CS522A (Fall 2003) Assignment 1 – SOLUTION

Thomas

October 6, 2003

Download: [PDF] [PS]

Contents

1 Code

The following files are modified. Replace the old ones in the hand-out package gives you the final solution to Assignment 1.

- `SIM_startResume.py`. Put it in `CausalBlockDiagram/`. This file implements the causal block solver.
- `Graph.py`. Put it in `CausalBlockDiagram/`. This file fixes the fatal bugs in the old Topological Sort algorithm.
- `circle_CausalBlockDiagram_delay_mdl.py` Put it in `CausalBlockDiagram/models`. This is the circle model with delay blocks. (To answer a question from some students, this file is *automatically generated* and bugs are not found until this moment. Please don't modify it manually or even try to understand it.)

2 Purely Algebraic Models

The simulation result of `algebraTest_CausalBlockDiagram_mdl.py` (with `time_max`=0) is shown in Figure 1.

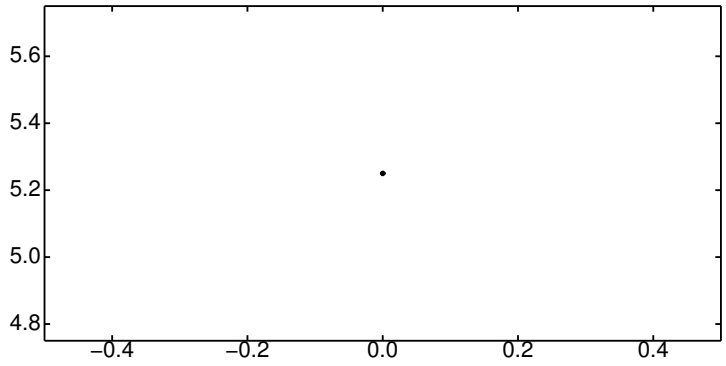

Figure 1: Simulation result of algebraic model (`time_max`=0), *time vs y*

Another test is run, with `time_max`=50, as shown in Figure 2.

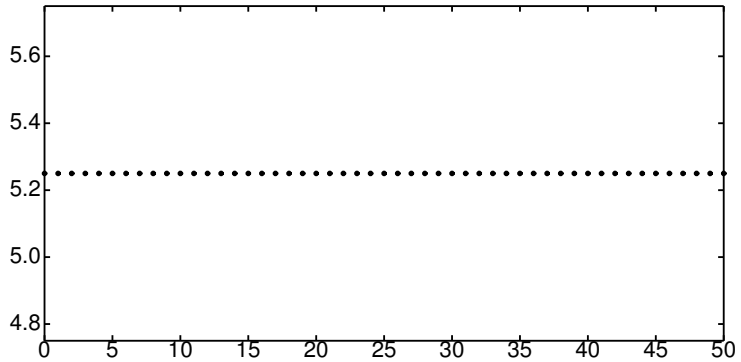

Figure 2: Simulation result of algebraic model (`time_max`=50), *time vs y*

3 Dependency Loops

The solver detects dependency loops and highlights them (by calling the HighLight function of the graph objects). It runs `loopTest_CausalBlockDiagram_mdl.py` and gives the output shown in Figure 3.

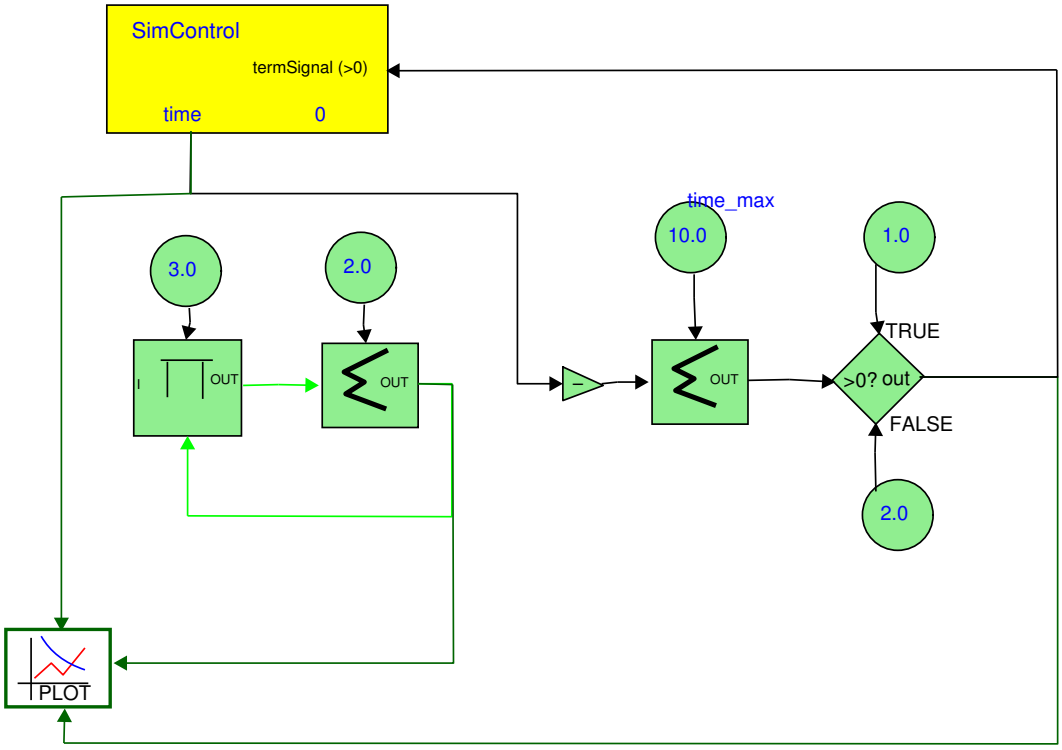

Figure 3: A loop detected in the loop model

In addition, such an exception is thrown and dumped to the console:

```
Exception in thread Thread-1:
Traceback (most recent call last):
  File "/home/thomas/tools/lib/python2.3/threading.py", line 436, in __bootstrap
    self.run()
  File "/home/thomas/tools/lib/python2.3/threading.py", line 416, in run
    self.__target(*self.__args, **self.__kwargs)
  File "CausalBlockDiagram/SIM_startResume.py", line 105, in initSimul
    raise NotImplementedError, "algebraic loop detected"
NotImplementedError: algebraic loop detected
```

4 Delay Blocks

The solver recognizes delay blocks. Model `delayTest_CausalBlockDiagram_mdl.py` runs correctly (Figure 4).

5 Signals

All the (previous and current) values of the connections are stored in their `signal` lists. Please read the commented SIM_startResume.py file to get more insight. `block_tmp_value` is completely removed. `block_out_value` is

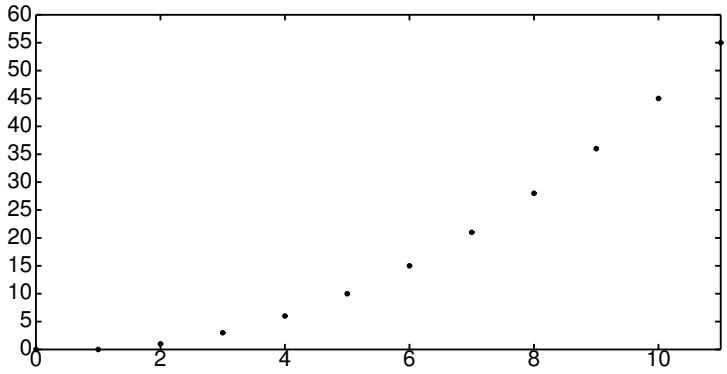

Figure 4: Simulation result of the delay model, *time vs x*

used only to retrieve the values of constant blocks (whose pre-defined `block_out_value` attribute is given by the designer).

6 Completeness of the Solver

All the blocks are considered, except `Integrator`, `Derivative`, `Time` and `FileIO`. If any one of them is encountered in a model, an exception is raised and the simulation stops.

7 Discretized Circle Test Model

The circle is defined by equations:

$$\begin{cases} x'' = -x \\ x_0 = 0 \\ x'_0 = 1 \end{cases}$$

This can be rewritten as:

$$\begin{cases} \dfrac{dx}{dt} = y \\ \dfrac{dy}{dt} = -x \\ x_0 = 0 \\ y_0 = 1 \end{cases}$$

And since $\dfrac{dx}{dt} = \dfrac{x_i - x_{i-1}}{\Delta t}$, we can substitute the above equations as:

$$\begin{cases} \dfrac{x_i - x_{i-1}}{\Delta t} = y_{i-1} \\ \dfrac{y_i - y_{i-1}}{\Delta t} = -x_i \\ x_0 = 0 \\ y_0 = 1 \end{cases}$$

Note that in the first equation ($\dfrac{dx}{dt}$), backward approximation (y_{i-1}) is used, while in the second one, we use forward approximation (x_i). This is because:

- If we use forward approximation in both of them, an algebraic loop is created (x_i depends on y_i and y_i also depends on x_i), which is hard to solve.
- If we use backward approximation in both of them, the convergence speed is much slower.

The graphical model is depicted in Figure 5.

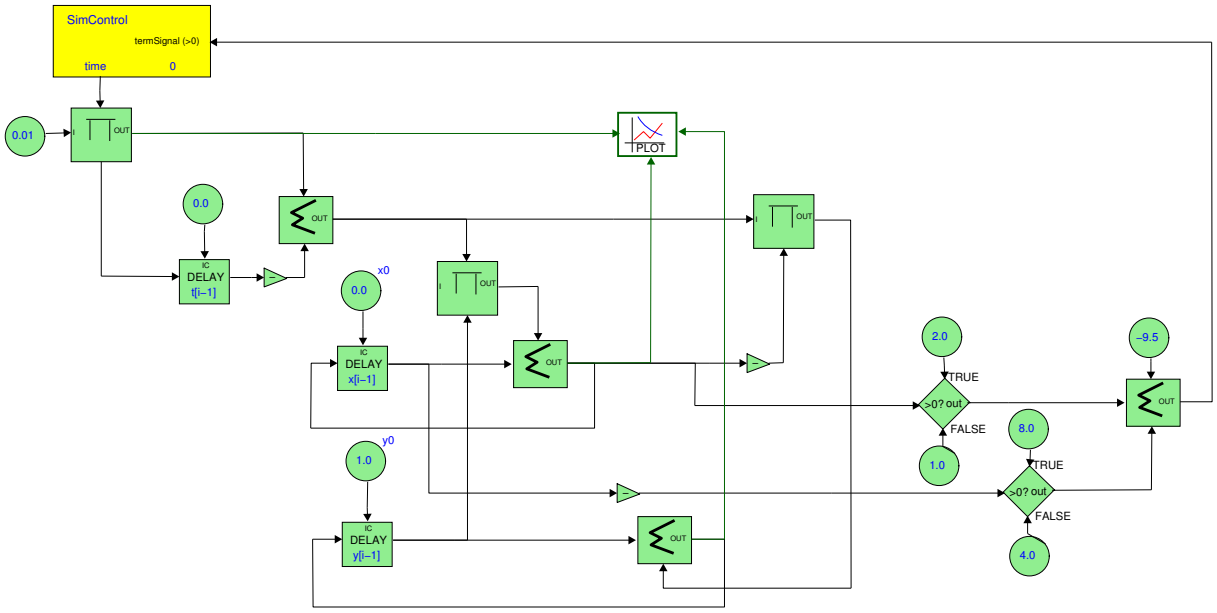

Figure 5: Discretized Circle Model

The termination condition is the completion of a full circle. This is tested with 2 test blocks: the upper one returns 2 if x_i is larger than 0; the lower one returns 8 if $-x_{i-1}$ is larger than 0 (that is, $x_{i-1} \leq 0$). This two test results are added. Only when both conditions are satisfied (the plot crosses the last point) will the final result be (subtracted by 9.5) larger than 0. Simulation stops at that time.

Figures 6, 7 and 8 shows the simulation result.

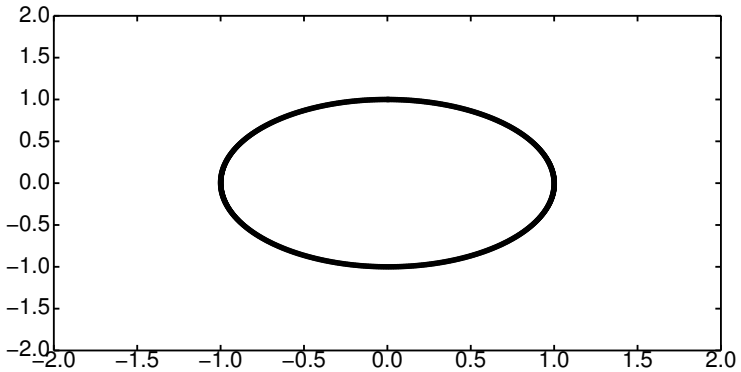

Figure 6: Simulation result of the discretized circle model, *x vs y*

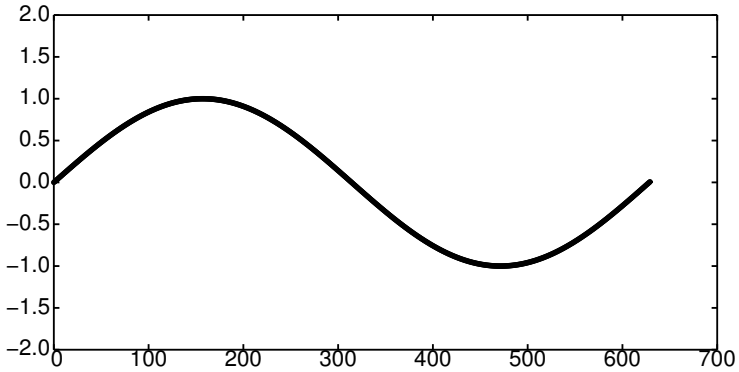

Figure 7: Simulation result of the discretized circle model, *t vs x*

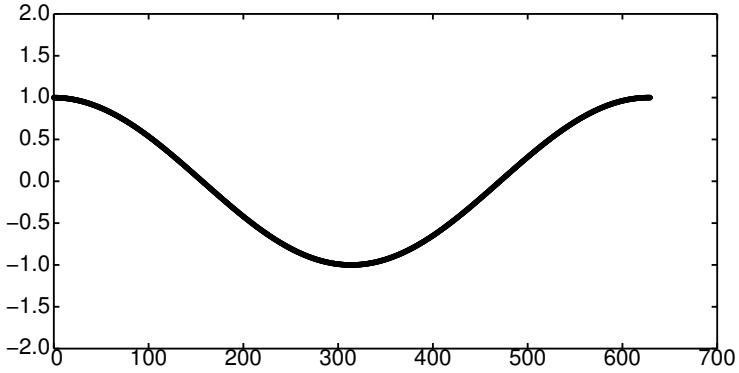

Figure 8: Simulation result of the discretized circle model, *t vs y*