the algorithm takes advantage of the fact that they allow the player to make adjustments in the location of the bow. It makes the upbows longer, if the bow position is too close to the tip, and the downbows longer, if the bow position is too close to the frog. This means that the algorithm can work its way out of awkward positions and move back to the middle of the bow, just as the player would.

Complex blocks require some care because they are difficult to bow properly. The algorithm originally handled such a block by backtracking (trying all possibilities). If the first pass failed, then the note being bowed was made shorter, and if that failed, then it was made longer. If all of these failed, then the algorithm backed up to the previous note and tried a new alternative. Backtracking was effective at improving the end of a complex block, but the solutions found were not particularly good, and the task of assessing the value of a solution was incompatible with a backtracking algorithm.

The new algorithm calculates all possible bowings through a complex block, and then applies several functions which assess the value

of the bowing from a variety of angles. The three different scoring functions, *Average*, *Successive* and *Extremes*, produce a value from 1 to 100, where 100 is the worst and 1 the best, and the composite score assigned a bowing is the average scores of the three.

Any path which runs out of bow is given 100 without further assessment. The first function, *Average*, takes the average bow position, returning a high score if the average is close to the middle of the bow, and a lower score if it is closer to the tip or frog. While a useful indicator of the value of a bowing, the average may be misleading if there is a sequence of notes at one end followed by another sequence at the other end. For this reason, the second algorithm checks for successive strokes close to either the frog or the tip. If there are more than 4 strokes at the tip, or more than 2 at the frog, then a lower score is returned. Finally, bowings that require the bow to be at an extreme end of the bow make it more likely that the algorithm will run out of bow somewhere along the line, so the third algorithm assigns these lower scores.

4 Some details of the algorithm

The algorithm, which was implemented in Ada, has several key components, which have been described but will not be explained further for the sake of simplicity. The core of the algorithm is the *nextnote* procedure, which takes a record representing one note, the position of the previous note, and returns a result describing whether this note causes the algorithm to run out of bow.

Every note has a start state that tells the algorithm whether the musician has decided to override the calculated direction of the bow (for example, if this note occurs after a rest). The value *P_continue* states that the direction of the bow will be determined by the context of the previous note. The function *translate_pattern* converts the start point, expressed as an enumerated type, into a rational number.

```

procedure nextnote (this : in out music_unit_type;
                   old_bowpos : in rational;
                   result : out result_type) is
old : rational := old_bowpos;
begin
  if this.start /= p_continue then

```

```

    old := translate_pattern(this.start);
  end if;
  this.old_bowpos := old;

```

The length of the current note's stroke is calculated by multiplying a standard length *unitlen* times a metronome factor, the duration of the current note, a dynamics factor and an *epsilon* value which is assigned during the analysis of complex blocks and which may shrink the length of the stroke.

```

this.len := unitlen * (60 / metro) * this.duration
           * translate_dynamics (this.dynamics)
           * this.epsilon;

```

Next, the stroke length is extended if the note occurs in a simple block and the old bow position is in an awkward part of the bow.

```

if this.bowing = up and old < 2/5 and
   simple_block(this.block) then
  this.len := this.len + 1/20;
elsif this.bowing = dn and old > 3/5 and
   simple_block(this.block) then
  this.len := this.len + 1/20;
end if;

```

The stroke length is then added to the old bow position if the stroke is an upbow, and subtracted if it is a downbow.

```

if this.bowing = up then
  this.bowpos := old + this.len;
elsif this.bowing = dn then
  this.bowpos := old - this.len;
else raise nextbow_error;
end if;

```

The bow pattern tells the user the position of the bow after the current note, so this information is recorded for each note.

```

if this.bowpos >= zero and this.bowpos < 3/10 then
  this.pattern := tip;
elsif this.bowpos >= 3/10 and this.bowpos < 7/10 then

```

```

    this.pattern := mid;
elseif this.bowpos >= 7/10 and this.bowpos <= one then
    this.pattern := frog;
end if;

```

Finally, the bow position after playing the current note is examined, and if the algorithm runs out of bow in either direction, the result is reported.

```

if (this.bowpos > one or this.bowpos < zero) then
    result := disaster; else result := ok; end if;
end nextnote;

```

5 What happened when the algorithm was applied

We applied the algorithm to six pieces: the first, second and third movements of the Bach double violin concerto (solo violin 1), and three movements from Handel's sonatas for violin and figured bass (the first movement of the sonata in A major, the third movement of the sonata in D major, and the fourth movement of Handel's sonata in E major). All of these were assigned viable bowings; that is, none of the bowings run out of bow at some point during the analysis.

Each of these movements had some interesting problems for our algorithm to address. The algorithm assigned bowing annotations without stopping for direction by the musician (unlike our previous work). The musician provided music that was marked with dynamics and wrote in any slurs required for known technical reasons (e.g. the player would run out of bow here otherwise) or musical reasons.

In many cases, the algorithm found a reasonable solution. It marked each note with the words T(IP), M(ID) or F(ROG), depending upon the bow position after that note was played. These markings appear in the examples page at the end of the paper (excluding M markings, to reduce clutter). In our examples, some notes are marked S, indicating that the algorithm required the note to have two thirds of its original calculated length.

The following edited output from the program shows the bar number (fractions indicate position of a note within the bar), the old bow

position that was current for the previous note, the new bow position, the information provided to the musician, and the factor used by the analysis of a complex block to control the length of the stroke. The final column is the note duration as a fraction of the value receiving one beat (so .25 is a sixteenth note because the time signature is 4/4).

bar	old_bowpos	bowpos	pattern	epsilon	duration
10.44	0.49	0.54	MID	1.00	0.25
10.50	0.54	0.17	TIP	0.67	2.50
11.13	0.17	0.28	TIP	1.00	0.25
11.19	0.28	0.22	TIP	1.00	0.25
11.25	0.22	0.33	MID	1.00	0.25
11.31	0.33	0.27	TIP	1.00	0.25
11.38	0.27	0.38	MID	1.00	0.25
11.44	0.38	0.32	MID	1.00	0.25
11.50	0.32	0.69	MID	0.67	2.50

Here are some of our conclusions after applying the algorithm to our test pieces:

1. The first movement of the Bach was relatively straightforward. It had lots of blocks of sixteenth or eighth notes which allowed the algorithm to maintain the desired average bow position (the middle of the bow). In our example, a note appeared which was so long that it had to be handled in one of two ways. Either the bow had to be at the frog when it started, or the stroke had to be shortened. The algorithm shortened the duration of that stroke. The notes between the two half notes were few enough that the second half note was reached before the algorithm moved back to the middle of the bow. The precise analysis of part of this example is shown above.
2. The second movement of the Bach concerto was interesting because it contained a passage (see examples), which could not be played without shortening the long notes. This was duly discovered by the algorithm.
3. The first movement of the first Handel sonata in A major contained lots of tricky sections including the example, where sequences of four or five strokes were twice separated by longer notes that moved in the same direction. This was handled well

by the algorithm, but might have been more of a problem if there had been a longer sequence like this.

4. The third movement of the Handel sonata in D major was challenging because the tempo was so slow that long notes tended to require a whole bow, leaving little margin for adaptation, as in the example shown. The algorithm shortened these.
5. The fourth movement of the Handel sonata in E major was straightforward, even for uneven sequences like the one in the example. Interestingly, the algorithm marked a different passage to be played near the frog, then managed to move away to the middle of the bow. This occurred just after a long note left the player near the frog, and was the same as Hall's solution while performing the piece.
6. The third movement of the Bach had one tricky type of passage, in which two sets of four sixteenth notes are played, the last three in each slurred together, and then two slurred triplets. The algorithm got into dangerous waters at the end of a long passage of these and marked most strokes in the last bar with an F because they occurred in a mixture of simple and complex blocks which forced the algorithm to have a local and not a global view of the situation.

6 Conclusions

Our goal in designing this new algorithm was twofold. We wanted to reduce the burden of the user, so that less intervention was required. We also are interested in applying this work to other problems such as editing string music and improving synthesized string playing, e.g. melding the sound of a note played downbow with the usual synthesized sound.

This work is useful in that it is progress towards a more sophisticated, mostly automatic system called BowTech. It sometimes finds good solutions, in the sense that a musician can do no better. It is not a constraint solving system, so it may not do well when the context requires preparation of a stroke by the musician. However, this is something for a more sophisticated algorithm; we were surprised that our simple approach could do as well as it did.

There is a wide variety of further work that can be done on the BowTech algorithms.

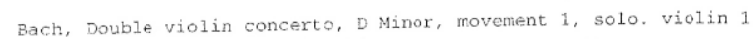
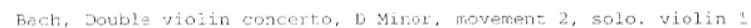
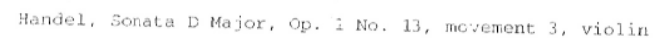
- We currently do not calculate the bow's distance from the bridge (although notes that are marked as short should be played closer to the bridge).
- Bow speed and bow weight (pressure) are also issues that should be addressed.
- More needs to be done on propagating as much context information as possible, so that a simple algorithm can look ahead and take advantage of it.
- We also would like to help the musician find out which notes must be slurred in order to escape bowing problems, and distinguish these from musical slurs. At present, musicians mix all of these together; we may someday help purists who would like to stick with the original bowing as much as possible.
- A movie that visualises the bow strokes made during a piece of music would be helpful for users, as they may be used to developing their technique using a mirror.
- We may also develop an interface in which the musician can hear the results of this work, by driving a sample playback synthesizer. While this would tend to make the string sound more realistic, the individual bowings would not usually be audible. However, if we can calculate bow articulation, then we can produce music played in the baroque style, which would be audibly different from the modern style.
- Lots of rhythmic patterns and their usual solution could be incorporated into the algorithm. This would allow the user to construct a specialised bowing technique that could be explicitly recalled, developed and taught.
- There is an interesting problem in specialising this work to the cello as opposed to the violin.

7 Acknowledgements

John O'Donnell suggested that this problem might be similar to that of text typesetting and that Knuth's solution [4], which was to assign

12

- [1] Askenfelt, A., *Measurement of the bowing parameters in violin playing. II: Bow-bridge distance, dynamic range, and limits of bow force*, J. Acoust. Soc. Am. 86(2), 503-516.
- [2] Flesch, C., *The Art of Violin Playing*, Bk. 1, Carl Fischer, Inc. 1939.
- [3] *Simulation of Bowing Decisions on a String Instrument*, JIM'97 (Journées d'Informatique Musicale), Lyon, France, June 6-7, 1997.
- [4] Knuth, D. E., *The TeXbook*, Addison Wesley, 1990
- [5] Schelleng, J., *The bowed string and the player*, J. Acoust. Soc. Am. 53(1), 26-41.



Handel, Sonata A Major, Op. 1 No. 3, movement 1, violin

Handel, Sonata D Major, Op. 1 No. 13, movement 3, violin

Handel, Sonata E Major, Op. 1 No. 15, movement 4, violin

Bach, Double violin concerto, D Minor, movement 2, solo. violin 1

Bach, Double violin concerto, D Minor, movement 1, solo. violin 1