Solving PDEs on an analog computer

This application note shows a simple and straightforward way to solve partial differential equations (*PDE*s for short) on an analog/hybrid computer. It also shows how to use the Analog Paradigm hybrid controller to control the analog computer, gather data and generate a plot of the results.

The following is based on the one-dimensional wave equation as an example:

$$\frac{1}{c}\ddot{u} - \frac{\partial^2 u}{\partial x^2} = 0$$

Since an analog computer only has time as a free variable, a straightforward way to machinize a problem like this is to approximate the differential quotient $\frac{\partial^2 u}{\partial x^2}$ as a quotient of differences thus discretizing space yielding

$$\frac{1}{c}\ddot{u} - \frac{u_{i-1} - 2u_i + u_{i+1}}{(\Delta x)^2} = 0$$

with $\Delta x = \frac{x}{n}$ and $n \in \mathbb{N}$. Defining

$$\lambda := \frac{c}{(\Delta x)^2} = \frac{cn^2}{x^2}$$

results in a form suitable to derive an analog computer setup:

$$\ddot{u}_i = \lambda \frac{u_{i-1} - 2u_i + u_{i+1}}{(\Delta x)^2}.$$

Setting $n := 4$ and $\lambda := 1$ to simplify things further yields

$$\begin{aligned}
u_0 &= \delta(t)\\
\ddot{u}_1 &= u_0 - 2u_1 + u_2\\
\ddot{u}_2 &= u_1 - 2u_2 + u_3\\
\ddot{u}_3 &= u_2 - 2u_3 + u_4\\
\ddot{u}_4 &= u_3 - 2u_4 + u_5\\
u_5 &= 0
\end{aligned}$$

with $\delta(t)$ being an impulse occurring at $t = 0$.

These four coupled ordinary differential equations (*ODE*s for short) can now be transformed into an analog computer program by integrating twice over each second derivative $\ddot{u}_i$, yielding the corresponding u_i term. The overall setup is shown in figure 1. The input pulse is applied as an initial condition to the first integrator.

Using a four-channel oscilloscope the problem variables $u_1, \dots, u_4$ could be displayed easily, running the analog computer in repetitive mode with a high time scale factor set on the integrators. In the following, the Analog Paradigm hybrid controller (HC) was used to control the analog computer from an attached digital computer. The HC module can also act as a data logger, a feature used to generate the graph of the resulting functions.

The hybrid controller is programmed using a Perl-module `IO::HyCon`[1]. This expects the analog computer setup to be described in a configuration YAML-file as shown below:

[1]See `https://metacpan.org/pod/IO::HyCon`.

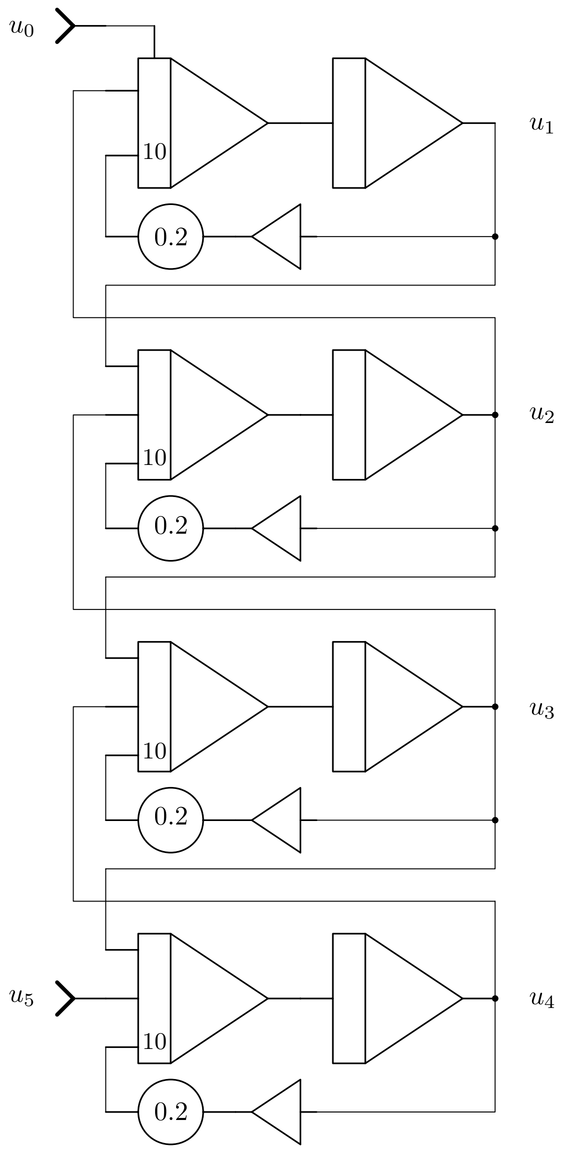

Figure 1: Computer program for solving the one-dimensional wave equation

wave_equation.yml

```
serial:
    port: /dev/cu.usbserial-DN050L21
    bits: 8
    baud: 115200
    parity: none
    stopbits: 1
    poll_interval: 10
    poll_attempts: 20000
types:
    0: PS
    1: SUM8
    2: INT4
    3: PT8
    4: CU
    5: MLT8
    6: MDS2
    7: CMP4
    8: HC
elements:
    U1: 0260
    U2: 0261
    U3: 0262
    U4: 0263
problem:
    times:
        ic: 10
        op: 200
    ro-group:
        - U1
        - U2
        - U3
        - U4
```

wave_equation.yml

This configuration file consists of the following sections:

`serial`: This section contains the settings required for the USB-based serial communication interface. Typically, only the `port` has to be changed when using a different computer setup. All other parameters are at their default values.

`types`: Here all known computing elements are defined with their respective ids.

`elements`: This section contains a list of all computing elements with their respective hexadecimal addresses. These elements can be used as parts of a read-out group (see below).

`problem`: This section contains several subsections of which only two are used in this example: The `times`-subsection contains the settings for the duration of the IC- and OP-phase of a simulation run. The subsection labelled `ro-group` consists of a list of all computing elements which should be read out continuously during a run. In this case these are the four integrators labelled `U1`,...,`U4` from the `elements` section above.

The corresponding Perl program using this configuration file[2] is shown in the following listing:

wave_equation.pl
```
use strict;
use strict;
use warnings;
use IO::HyCon;              # Load the hybrid controller Perl module.

```

[2]It should be noted that the name of the Perl program and the configuration file have to be identical, the only difference being the file extensions which are `.pl` and `.yml` respectively.

```
my $ac = IO::HyCon->new();  # Create a new hybrid controller object.
$ac->setup();               # Setup the analog computer as described in the
                            # configuration file.
$ac->single_run_sync();     # Start a single run.
$ac->get_data();            # Get the data gathered by the hybrid controller
$ac->plot();                # and generate a plot.
```
wave_equation.pl

First, the Perl module `IO::HyCon` is loaded, and an analog computer control object `$ac` is instantiated (as a singleton). Calling the method `$ac->setup()` automatically configures the hybrid controller and the analog computer. In this case only the IC- and OP-times as well as the readout group consisting of the four integrators `U1`,. . . ,`U4` are set.

Calling `$ac->single_run_sync()` initiates a single run of the analog computer in synchronous mode, i. e. the program waits until the IC- and OP-phases have been completed. The method call `$ac->get_data()` reads the data gathered by the hybrid computer during the OP-phase and stores it in an internal data structure of the `$ac`-object.

A plot of the four functions $u_1, \dots, u_4$ can then be generated by calling `$ac->plot()`. The result of a typical computer run with $u_0 = +1$ is shown in figure 2.

Calling `$ac->plot(type => '3d')` generates a 3d(-ish) plot as shown in figure 3.

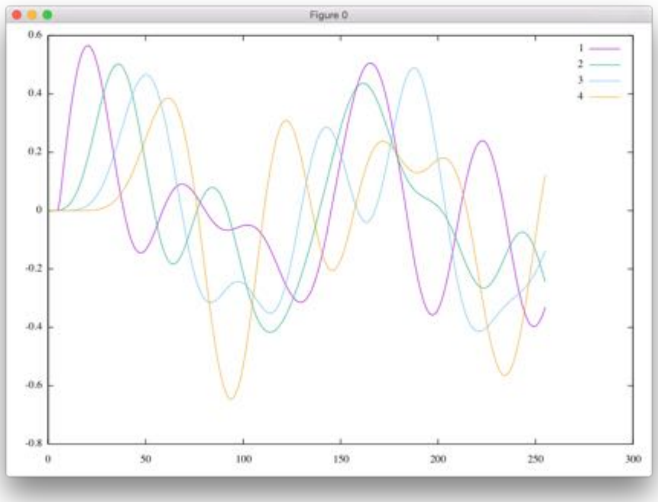

Figure 2: Solution of the one-dimensional wave equation with the range x divided into four sections of equal width

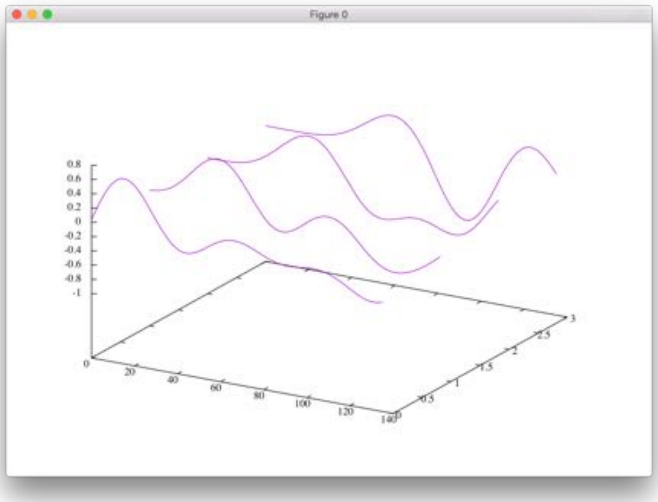

Figure 3: Solution of the one-dimensional wave equation with the range x divided into four sections of equal width plotted with `type => '3d'`