The Interpretation of MIDI Velocity

Roger B. Dannenberg

School of Computer Science, Carnegie Mellon University
dannenberg@cs.cmu.edu

Abstract

The MIDI standard does not specify how MIDI key velocity is to be interpreted. Of course, individual synthetic instruments respond differently, but one would expect that on average, instruments will respond about the same. This study aims to determine empirically how hardware and software MIDI synthesizers translate velocity to peak RMS amplitude. Analysis shows synthesizers roughly follow an x-squared rather than exponential mapping. Given a desired dynamic range (from velocity 1 to 127), a square-law mapping from velocity to RMS is uniquely determined, making dynamic range a convenient way to summarize behavior. Surprisingly, computed values of dynamic range for commercial synthesizers vary by more than 60dB.

1 Technical Introduction

MIDI key velocity (Rothstein 1995) is normally an indication of dynamic level or loudness, but the MIDI standard (MMA 1996) does not specify exactly how velocity should be interpreted. In synthesizers, key velocity can control many parameters, including amplitude, FM modulation depth, and sample selection. Even when velocity is used simply to scale audio amplitude, it is unclear how to map MIDI velocity to amplitude.

In order to create a truly "MIDI compatible" system, one should try to be consistent with existing implementations so that similar key velocity values result in similar output levels. Lacking any published recommendations or specifications, I measured many programs (instruments) on a handful of synthesizers to determine how key velocity maps to peak RMS amplitude.

At the outset of this work, I assumed that MIDI velocity would be logarithmically related to amplitude since it is well known that perceived loudness is also quasi-logarithmic. A logarithmic scale would allow a wide dynamic range to be represented efficiently by the 7-bit velocity value in MIDI messages. One finding is that a logarithmic relationship is *not* a good fit to a variety of commercial synthesizers and patches. Overall, a square-root function is a better, and in some cases nearly exact, model of MIDI velocity as a function of RMS amplitude. However, synthesizers and programs appear to be quite inconsistent. We can make recommendations for MIDI-controlled instruments, but there is hardly a de facto standard.

Originally published as: Roger B. Dannenberg, "The Interpretation of MIDI Velocity," in *Proceedings of the 2006 International Computer Music Conference,* San Francisco, CA: The International Computer Music Association, 2006, pp. 193-196.

The next section expands upon the motivation as well as some musical and esthetic concerns relating to this work. Section 3 describes what I measure to study velocity, and Section 4 describes *how* I measure it. Section 5 introduces a model for sound variation as a function of velocity, and Section 6 fits this model to actual synthesizers. This is followed by some discussion and conclusions.

2 A Broader Introduction

MIDI seems to be a permanent fixture on the computer music landscape. It has been used for more than twenty years almost without change, and it has survived a major transition from serving as a real-time hardware control protocol to a data format for music that is realized entirely in software. MIDI is even used to specify ring tones for cell phones, possibly the largest application of music synthesis technology to date.

MIDI is not without shortcomings, and the limitations of MIDI have been lamented by more than one author. (Moore 1988, Wessel and Wright 2002) Various proposals to extend or replace MIDI have also appeared. (McMillen 1994, Wright 1997) Nevertheless, MIDI has proven to be resilient, durable, and well understood. The perceived benefits of MIDI compatibility usually outweigh any implementation difficulty, so most computer music systems handle MIDI messages and standard MIDI files.

One of the features of MIDI is that it supports the notion of the music score (or sequence of MIDI messages) as an abstract specification that can be "performed" by a variety of synthesizers. This has roots in Western music, common practice notation, and music performance practice – music scores can be played by different performers using different instrumentation.

Many computer musicians have rejected this notion outright, replacing vague sequences of MIDI messages with precise specifications of the entire sound production process using software synthesis. In this approach, the "instrument," its control, and even the notion of score are often integrated into a single software program, patch, or configuration, giving precise control of sound from human gesture all the way down to the details of sample-by-sample computation.

Now that we have experienced music making with MIDI-based systems and more general software-based systems, we can observe that these two approaches naturally

lead composers in very different directions. In particular, we hear a much stronger orientation toward "notes" and "sound events" with MIDI, and much more effects processing and continuous sound transformation with software systems. Secondly, MIDI-based systems, by standardizing on a common representation for performance information, invite experimentation with different instrumentation.

Because software allows us to define our own instruments, there is a tendency to treat MIDI data as arbitrary. A note-on message can just as well control a digital audio effect as play a note, and parameters such as pitch and velocity can be treated as arbitrary bits of information rather than specific meaningful parameters. However, creative practice often benefits from the juxtaposition of different devices, concepts, and technologies. Respecting some conventions for MIDI semantics in MIDI controllers, MIDI control software, and MIDI synthesizers simplifies experimentation and creative combination.

Conventions are almost necessary as a starting point in the infinitely malleable world of digital media. Therefore, it seems helpful to understand how one might handle and interpret MIDI velocity in a conventional way.

3 What to Measure

Velocity literally refers to the rate at which a keyboard controller key is pressed. The focus in this paper is how synthesizers respond to this parameter. But how can we measure and compare a complex audio response in a simple, objective way? One approach is to use psycho-perceptual methods, using humans to judge and compare differences between two synthesizers. (Martens 1985) These methods allow us to measure *perceptual* features and to calibrate control parameters accordingly. A new calibration would be necessary for each new synthetic instrument.

An interesting alternative is to pursue *models* of perception, in particular loudness models (Zwicker and Fastl 1990), so that velocity could be translated automatically to appropriate control values that produce the desired loudness level as detected by the model. Unfortunately, rather sophisticated models are required to deal with the time-evolution of sounds. For example, how does one compare an instrument with an intense attack followed by a soft sustain to one with a medium but sustained intensity?

I have chosen to ignore a number of thorny perceptual issues by considering only the maximum of the evolving short-term RMS envelope as a function of MIDI velocity. The problem with this measure is that two sounds with the same peak RMS value may not exhibit the same perceptual loudness. On the other hand, this approach is objective and simple to implement.

To measure the peak RMS value of a synthesized tone start with audio sampled at 44100 Hz, and process non-overlapping windows of 1050 samples:

$$W_{i,j} = x[i + 1050j] \tag{1}$$

From these windows, compute the RMS value as follows:

$$rms_j = \sqrt{\sum_{i=0}^{1049} W_{i,j}^2 / 1050} \tag{2}$$

The sequence rms_j, with a sample rate of 42 Hz, describes an envelope. The peak RMS value is simply the maximum:

$$peak = \max_j rms_j \tag{3}$$

We are interested in how this peak varies with MIDI velocity, so consider $peak_v$, the peak RMS value resulting from a MIDI velocity of v (for now, assume MIDI key number and other parameters are constant). After normalizing by $peak_{127}$, the peak observed with the velocity value at the maximum of 127, we obtain:

$$f(v) = peak_v / peak_{127} \tag{4}$$

We want to find a function f that is characteristic of many synthesizers and programs. If one exists, we can claim that f is a de facto standard for the interpretation of MIDI velocity.

4 Implementation: Peak RMS

To estimate $f(v)$ for a variety of instruments, I use an automated procedure. First a standard MIDI file is generated to play sequences of MIDI notes on middle-C with 15 equally spaced velocity values: 1, 10, 19, … 118, 127. Notes are 0.3 s long and a new note starts every 0.5 s. (It was discovered that reverb tails on some patches are long enough to cause some overlap, so wider note spacing should have been used.) A 15-note sequence is generated using every MIDI program from 0 to 127 for a total of 1920 notes (only 32 programs were used for the DX7 synthesizer).

This sequence is synthesized and recorded. A sequence of RMS amplitudes is computed. Knowing the starting times and durations of notes from the MIDI file, it is simple to scan the RMS amplitudes, computing a peak RMS value for each note. There are now 128 peak RMS values for each velocity, one for each MIDI program. Call these values $p_{v,i}$.

Next, the peak RMS values for each velocity are normalized by the peak at velocity 100 (the choice of this point is arbitrary) and averaged across all programs:

$$\bar{p}_v = \sum_i \frac{p_{v,i}}{p_{100,i}} \times \frac{1}{128} \tag{5}$$

The normalization step insures that we measure *relative* changes in peak RMS amplitude as a function of velocity (rather than absolute RMS amplitude), and the averaging step summarizes the variation across different programs.

Figure 1 shows a typical result, this one from a Roland Sound Canvas software synthesizer.

5 Analysis

One might expect amplitude to vary exponentially with velocity, i.e. velocity is logarithmic. Figure 2 plots the log of Peak RMS vs. Velocity for the Roland Sound Canvas

Synthesizer. The result should be a straight line if the logarithmic relationship holds. Figure 2 also plots the square root of amplitude as a function of velocity. This is much closer to a straight line, leading to the equation:

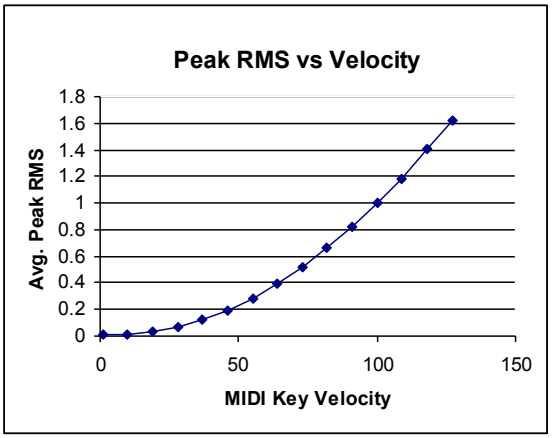

Figure 1. Peak RMS, normalized to 1.0 at MIDI Velocity 100, then averaged over all 128 programs and plotted as a function of velocity (data from Roland Sound Canvas software synthesizer).

$$a = f(v) = (mv + b)^2 \quad (6)$$

where a is the peak RMS value, v is velocity, and (m, b) are coefficients for the straight line model. Given a set of measured data points, a least-squares linear regression can be performed (using square roots of peak RMS values rather than the original values) to determine (m, b).

While m and b are not very intuitive, we can derive a more meaningful parameter: the dynamic range from velocity 1 to velocity 127. Simply plugging in numbers, and letting r be the dynamic range:

$$r = (m \cdot 127 + b)^2 / (m \cdot 1 + b)^2 \quad (7)$$

or in decibels:

$$r_{dB} = 20\log((m \times 127 + b)^2 / (m \times 1 + b)^2) \quad (8)$$

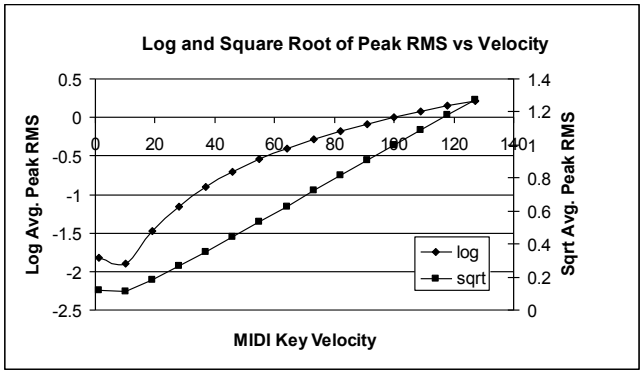

Figure 2. Log and Square Root of Peak RMS data from Figure 1, indicating that the square-root function is a better approximation than a log (decibel) scale.

Note that although (m, b) appear to be two independent parameters to this model, the data is normalized so that a=1 where v=100. This eliminates one degree of freedom. Assuming that Equation 6 holds, the dynamic range completely characterizes f. Given a desired dynamic range r_{dB}, we can solve for m and b to obtain:

$$r = 10^{r_{dB}/20} \quad (9)$$

$$b = 127/(126\sqrt{r}) - 1/126 \quad (10)$$

$$m = (1 - b)/127 \quad (11)$$

The dynamic range is an interesting and intuitive property that can be used to compare different synthesizers.

6 Results

I analyzed 7 synthesizers: the Roland Sound Canvas software synthesizer, the Microsoft GS software synthesizer, a Yamaha DX7 and SY22, a Roland U220, a Kurzweil K2000R with Kurzweil orchestral samples, and the Garritan Personal Orchestra Finale Edition (included with Finale 2006). All of these are based on sampling synthesis except for the DX7, which uses FM, and the SY22, which uses a mix of FM and sampling. The dynamic range for each of these is shown in column 2 of Table 1 and graphs of f as measured from audio are shown in Figure 3.

Table 1. Synthesizers and measured average dynamic range (characterizing response to MIDI Key Velocity).

Synthesizer	Dynamic Range	Dynamic Range (Pn)
Roland Sound Canvas	89	45
Microsoft GS	61	81
Yamaha SY22	21	18
Yamaha DX7	11	15
Roland U220	20	51
Kurzweil K2000R	25	37
Garritan Pers. Orch.	44	105

One could argue that averaging measurements from many instruments is a bad idea because instruments are intentionally designed with different velocity curves. The analysis was applied to a single piano sound on each synthesizer to obtain the third column of Table 1. Far from being consistent, the different piano implementations show an even wider variation in dynamic range.

In all synthesizers but the K2000R, the velocity is approximately linearly related to the square root of the peak RMS value. This relationship is much more exact and consistent among the software synthesizers. The hardware synthesizes (SY22, DX7, U220) seem to exhibit much more variation among different programs and overall, a much lower average dynamic range. Particularly for the DX7 (using 32 factory presets), it seems that velocity has a great effect on timbre through FM modulation depth and a lesser effect on amplitude.

The K2000R peak RMS value more closely fits the log of the velocity than the square root, both in the average over 128 programs and considering just the grand piano program.

In Figure 3, there is an apparent anomaly in that amplitude seems to *increase* at the lowest velocity levels. This is due to some long decays and overlapping reverberation tails from the previous note when the synthesizer was recorded. These first two points of each curve were ignored in subsequent calculations.

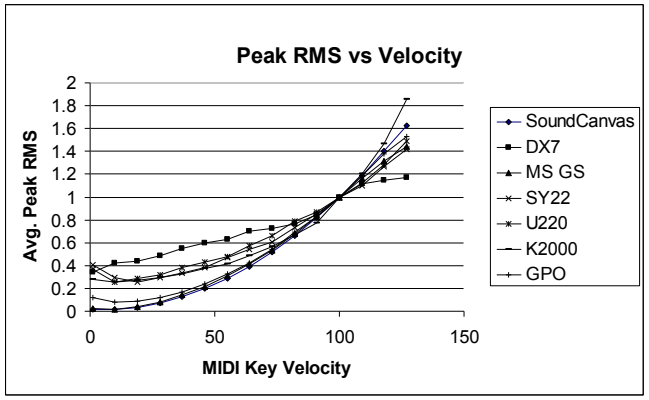

Figure 3. Peak RMS vs. Velocity for various synthesizers.

While the software synthesizers show much more conformity to Equation 1, they have decidedly different dynamic ranges (44dB vs 89dB, see Table 1).

7 Discussion

One surprising result of this work is the very great differences among different synthesizers. Presumably, all of these synthesizers are intended to perform MIDI data from keyboard performances, either directly or from recorded MIDI sequences. Why would there be so much variation? One might assume that the low dynamic range of the DX7 (11 dB) was designed to mask the noisy 12-bit DAC or that timbral variation via FM takes the place of amplitude variation. But the SY22 is much quieter and relies more on sampling, yet it also has a low dynamic range (21 dB). Note that *dynamic range* is used here specifically to describe the variation of average peak RMS with MIDI key velocity (r_{dB} in Equation 8), and this does not imply anything about the signal-to-noise ratio, DAC's, or other components. Another observation is that the three software synthesizers have a larger dynamic range that the four hardware synthesizers.

Based on these observations, how should a new software synthesizer respond to MIDI velocity? For compatibility with most existing synthesizers and MIDI files, the peak RMS value should be related roughly to the square of velocity as in Equation 6. Given this equation, one can choose the dynamic range from softest to loudest. Values from 20 to 60dB seem to be typical. In the synthesizers studied, the dynamic range varies from one instrument to another, even on the same synthesizer. From the standpoint of making instruments more interchangeable, it seems advisable to give all instruments a similar dynamic range. Once a range, r_{dB}, is chosen, Equations 9-11 can be used to compute the coefficients m and b for use in Equation 6, which computes a linear scale factor given a MIDI key velocity.

8 Summary and Conclusions

This work began with the desire to be consistent with commercial MIDI synthesizers. To do this, I measured how synthesizers handle MIDI velocity, and I hoped to adopt consistent mappings in my own software. To measure the effect of velocity on synthesizer output, I compute the peak RMS value of the amplitude envelope of a short (0.3s) middle C. I was surprised to find a rather wide range of interpretations for MIDI velocity.

One fairly consistent trend, however, is that velocity is linearly related to the square-root of the peak. Assuming this relationship to hold, and if velocities are normalized, then there is only one parameter left to choose. A convenient form of that parameter is the dynamic range from velocity 1 to velocity 127.

Given the lack of a standard or even a suggestion within the MIDI specifications, we are free to choose as we please. My suggestion is to adopt a dynamic range of 60dB (a nice round number, and about 1000:1 in linear terms). Consistent interpretation will make software instruments easier to use by being more predictable and more interchangeable.

I would be happy to share software developed in this project so that others can easily gather data and characterize other synthesizers.

References

McMillen, K. 1994. "ZIPI: Origins and Motivations." *Computer Music Journal* 18(4):47-51.

Martens, W. L. 1985. "PALETTE: An Environment for Developing an Individualized Set of Psychophysically Scaled Timbres." *Proceedings of the 1985 International Computer Music Conference*. International Computer Music Association, pp. 355-365.

Midi Manufacturers Association. 1996. The Complete MIDI 1.0 Detailed Specification. Los Angeles, CA: The MIDI Manufacturers Association.

Moore, F. R. 1988. "The Dysfunction of MIDI." *Computer Music Journal* 12(1), 19-28.

Rothstein, J. 1995. *Midi: A Comprehensive Introduction*, 2nd ed. Madison, WI: A-R Editions.

Wessel, D., and M. Wright. 2002. "Problems and Prospects for Intimate Musical Control of Computers." *Computer Music Journal* 26(3):11-22.

Wright, M., and A. Freed. 1997. "Open Sound Control: A New Protocol for Communicating with Sound Synthesizers." *Proceedings of the 1997 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 101-104.

Zwicker, E., and H. Fastl. 1990. *Psychoacoustics: Facts and Models*. Springer.