Causal Block Diagram assignment

Fall Term 2003

General Information

- The due date is **Sunday 5 October 2002**, before 23:55.
- Submissions must be done via WebCT. Beware that WebCT's clock may differ slightly from yours. As described on the Assignments page, *all* results must be uploaded to WebCT and accessible from links in the index.html file. There is no need to upload AToM^3.
- The assignment must be made in teams of two people. It is understood that all partners will understand the complete assignment (and will be able to answer questions about it).
- Grading will be done based on correctness and completeness of the solution. Do not forget to document your requirements, assumptions, design, implementation *and* modelling and simulation results in detail !
- Extensions, if given, will involve extending not only the alotted time, but also the assignment.

The assignment

1. You will use the meta-modelling environment AToM^3 AToM3-2.2.tgz to implement a Discrete-Time Causal Block Diagram simulator.

   All relevant files are found in the directory `CausalBlockDiagram/`. Example models are found in the directory `CausalBlockDiagram/models/`.

   Implementing the simulator means you will modify the file `SIM_startResume.py`.

   You may wish to have a look at `SIM_pause.py` and `SIM_reset.py`. Actually, if you wish to understand how the plotting works, have a look in the `Plotting/` directory. The `PlotTest.py` is a stand-alone script demonstrating the working of the plotter.

   As a starting point, you *MUST* read and understand `DUMP_model.py`. It contains examples of *ALL* parts of the AToM^3 API you'll need.

   The `SIM_startResume.py` given is the solution to last year's assignment. It will work on models `circle_CausalBlockDiagram_mdl.py`, `ballistic_CausalBlockDiagram_mdl.py`, and `lorenz_CausalBlockDiagram_mdl.py`. That solution provides a Continuous-Time Causal Block Diagram simulator. Also, there is no `SimControl` block. Rather, `simul_t_init`, `simul_t_final`, `simul_delta_t`, and so on are given as explicitly as global model attributes.

2. The simulator must be able to handle Discrete-Time models as well as purely algebraic models as shown in model `algebraTest_CausalBlockDiagram_mdl.py`. The model is shown in Figure 1 Note how the `time_max` is `0.0`. Your simulator should calculate a solution at that time only.

3. The simulator must be able to detect dependency loops. The model `loopTest_CausalBlockDiagram_mdl.py` depicted in Figure 2 for example contains an algebraic loop. You must isolate the algebraic loop and `HighLight` the links involved. An example of highlighting is given in `DUMP_model.py`.

4. Above all, the simulator must be able to handle delay blocks. The `delayTest_CausalBlockDiagram_mdl.py` model depicted in Figure 3 is an example. Note how in this example, the simulation termination condition uses the

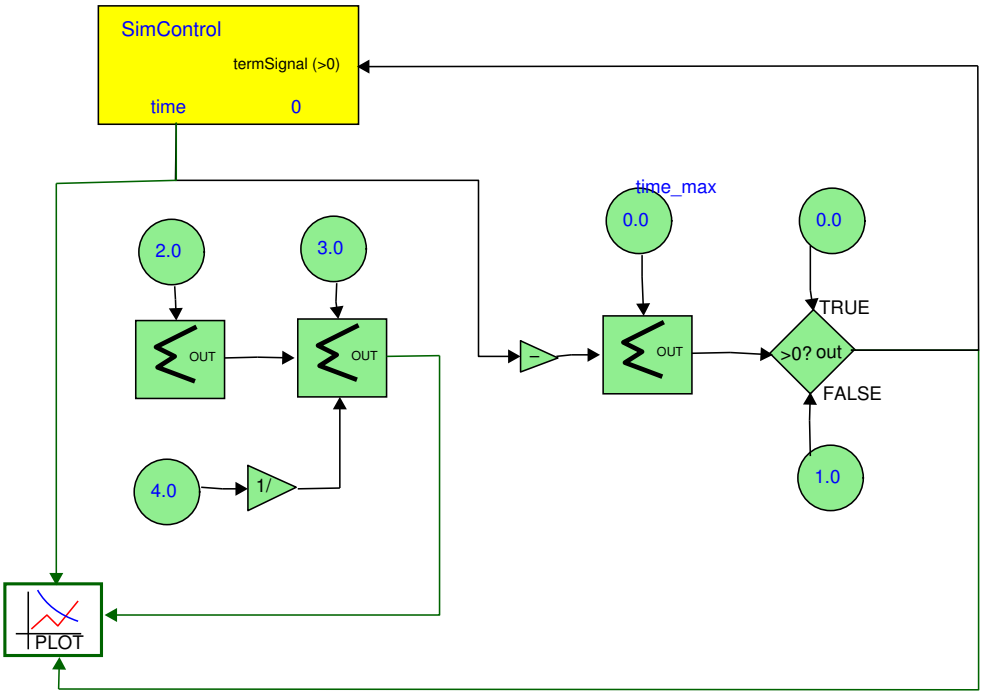

Figure 1: Algebraic Model CBD

state variable `x` rather than time. When `x` reaches the value `50.0`, the simulation should stop.

5. Unlike last year, you must now keep signal values in the `signal` attribute (an AToM^3 list) of each link node. You can now ignore block attributes such as `block_out_value` and `block_tmp_value`.

6. Note how you must handle all block types except `Integrator, Derivative, Time, FileIO`.

7. You must build a model for the "circle test". This is the Ordinary Differential Equation (ODE)

$$x'' = -x, x(0) = 0, x'(0) = 1.$$

   Using integrator blocks, this is shown in Figure 4. You must us a discretized version (using only Delay blocks) of the ODE. You must however use a discretization of Derivatives rather than of Integrators.

8. You must include plots of relevant simulation results in your solution report. The best way to do this is to `File/Generate Pos` from the AToM3 menu bar. You can then convert this `.eps` file to a bitmap using for example the `display` tool on Linux machines.

Using the AToM^3 environment

Download AToM3-2.2.tgz archive.

Expand it locally (for example with the command `tar --ungzip -xvof AToM3-2.2.tgz` if you're working on a UNIX machine. This will create a directory `AToM3-2.2`.

To start the AToM^3 environment, `python ATOM3.py`. `python` should be at least version 2.2 of Python (this is installed in the SOCS labs). On UNIX, the script `atom3` is a shortcut for the above command. On windows, you can just click on the ATOM3 icon to launch it.

AToM^3 will be started with the CausalBlockDiagram formalism loaded. You can either build your own model (and `File/Save` it) or `File/open` an existing one. In the directory `CausalBlockDiagram/models/` you will find the example models.

When AToM^3 starts with the CausalBlockDiagram formalism, it will use a number of files in the `CausalBlockDiagram` directory:

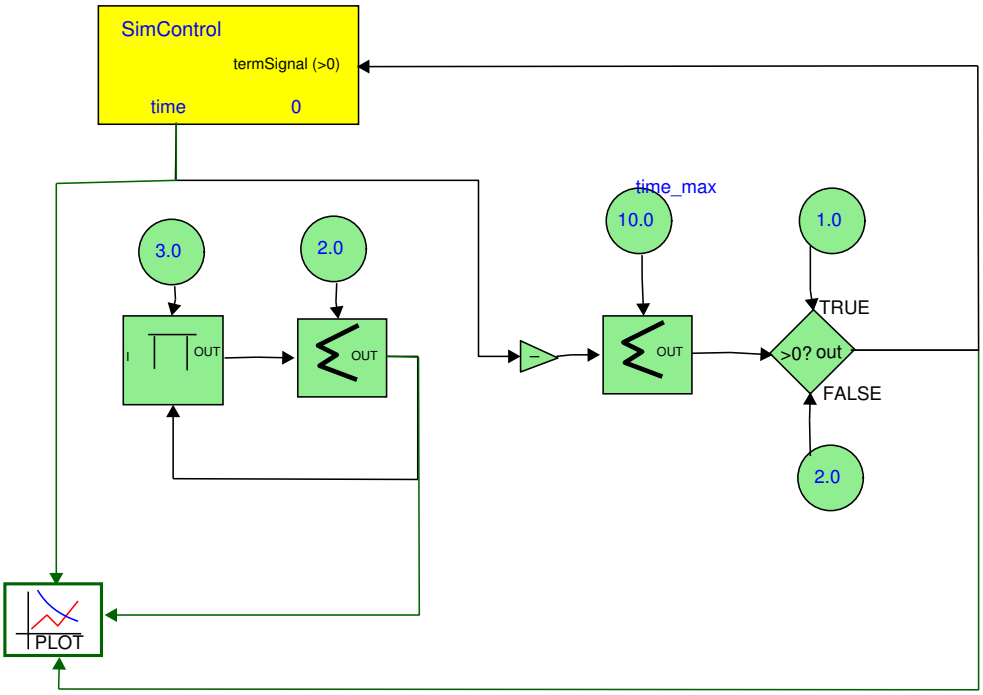

Figure 2: Model containing Algebraic Loop

- `MODEL_plotWindow.py` contains code which gets called when you click on a Plot icon. It opens a plot window. When pressing "new" in that window, it is possible to add plot items (one variable as a function of another).
- `EXPORT_LaTeX.py` constains code which gets called when you press the "to LaTeX" button.
- `EXPORT_Mfile.py` contains code which gets called when you press the "M-File" button.
- `SIM_startResume.py` contains the startResume(parentApp, model) method which gets called when you press the "Exp Start/Resume" button. Currently it starts a thread which contains a data generator. You have to replace this generator by a Time Slicing simulator which produces data using the model structure and node attributes.
- `SIM_pause.py` and contains code which pauses the simulation. It gets called when the "Exp Pause" button is pressed.
- `SIM_reset.py` contains code which gets executed when the "Exp Reset" button is pressed.

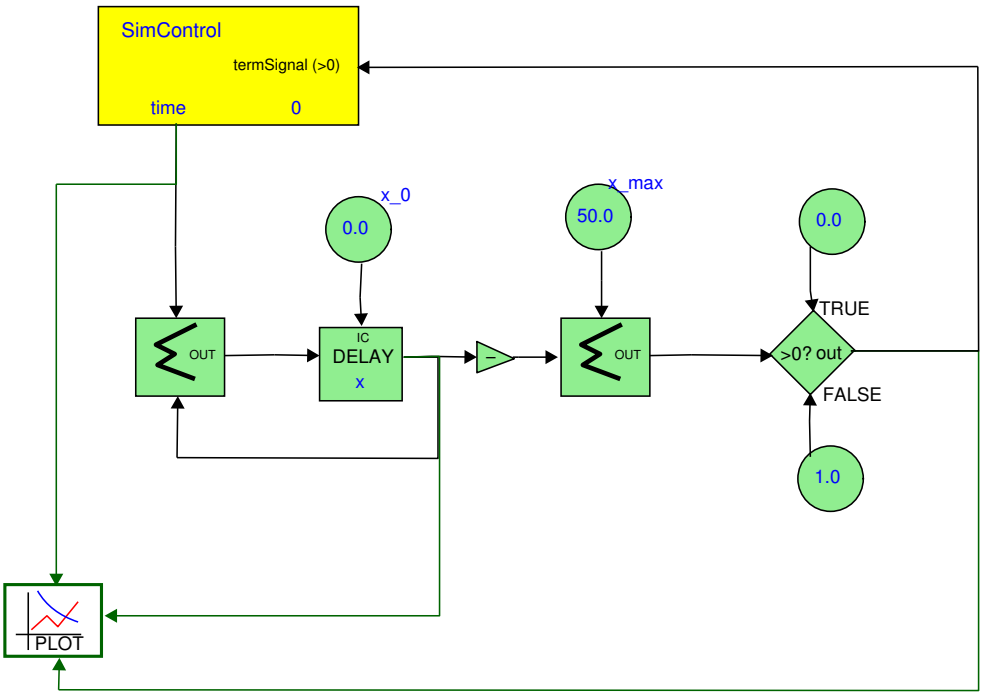

Figure 3: Model with Delay Block

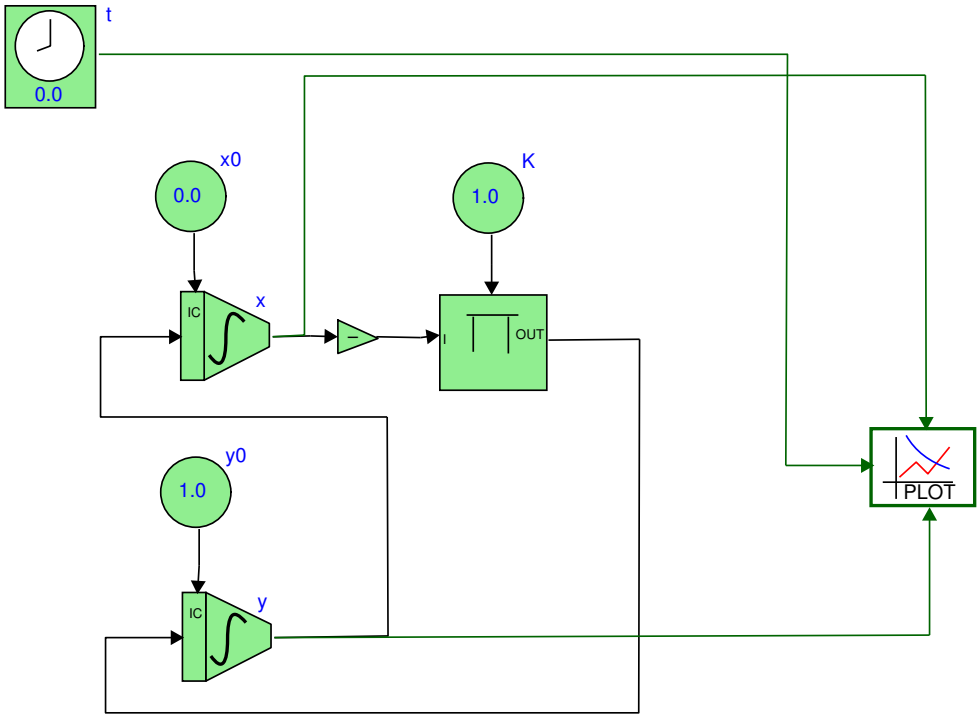

Figure 4: Circle Test Model