Od: Lorenz-system-based voltages for the Music Thing Modular Workshop System Computer [1]

[1] musicthing.co.uk/workshopsystem/

mcclintic.sphere.fx

December 18, 2024

Od is a program for the Music Thing Modular Workshop System Computer module that produces (loopable) modulation and pulse signals by simulating the Lorenz system.

The Lorenz System

The Lorenz system is a set of three coupled, first-order ordinary differential equations.

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = x(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$

Here, x, y and z are the "coordinates" of the system, their evolution as a function of time are the solutions to the set of differential equations, and will be referred to as "trajectories".

The constants ρ, β and σ determine the overall behavior of the system. For values of $\rho < 1$, all trajectories converge to the origin ($x, y, z = 0$). At $\rho = 1$, a bifurcation occurs such that for values of $\rho > 1$ there are a pair of fixed point attractors, and all trajectories will evolve towards one of the two.

$$x, y, z = (\sqrt{\beta(\rho - 1)}, \sqrt{\beta(\rho - 1)}, \rho - 1)$$

$$x, y, z = (-\sqrt{\beta(\rho - 1)}, -\sqrt{\beta(\rho - 1)}, \rho - 1)$$

For the values $\sigma = 10$, $\beta = 8/3$, $\rho = 28$ the system exhibits chaotic behaviour, and this corresponds to the "Lorenz Attractor". Trajectories of this system will propagate forever, never repeating or converging.

Solving the System

The solution of the system of ODEs is an "initial value problem", that is, the coordinates of an initial point are given, and the goal is (in general) to find the values x, y, z at some discrete list of time values. In this specific case, the goal is to be able to continually determine the next step in the trajectory so that the functions x, y and z can be exported as control voltage via the Computer's CV output sockets. The simplest approach to solution of an initial value problem is the "Euler method", which discretizes time into individual constant steps, Δt, allowing the next point on a trajectory to be obtained via a simple formula.

$$x_{n+1} = x_n + \Delta t f_x(t, x, y, z)$$

The error is on the order of Δt^2. For most applications, this approach is woefully poor. However, when generating solutions for musical purposes, trajectories that have the appropriate character are sufficient, and the simplest (and cheapest) approach can be relied upon.

The Module

The module itself utilizes a pair of Lorenz systems (A and B) which run concurrently at different rates. Inspired by the oCHD module, the relative rate is fixed at a musically engaging ratio. The overall rate is controlled by the main knob and one of the CV inputs. The trajectories of the system evolve according to the specified rate for A, and are made available as control voltage at the CV and CV/Audio output sockets of the Computer, using the full available range of approximately -6 to 6 V. Two Lorenz systems produce three trajectories each for a total of six, however only four can be sent out of Computer. Given the strong similarity between the trajectories of x and y, variables x and z are output for each system. This allows plotting of the classic Lorenz butterfly in two dimensions, and the creation of sound pictures of the same. The Y-knob is an overall attenuverter for the four CV outputs

The two pulse outputs of the computer fire whenever the x-coordinate crosses zero. Examining the plot of the trajectories shows that this will produce a rhythm that feels random, although it is completely deterministic. The combination of zero-crossing triggers from the two Lorenz systems will provide contrasting fast and slow rhythms, similar in some sense to the output of Rollz circuits. The top row of LEDs is used to show when the pulses are fired, left is the slower system and right the faster system.

Looping

It would seem ignorant to implement a module using the Lorenz system without making use of the fact that these dynamical systems are highly sensitive to initial conditions. This implies that two trajectories whose starting positions differ by a small amount will initially behave similarly, but over time will drift apart. This suggests the idea of returning to the start of a trajectory, but perturbing the initial position before computing the trajectory. The magnitude of this perturbation will determine for how long the first and second solutions of the system will remain similar before diverging.

This idea is also based on the concept that repeating randomness becomes not random at all. While the trajectories are not "random" in any sense - they are in fact completely deterministic - they are not repetitive in an obvious way and give a sense of back-and-forth evolution. It is not predictable to a listener when or how quickly the system will move from one attractor to another, or how long an orbit on an attractor will last. However, if a chunk of the trajectory is repeated over a short enough time range, the repetition will become obvious to the listener. Thus, we can take an unpredictable system and make it predictable by looping a section of a trajectory between an initial position x_0, y_0, z_0 and a final position x_N, y_N, z_N where N is the number of steps taken between the initial and final positions. Given this loop, the sensitivity to initial conditions can be made use of. By allowing the musician to control the magnitude of random noise added to the initial position, the loop can be tuned to be either the same every time ($\zeta = 0$), or to diverge on each repeat to a desired degree. This opens the door to a fuzzier definition of repetition where the loop is "similar enough" each time for the listener to detect repetition, but different enough to remain somewhat unpredictable, at the direction of the musician. This effect will also be felt on the pulse outputs. The X knob is used to set the magnitude of randomness.

The start point is set by sending a trigger into the leftmost CV input. This will store the position at the time the trigger was detected as t_0. Looping can be controlled either by a trigger at the rightmost CV input or by setting the Z-switch to the middle position (up corresponds to turning looping off). The goal of including the switch is playability of the loop, and the trigger enables some amount of repeatability. When looping is turned on, the position at the current time is used as t_N and the system will immediately return to t_0, evolving until the number of steps N has elapsed then returning to the position at t_0. Turning looping off un-loops the time circle back to a flat line and the system will continue to freely evolve without affecting t_0; t_N is forgotten.

The center row of LEDs shows the loop status. The leftmost LED is lit when looping is enabled (NB this is redundant with looking at the switch), and the rightmost flashes when the loop boundary is crossed (i.e. at the simultaneous start/end of the loop).

NB: maybe it would be nice if you could always return to your exact same loop after meandering, i.e after the first time you start a loop t_N would be retained unless explicitly altered, but would need to decide how (switch click?)

It would be maybe easier at first to prevent any control over system parameters that would lead to a stable state. Can the ranges be determined? Could you hold Z down to change the system params with the 3 knobs? Probably would want to switch looping to up so the clicker could be used for setting system params this way. Looping on Z switch may be pointless TBH, if not using it for that it could be used to change 2 modes? In system params mode the LED matrix could be used to indicate stability boundaries etc. if needed.

There are three system parameters, together they determine whether the system is stable, which is true iff the following expression is true.

$$\rho < \sigma \frac{\sigma + \beta + 3}{\sigma - \beta - 1}$$

The above holds only if $\sigma > \beta + 1$ which provides a hard limit on σ as a function of β. Changing sigma depends on where beta is - if the sigma knob tried to take the system to a stable state then sigma will stop changing, but a later change to beta may make the knob final position newly acceptable, in which case the system either has to move to that value of sigma or retain the stuck sigma value until the sigma knob gets moved again. Any time σ, ρ or β is changed, the stability must be evaluated. If sigma or beta is changed, the criterion making the stability expression valid has to be evaluated, before evaluating the stability.

The module wants to limit parameters such that this equation is always false.

Future Work

The implementation of Od takes care to avoid stable solutions of the Lorenz system (i.e. those with $\rho < 1$, or $\rho > 1$ where the stability test is able to confirm stability of the system. This in done to ensure the outputs of the module are chaotic. However, stable solutions could provide interesting, loopable decaying oscillatory envelopes. This could be incorporated in Od or act as the basis for a separate module or half-program of Computer.