Revista Vórtex | Vortex Music Journal | ISSN 2317-9937 | http://vortex.unespar.edu.br/
D.O.I.: https://doi.org/10.33871/23179937.2021.9.2.13

What Pd Can't Teach; What It Can

Joshua Hudelson

Lebanon

Abstract: Pure Data's long appeal is partly owed to its visual layout and intuitive mode of use. This might lead one to consider it a useful pedagogical tool, even if it now meets stiff competition from glitzier and more user-friendly applications. In the author's experience, however, Pd's ambiguous location on the spectrum of virtualization makes it bad for teaching but gives it a far more important function for composers. In occupying an uncanny valley between the digital-as-digital and the digital-as-analog, it resists the general tendency of digital technology to obscure awareness of itself, that is, of the digital computer as the current high watermark of a longer history of rationalization. It thus pushes composers of computer music back towards the questions that are at the heart of their work. This dynamic is illustrated by the author's experience of developing and tinkering with a novel technique of sound synthesis.

Keywords: pedagogy, digital, analog, rationalization, statistical feedback

Received on July 1st, 2021. Approved on: November 1st, 2021. Available online on: 30/11/2021. Guest Editor: Dr. Alexandre Torres Porres.

In the fall of 2012, my classmate Joe and I crossed the street from the NYU Department of Music to the Steinhardt School of Culture, Education, and Human Development, where I'd arranged to give a presentation on Pure Data to a group of education scholars. We thought Pd had great potential as a pedagogical tool, and we were curious to know if education scholars would think the same. More than a signal processing platform, Pd struck us as a limpid, conceptual tool for exploring everything from music to mathematics. The white sandbox of the patching window seemed certain to appeal to students as an intuitively playful space—a bright, open space in which making connections between different objects would lead to pleasure, creativity, and ultimately understanding. Best of all, it was free.

The group at Steinhardt, while encouraging, was somewhat less enamored with Pd than we were. They didn't buy the idea that its graphical layout was inherently easier to understand than text. Why would a box with one outlet and two inlets—only one of which triggered anything to compute—be any more intuitive than an old fashioned plus sign? They noted that there was abundant research being done on the role of technology in pedagogy—had we looked at any of that? They gave us a list of people, projects, and books to check out, all of which I promptly forgot about, still thoroughly convinced of Pd's unique explanatory power.

Several years later, as a visiting professor at the American University of Beirut, I had the opportunity to test my convictions. In a class titled "Digital Sound and Music," I showed my students how concepts from music, psychoacoustics, and the physics of sound could be illustrated with Pd's boxes and lines. What struck me as success in the early weeks of class, however, soon turned out to be an illusion. Yes, my students liked Pd—but mainly because *I* liked it. I appeared to them as an amusingly nerdy American who took nostalgic pleasure in a relatively drab and abstruse piece of software from the 1990s. Beyond that, however, Pd did not strike them as a particularly clarifying tool. Even my computer science majors struggled to rewire their minds for its unbecoming, pixellated thickets. By the end of the semester, my students had done an admirable job of learning Pd, ultimately performing a collaboratively-composed tribute to the Beirut soundscape at a public concert. But they managed this in spite of the software, not because of it. I finally had to grapple with the fact that Pd was not the pedagogical panacea I'd wanted it to be.

Of course, Pd was never meant to be that. As Miller Puckette has noted, it was not even intended to be the open-source alter-ego to Max/MSP that is its reputation today. Originally, the

exciting graphical component of Pd was the now little-used “struct” object, with which composers were meant to sonify two-dimensional shapes (PUCKETTE, 2020). The Max data-flow paradigm was only added for the sake of functionality. By the time I was teaching in Beirut, Max/MSP's sleek graphics made it nearly a different species of program. All of which raises the question of what I thought was so great about Pd in the first place. What had it taught me that I thought was so important to teach others? To answer this question, a short detour is necessary.

1. An Analog of the Digital

Before I'd ever heard of Pd, I was making bad electronic dance music on digital audio workstations. I can still call to mind the bouncing motion made by the cables that linked different pieces of virtual studio gear, as though they were really rubber-coated and beholden to the gravity and Hookes's Law. The sensuous qualities of such workstations have only increased since then, with ever more precise—which is to say, precisely imprecise—virtualizations of physical equipment appearing on the market each year.

It's not uncommon to hear this tendency towards virtualization criticized from the position of expertise. According to a certain type of personality (myself all too often included) the more approachable and user-friendly the tool, the less serious the composer using it must be. A better critique, however, turns an eye to the ideological underpinnings of virtualization. The apparent physicality of a device not only makes it more approachable, it also lends it an aura that belies the machinic processes actually at work. The three-dimensional space of the gear rack, the textured surfaces of the gear, the inexactitude of its knobs, the springiness of the cables—all of these serve to bolster the impression that whatever comes out of the speakers was not born on a Procrustean bed of discrete computation. Instead, the sound must have its source in an open space of play and possibility. This distinction is made all the more drastic by the fact that much of the gear that DAWs virtualize was originally “analog”, connoting a similar warmth, humanity, and openness when set against the digital.

According to Alexander Galloway, the analog in its broadest philosophical sense denotes hybridity: the interaction of heterogeneous entities (GALLOWAY, 2014, p. 103). This is perhaps most famously characterized by Gilles Deleuze's example of the orchid and the wasp, whose

interactions are not prescribed by a rigid evolutionary explanation so much as unfurled via an open-ended co-becoming that is possible between entities (DELEUZE and GUATTARI, 2004, p. 11). Analog electrical technology can be read along these lines. It produces errors (sometimes quite fortuitous ones) rooted in the immanent heterogeneity of its physical components, and it is susceptible to interference and interaction via a host of outside forces, from vagrant electromagnetic waves to whatever object manages to plug into it.

Of course, the debates surrounding the digital and analog are too plentiful to rehash here, so the reader will have to suffer with my personal and contextually narrow intervention. Having spent time in a Masters program that was changing its name from “Electro-Acoustic Music” to "Digital Musics" (yes, a plural abstract noun), I have long been interested in what, exactly defines the shift to the digital. Like some, I would argue that there is no reliable split between the analog and the digital, at least not when the scope is narrowed from Galloway's grand, ontological vision to the realm of technological devices. A circuit that uses transistors and a circuit that does not are, within this view, the same type of technology. But, unlike many who would agree with this point, I do not ground it on the fact that both technologies are physically comparable (i.e., they both move electrons through conductive materials). Rather, analog and digital technologies share in a historically contiguous and conceptually overlapping advancement of the rationalization of material world for the purpose of computation. The same epistemological decision precedes both: that the world be rendered, more or less completely, in the form of measurements. And these measurements have some level of tolerance, resolution, or margin of error. Indeed, the presence of some degree in analog technology is not what distinguishes it from digital technology, where error gets decimated many orders of magnitude. Instead, oddly, it is what links the two. The basic elements of analog circuits—resistors, capacitors, inductors, and so on—were hardly designed to be capacious in their affordances or fickle in their responses. Viewed without the tinted glasses of digital-era nostalgia, analog technology was another step in a long history of winnowing the uncertainty of the material world, getting rid of the very promiscuity that is now its claim to superiority. One might even argue that the rise in popularity of stochastic processes, error, and interference in 20th century music—and especially now, in the boutique market of vintage analog devices—occurs in direct proportion to the threat that this process of rationalization poses to established ideas about composers, the act of

composition, and the supposedly open field of human creativity. Sources of randomness, in other words, offers relief.

Returning to virtual representations of physical gear, we can now see that contemporary DAWs attempt to leverage the perceived warmth of the analog in order to obfuscate the more general thrust of computational technology.[1] Pd, then, occupies a strange midpoint: an uncanny valley between the digital-as-analog and the digital-as-digital. In failing to choose a side, it illuminates something about the terrain on which composers of computer music find themselves today. Of course, technically speaking, Pd is entirely digital. But its hybrid origins reveal themselves at every turn. Among its objects, one finds a set of strictly digital, bitwise operations but also a "voltage-controlled filter", which references the realm of analog computing. The inlets and outlets of the objects, too, hark to modular synthesis. And the digital/analog divide is strengthened further by the distinction between thin "message" wires whose values are easy to interrogate and thick "signal" wires whose contents are better concealed.[2] The objects themselves, however, feel more scriptural than material; the need to type out their names to instantiate them is a vestige of command line programming. Further, the objects appear to have hidden interiors similar to devices in the analog world. That Pd is based around patching is already, of course, a reference to the patchbays of analog computers and synthesizers. And the open field of the patching window would seem to offer the same aura of seemingly infinite possibilities as the patch-able DAW I described above. And yet the space is also sterile and diagrammatic, suggesting a mode of thought more rigid than intuitive. At times, the digital asserts itself against intuition—for example, when the right-to-left processing of objects and inlets disrupts the at-all-once flow intended for a system.

In his reflections on Max, Puckette suggests that there are aspects of the patching layout that are loaded in terms of Western musical cultural influence—namely, the focus on paper and writing as essential features of music-making (PUCKETTE, 2002, p. 39). I would suggest, however, that Pd's far more important ideological work lies in how it both conjures up and troubles the rationalizing force of the digital. It returns us again and again to the fact that, as composers of

[1]It's important to note a similar obfuscation at the extreme opposite pole. The figure of the clear-eyed, or perhaps genius, computer programmer performs a similarly Romantic role of "ghost in the machine" for the digital computer.
[2]This distinction can be traced back to Max Mathews 1963 paper outline the theory of the MUSIC-N language. (MATHEWS, 1963, p. 555, figure 2).

computer music, we must either contend with the nature of the digital or persist in deceiving ourselves about what our work is. In the following section, I will use an example from my experience programming a Pd external to illustrate how this effect plays out in practice.

2. Feeding-back Statistical Feedback

I got the idea for the "statfeed~" object during a lecture by the composer Larry Polansky at the Workshop in Algorithmic Computer Music (WACM) at UC Santa Cruz in 2014. Polansky was speaking about "statistical feedback", a heuristic for modulating the degree of randomness in a sequence generated from some finite number of elements. It had been the go-to method for stochastic choices in the works of James Tenney, who had been one of Polansky's teachers and close friends.[3] And Tenney, himself, had grounded it in the work of an earlier generation of American composers of atonal music and their concept of "dissonant counterpoint" (Polansky, Barnett, and Winter; 2002, p. 4). Charles Ames also worked on the heuristic, which he deemed "a plausible emulation of how living composers dealt with balances before serialism was introduced" (AMES, 1995, p. 38).

What, exactly, is statistical feedback? At its simplest, it's a process by which a set of probabilistic weights corresponding to a set of musical elements (notes, for example) is updated each time an element is selected. Specifically, the selection of an element causes its probabilistic weight to be reduced to zero, while the weights of the other elements grow by some amount. In practice, the longer an element remains unchosen, the higher its likelihood of being chosen next. The growth of the weights of unchosen elements need not be linear, however. Tenney, for example, often used a growth function of c^x, in which x was the number of times the element had gone unchosen. At c=1, the weights of all elements remain equal regardless of time, resulting in purely random selections. But at values of c greater than 1, the differences between the weights of unchosen elements becomes steeper and steeper as time passes. Ultimately, the higher the value of c,

[3]See, for example, the interleaving note and register selections in his composition *To Weave: A Meditation* as discussed by Polansky. (POLANSKY et al, 2002, p. 25).

the more fixed and periodic the sequence of element choices, as any given element will not be chosen a second time before all of the others have been chosen first.

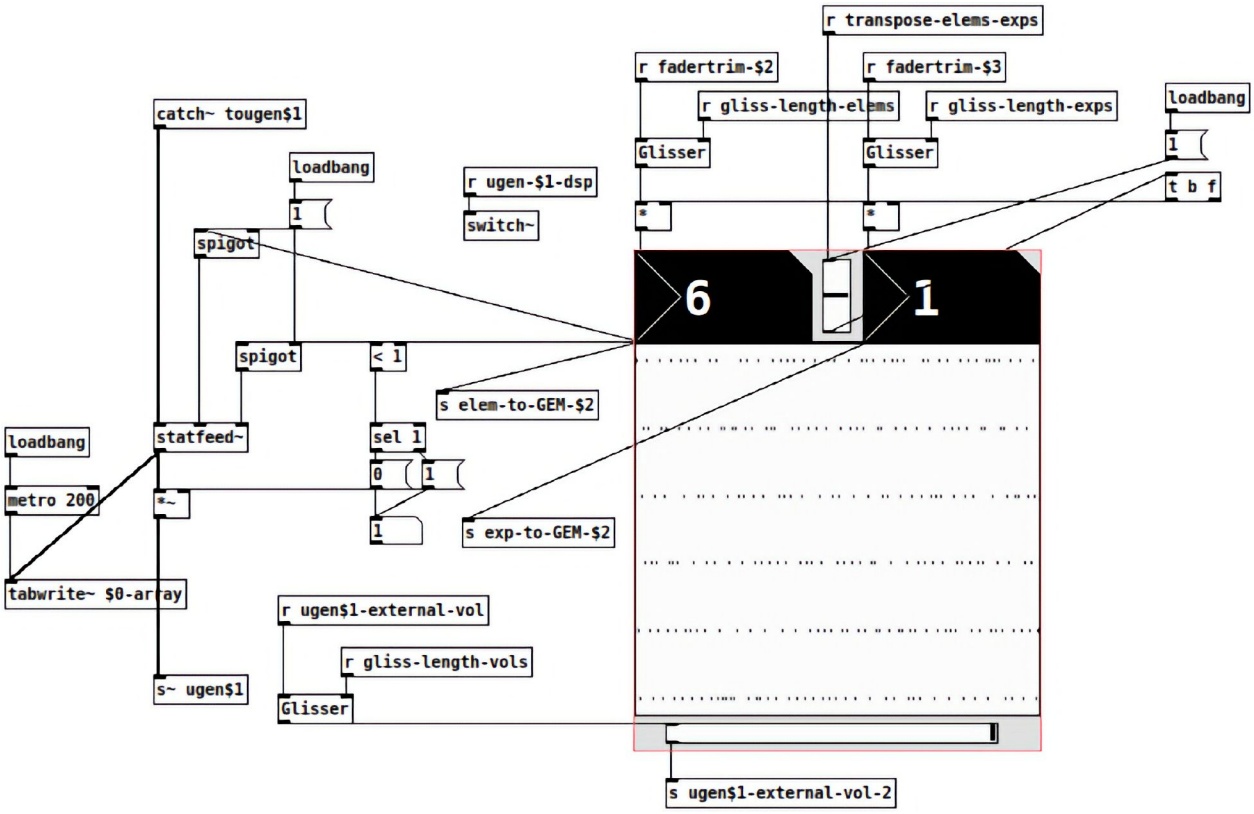
Knowing that a sequence of random sample values in an audio recording produces white noise, while a sequence of repeating values yields a pitched tone, I realized that running the statistical feedback algorithm at audio rate and tweaking the c-value would allow me to slide smoothly between poles of harmonic order and disorder. The project set me down a more tortuous path than I'd expected. Because WACM students were supposed to work in Lisp and MIDI, I finagled a low-quality form of audio synthesis by modulating the amplitudes of rapid-fire xylophone onsets. The result was rough, but promising. As expected, the value of c could be used to move between noise and tone, with pitch dependent on the number of possible elements (that is, the number of discrete values between -1 and 1 that could be assigned to a given sample). Timbre varied according to the particular sequence of elements and their degree of stability over time.[4] Eventually, the WACM director let me switch from Lisp to C, and I wrote a Pd external to explore the parameters further in real time.

The code for the Pd external differed from the one in Lisp, and not simply because Lisp and C are different. In the Lisp code, I had used native random number generation to compute each sample value selection based on the system's probabilistic weights. The same was, of course, possible to do in C, but Pd's native [noise~] object and its orientation toward decoupling and modularity suggested an alternative method. Instead of reproducing the Lisp code exactly, I built what was essentially a “transfer function” that maintains the probabilities of its elements as a cumulative list and updates itself after processing each value of a signal array. The resulting object had three inlets, then: two for the parameters used by Polansky and Tenney (number of elements and an exponent), and one for incoming random numbers from a [noise~] object.[5]

[4]In my version, I inverted Tenney's exponentiation: x^c. The user decides on a value of c, while x is the number of times the element has gone unchosen.

[5]The current code for this object can be found at www.github.com/joshuahudelson/Statfeed.git

FIGURE 1 — A patch using the statfeed~ object. The large, black, number boxes assign the number of elements and the exponent. The large canvas displays the signal.



Like any good user of Pd, my next thought was to wonder what sort of interesting output I might generate if I drove the object with something other than [noise~]. What about an [osc~] wave? Or a [sig~] with a constant value? As someone captivated by the numinous aura of cybernetics, I was particularly interested in driving the object with its own outputs—in other words, making a feedback loop of statistical feedback. Here, again, I had that familiar feeling of freedom about the open space of the patch bay, a sense that the analog had come to oxygenate the otherwise hermetic possibility space of the discrete and computational. Whereas Lisp, a much older language, required me to think in purely digital terms—concatenating the random number generation to the transfer function, making a decidedly unplayful digital tool—Pd seemed to opened up a virtual space of creativity rooted in the idea in the interaction between disparate entities. I would snip away the tendrils of digital determinism and unleash the becoming of the algorithm itself.

To make a long story short (circumventing months spent tinkering and numerous conversations with mathematicians and engineers), my fed-back version turned out to be, in

mathematical terms, something like a very bad pseudo-random number generator. In decoupling the [noise~] objects from the rest of the code, I'd drifted, I was told, into a corner of the vast field of Subshifts of Finite Type, which is itself a corner of the field of Symbolic Dynamics. More pointedly, the changes to my code that Pd had led me to with its propensity towards play and openness had, in fact, reproduced an even more primordially digital tool: pseudo-random number generation being the example *par excellence* of digital computation's dogmatism. This is not to say that the results weren't interesting. They are, and they led me to several new projects, including a numerical analysis of the fed-back statistical feedback algorithm and a project that uses virtual linear-feedback shift registers to perform sound synthesis. So it is not that Pd somehow inhibits creativity—far from it. But it does throw the openness of that creativity into question. If, as I claim, the digital is not about transistors but the much longer lineage of rationalization, then perhaps it makes sense that we find ourselves driven back to foundational concerns about our creative possibilities by a technology that constantly hints at that fact.

3. Conclusion

Much as Galloway claims that "language wants to be overlooked," (GALLOWAY, 2012, 62), we might say that one side of the nature of the the digital is to virtualize itself into invisibility. This is not in contrast to an "analog" that simply is what it presents itself as. Rather, as digital technology today does a better and better job of black-boxing itself, it makes the older, analog technology seem less digital by association. Pd is a wonderful tool because it gives composers power, but it does not rescue them from the primordial questions that are relevant for composers today: what sort of work is creativity in a world of widespread computing? Where do ideas of composer agency, choice, and creativity find themselves along a spectrum that runs between the randomness of c=1 and deterministic repetition? If Pd has something to teach students, it's that this question remains fundamental. In occupying the uncanny valley between digital and analog technology, it hardly makes learning easier, and it is not an idyllic open space of unbridled creative play. But it does, perhaps, bring to the fore the questions computer musicians need to ask themselves as we continue to think, create, and compute.

REFERENCES

AMES, Charles. Thresholds of Confidence: An Analysis of Statistical Methods for Composition, Part 1: Theory. *Leonardo Music Journal*, Vol. 5, 1995, pp. 33-38

PUCKETTE, Miller. Max at Seventeen. *Computer Music Journal*. Volume 26, Issue 4. Winter, 2002. pp. 31-43.

PUCKETTE, Miller. Interview on Future of Coding (podcast), episode #47, *Miller Puckette - Max/MSP and Pure Data*. May 12th, 2020. <https://futureofcoding.org/episodes/047.html> last accessed, November 26, 2021.

GALLOWAY, Alexander. *The Interface Effect*. Polity Press, 2012.

GALLOWAY, Alexander. *Laruelle: Against the Digital*. University of Minnesota Press, 2014.

DELEUZE, Gilles; GUATTARI, Felix. *A Thousand Plateaus: Capitalism and Schizophrenia.* Continuum Press, 2004 (original publication of translation, 1988).

MATHEWS, M. V. The Digital Computer as a Musical Instrument. *Science*, New Series, Vol. 142, No. 3592, 1963, pp. 553-557.

POLANSKY, Larry; BARNETT, Alex; WINTER, Mike. A Few More Words About James Tenney: Dissonant Counterpoint and Statistical Feedback. *Computer Music Journal*, Volume 5, Issue 2, 2011, pp. 63-82.

ABOUT THE AUTHOR

Joshua Hudelson is an independent scholar based in Beirut, Lebanon, where he studies the overlap between electrical infrastructure and electronic dance music culture. His book project, Spectral Sound: A Cultural History of the Frequency Domain, was supported by a Mellon/ACLS fellowship. Beyond his scholarly work, he develops interactive digital games for Braille literacy, and he was a co-recipient of the 2017 Louis Braille "Touch of Genius" Award from National Braille Press. He has taught at the American University of Beirut, New York University, The New School, the Migrant Community Center of Beirut, and the Sudanese Cultural Center of Beirut. ORCID: https://orcid.org/0000-0001-8168-435X. E-mail: joshua.hudelson@gmail.com