Chaos on LUCIDAC

1 Introduction

With anabrid's[1] fully programmable LUCIDAC[2] analog computer it is now possible to explore dynamic systems without the need for manually patching cables or setting potentiometers. The system consists of eight integrators with two selectable time scaling factors $k_0 = 100$ and $k_0 = 10^4$, four multipliers, sources for constant values,[3] and 32 coefficients offering values in the interval $[-10, 10]$.[4] As in classic analog computers, integrators perform an implicit change of sign while multipliers do no sign inversion. The overall system is shown in figure 1.

The system uses a unique and novel interconnect structure shown in figure 2. On the upper left are the eight integrators and four multipliers with a total of 16 inputs. The twelve outputs of these computing elements as well as four additional constant values are fed into a voltage coupled U-*matrix* having 32 columns. Due to the voltage coupling, a single output value can be mapped onto any number of columns in parallel.

Each output column of this U-matrix is connected to a coefficient element based on multiplying *DAC*s (*Digital-Analog-Converter*s). These elements multiply an input voltage with a digitally controlled constant in the interval $[-10, 10]$ and yield an output current which is then fed into one of 32 columns of a current coupled I-*matrix* at the bottom of the figure. Using currents instead of voltages in this matrix has the benefit that addition can be done implicitly within the coupling matrix by connecting several columns to a common row line. Due to this, the LUCIDAC system does not need explicit summers as computing elements, greatly simplifying the hardware and reducing the demands on the interconnect structure. The current output rows of this matrix are connected to the individual inputs of the computing elements, closing the overall loop of the interconnect structure.

[1] `http://anabrid.com`, retrieved 20.11.2025.
[2] `https://anabrid.com/lucidac`, retrieved 20.11.2025.
[3] The system also contains eight analog input lines, eight analog output channels, and a builtin signal generator. However, these are outside the scope of this application note.
[4] Coefficients in classic analog computers only span the range $[0, 1]$.

Figure 1: LUCIDAC system

A simple problem such as the ordinary differential equation (*ODE*)

$$\ddot{y} = -\omega^2 y$$

can now be implemented as follows on a LUCIDAC system:[5]

1. The topmost integrator 0 will be fed with $\ddot{y}$ eventually and yields a voltage representing $-\dot{y}$ at its output.
2. Closing $\mathcal{U}_{00}$ connects this output voltage to output column 0 of the U-matrix which is connected to coefficient $\mathcal{C}_0$ which is set to ω.
3. Closing switch $\mathcal{I}_{10}$ then connects the current representing $-\omega\dot{y}$ to the input of integrator 1 which in turn yields an output voltage representing ωy.

[5]The individual switches of the U- and I-matrices will be denoted by $\mathcal{U}_{ij}$ and $\mathcal{I}_{ij}$ with $0 \leq i, j$ in the following.

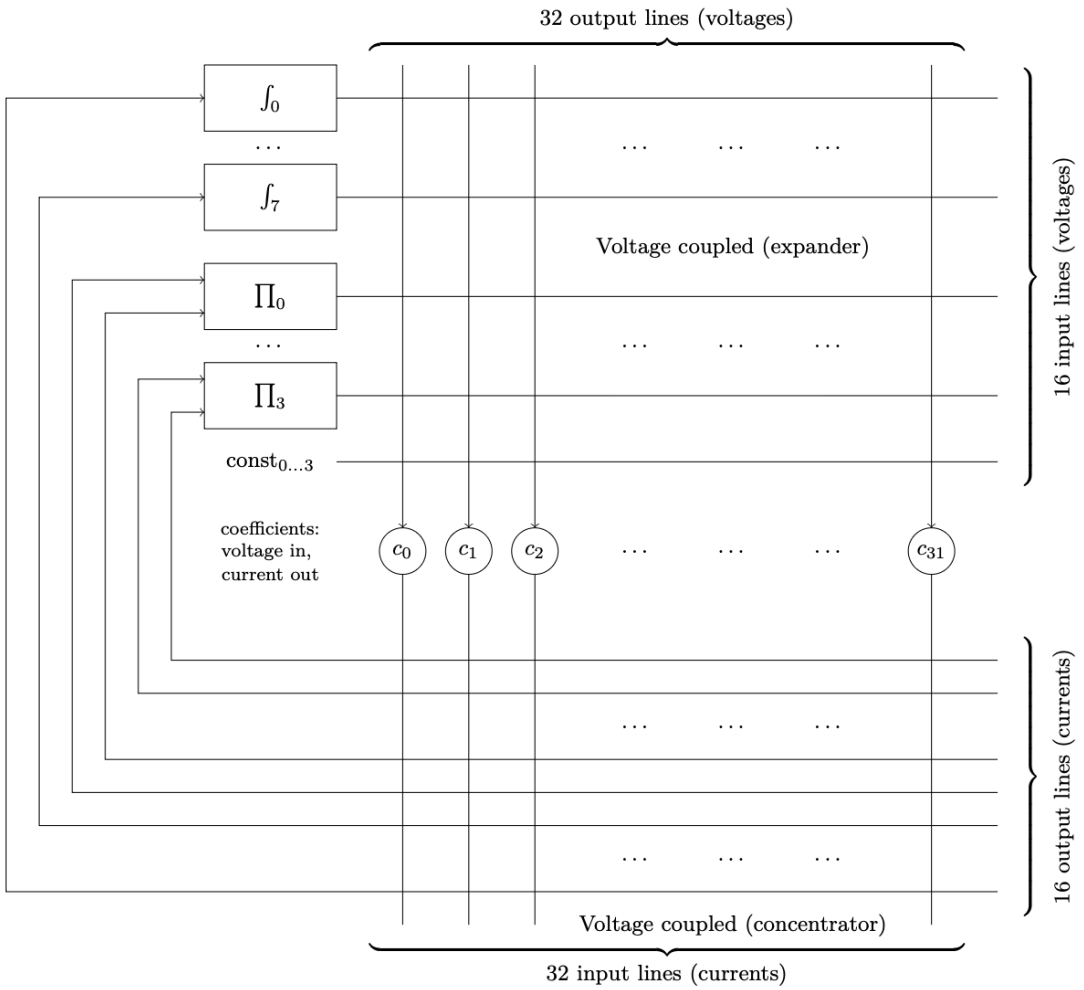

Figure 2: Interconnect structure of the LUCIDAC

4. The output of integrator 1 is then connected to $\mathcal{C}_1$ by closing switch $\mathcal{U}_{11}$. This coefficient is then set to $-\omega$ yielding a current proportional to $-\omega^2 y$ at the second column of the I-matrix.

5. Closing switch $\mathcal{I}_{01}$ now connects this value to the input of integrator 0 closing the overall loop.

6. Now, the desired k_0 and suitable initial conditions have to be set.

The actual programming of a LUCIDAC is done using a Python[6] package `pybrid-computing`.

2 Installation

In the following the `uv` package manager is used,[7] however, using Python venvs (virtual environments) or poetry will work, too. `uv` can be installed by typing[8]

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

in your shell and following the instructions on screen. In the next step a new environment is created and the `pybrid-computing` package will be installed:[9]

```
uv venv --python 3.13
uv pip install pybrid-computing
```

The environment is now ready to use. Assuming that the LUCIDAC is in the local network it can now be detected by executing

```
uv venv --python 3.13
```

which will generate an output like this:[10]

[6] `https://www.python.org`, version $3.11+$, retrieved 20.11.2025.
[7] `https://docs.astral.sh/uv/`, retrieved 20.11.2025.
[8] User input is shown in red in the following, output in blue, and source code in cyan.
[9] Currently only LINUX is supported, Mac OS X support will be available soon. Windows will take a tad longer...
[10] At the time of this writing there were three LUCIDACs in the local network.

```
Multiple LUCIDACs found, using the first one - use options -h and -p to
select a specific LUCIDAC.
Detecting network devices...
192.168.150.57 :5732 lucidac-17-E5-68._lucijsonl._tcp.local.
192.168.150.17 :5732 lucidac-15-87-A0._lucijsonl._tcp.local.
192.168.150.15 :5732 lucidac-17-E5-66._lucijsonl._tcp.local.
```

3 Systems

A number of examples for the LUCIDAC can be found at `https://github.com/anabrid/lucidac-resources/tree/main/examples`, two of which, namely the chaotic `Rössler` and LORENZ systems are shown below.

3.1 Rössler system

In 1976 `Otto Rössler` described a chaotic system[11] governed by the following three coupled differential equations:[12]

$$\begin{aligned}\dot{x} &= -(y+z)\\ \dot{y} &= x + ay\\ \dot{z} &= b + z(x-c),\end{aligned}$$

with parameters $a = 1/5, b = 1/5$, and $c = 5.7$. This system to be scaled accordingly so that no variable involved exceeds the value interval $[-1, 1]$ of the analog computer[13] yielding

[11] General information about the construction of chaotic systems can be found in [KUEHN, 2015][pp. 468].

[12] The remarkable thing about this system is that there is only one nonlinearity involved, see [RÖSSLER 1976] for details.

[13] Values are always restricted to $[-1, 1]$ while coefficients in a LUCIDAC system span the interval $[-10, 10]$.

the following system of scaled equations:[14]

$$\dot{x} = -0.8y - 2.3z$$
$$\dot{y} = 1.25x + 0.2y$$
$$\dot{z} = 0.005 + 15m$$
$$m = z(x - 0.3796)$$

This system of coupled differential equations can now be implemented by the following program:

```
from pybrid.lucidac.lucipy import Circuit, LUCIDAC
import matplotlib.pyplot as plt
import numpy as np

# Setup the circuit for the Roessler attractor:
r   = Circuit()                          # Create a circuit

x   = r.int(ic = .0066)                  # Integrator with IC
y   = r.int()                            # Integrator without IC
z   = r.int()
m   = r.mul()                            # Multiplier for nonlinearity
c   = r.const()                          # Constant source

r.connect(y, x, weight = -0.8)
r.connect(z, x, weight = -2.3)

r.connect(x, y, weight = 1.25)
r.connect(y, y, weight = -0.2)
```

[14] The code shown in the following can be found at https://github.com/anabrid/lucidac-resources/blob/main/examples/roessler.py, retrieved: 20.11.2025.

```
r.connect(c, z, weight = +0.005)
r.connect(m, z, weight = +5.)           # Multiple connections
r.connect(m, z, weight = +5.)           # get an overall
r.connect(m, z, weight = +5.)           # input weight of 15

r.connect(z, m.a, weight = -1)          # Compute nonlinear term
r.connect(x, m.b)
r.connect(c, m.b, weight = -0.3796)

r.measure(x, adc_channel=0)             # Connect ADCs to x, y, z
r.measure(y, adc_channel=1)
r.measure(z, adc_channel=2)

# Set environment variable LUCIDAC_ENDPOINT to a suitable
# connection string.
luci    = LUCIDAC()

luci.set_circuit(r)                     # Assign circuit

# Global control parameters
op_secs     = .1                        # OP time in seconds
sample_rate = 100_000                   # Samples per second for ADCs

luci.set_daq(num_channels=3, sample_rate=sample_rate)
luci.set_run(ic_time = 1_000, op_time=int(op_secs * 1_000_000_000))

run = luci.run()                        # Perform one OP cycle

samples = list(run.data.values())       # Read data
```

```
ax = plt.figure().add_subplot(projection='3d')
ax.plot(*np.array(samples), ls="-", marker="+", markersize=1.5)
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
plt.show()
```

To run this code a suitable environment variable pointing to the LUCIDAC to be used must be exported:

```
export LUCIDAC_ENDPOINT="tcp://192.168.150.57:5732"
```

The program can then be run by entering the following command

```
uv run python roessler.py
```

yielding an output like this:

```
2025-11-20 15:13:18 | INFO | computer | Start LUCIDAC at 192.168.150.57:5732.
2025-11-20 15:13:20 | INFO | computer | Executing run...
2025-11-20 15:13:20 | INFO | computer | Run done...
```

The results will be automatically displayed as a three-dimensional plot like the one shown in figure 3.

3.2 Lorenz '63 system

As simple as the Rössler system with its single yet vital nonlinearity is, it is not as well known as the chaotic system described by Edward Norton Lorenz[15] in 1963 (see [Lorenz, 1963]). Although Lorenz' seminal work was done on a Royal McBee LGP-30 digital computer, its defining differential equations

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = x(\rho - z) - y$$
$$\dot{z} = xy - \beta z,$$

[15] ⋆May 23, 1917, †April 16, 2008

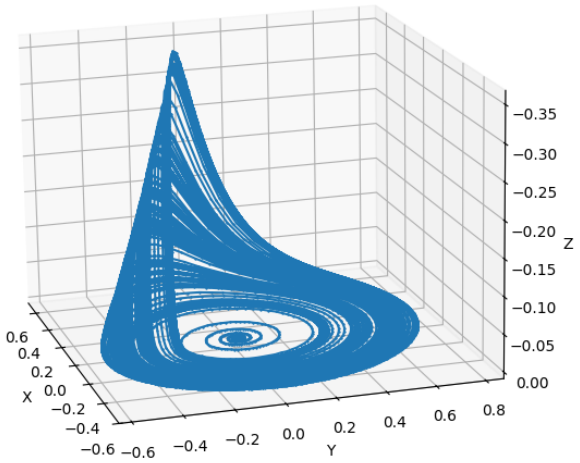

Figure 3: Typical three-dimensional plot of the RÖSSLER attractor

are ideally suited for implementation on an anlog computer. The parameters are $\sigma = 10, \beta = \frac{8}{3}$, and $\rho = 28$. After suitable scaling to ensure that no variable exceeds the value interval $[-1, 1]$, the system can be implemented on a LUCIDAC with the following program:[16]

```
from pybrid.lucidac.lucipy import Circuit, LUCIDAC
import matplotlib.pyplot as plt
import numpy as np

l   = Circuit()
```

[16] See https://github.com/anabrid/lucidac-resources/blob/main/examples/lorenz.py, retrieved 20.11.2025.

```
a   = 1.0
b   = 2.8
c   = 2.666 / 10

mx  = l.int()
my  = l.int()
mz  = l.int(ic = .3)
xz  = l.mul()
xy  = l.mul()

l.connect(mx, xz.a)                    # Product -x * -z = xz
l.connect(mz, xz.b, weight = 2)

l.connect(mx, xy.a)                    # Product -x * -y = xy
l.connect(my, xy.b)

l.connect(my, mx, weight = -a)
l.connect(mx, mx, weight = +a)

l.connect(mx, my, weight = -b)
l.connect(xz, my, weight = -5)
l.connect(my, my, weight = .1)

l.connect(xy, mz, weight = 2.5)
l.connect(mz, mz, weight = c)

l.measure(mx, adc_channel = 0)
l.measure(my, adc_channel = 1)
l.measure(mz, adc_channel = 2)
```

```
luci    = LUCIDAC()

luci.set_circuit(1)

op_secs     = .1
sample_rate = 100_000

luci.set_daq(num_channels=3, sample_rate=sample_rate)
luci.set_run(ic_time = 1_000, op_time=int(op_secs * 1_000_000_000))

run = luci.run()

samples = list(run.data.values())

ax = plt.figure().add_subplot(projection='3d')
ax.plot(*np.array(samples), ls="-", marker="+", markersize=1.5)
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")
plt.show()
```

With a suitably set environment variable `LUCIDAC_ENDPOINT` this program can now be run by the following command

```
uv run python lorenz.py
```

which generates a three-dimensional plot as shown in figure 4.

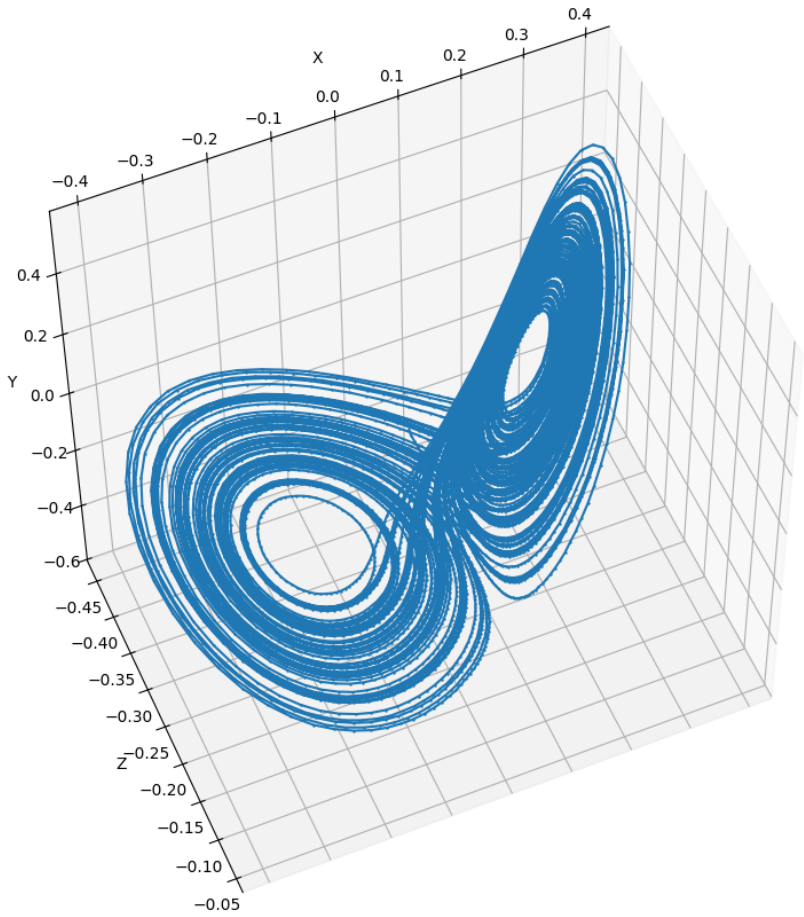

Figure 4: Typical three-dimensional plot of the Lorenz attractor

References

[Kuehn, 2015] Christian Kuehn, *Multiple Time Scale Dynamics*, Springer, 2015

[Lorenz, 1963] Edward Norton Lorenz, "Deterministic Nonperiodic Flow", in *Journal of the Atmospheric Sciences*, Volume 20, pp. 130–141, March 1963

[Rössler 1976] Otto E. Rössler, "An Equation for Continuous Chaos", in *Physics Letters A*, Volume 57, Issue 5, 12 July 1976, pp. 397–398