UNIVERSITÄT SALZBURG

An Introduction to Simulink – How Does It Work?

Gerald Stieglbauer

CS, University of Salzburg, Austria

SOFTWARE RESEARCH LAB

Resources

- Actor-oriented design

  http://ptolemy.eecs.berkeley.edu/presentations/main.htm

- Using Simulink

  http://www.mathworks.de/access/helpdesk/help/toolbox/simulink/

  (also available as PDF)

- Matlab/Simulink R13 installed at

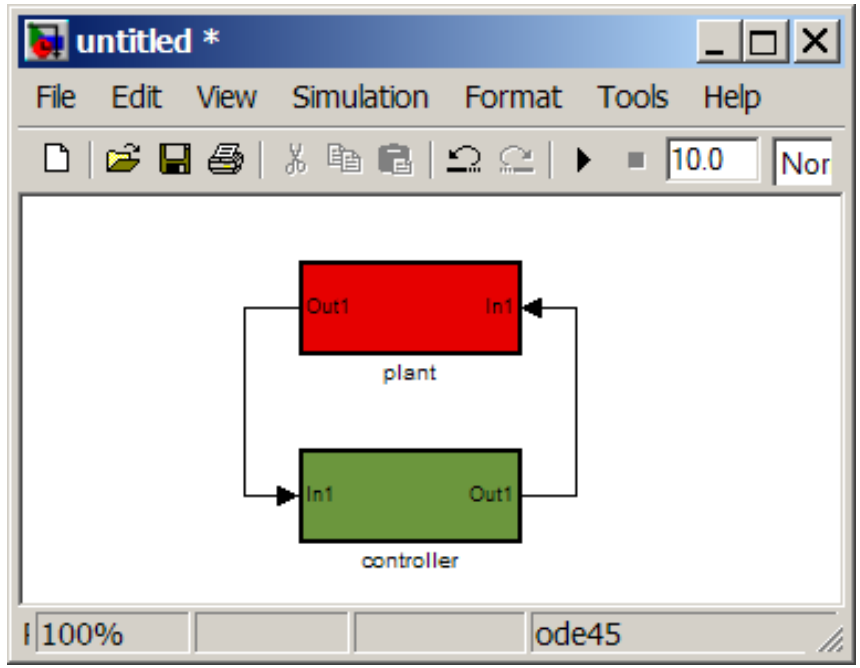
  /usr/common/matlab-13

What is Simulink?

- An actor oriented, model-based, graphical modeling tool
- A software package for modeling, simulating and analyzing dynamic systems
- It is one of the most used modeling tool in the industry
- It is an add-on of Matlab, a matrix computation program with its own language
- Supports code generation by add-ons like the Real-Time Workshop Embedded Coder

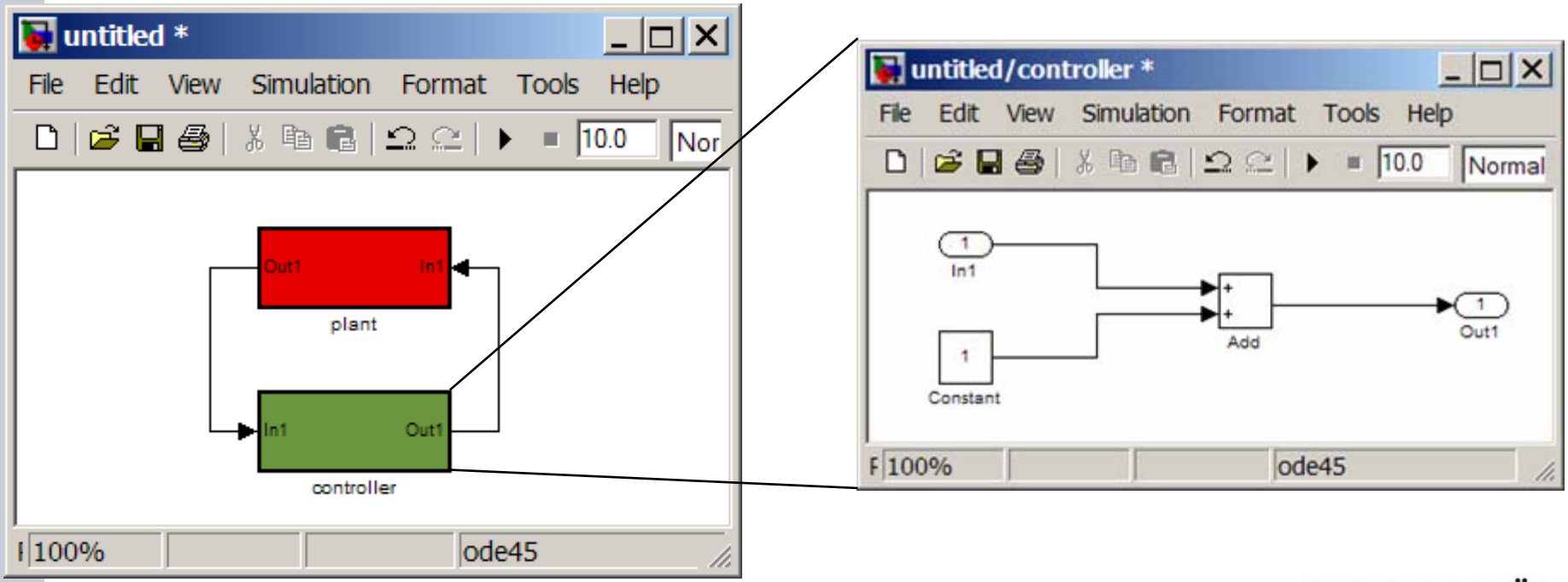
Blocks and Signals

- In Simulink an *actor* is called *block*
- A block consists of some *functionality* and an arbitrary number of *ports*
- Lines called *Signals* connected to the block‘s ports passes data between blocks



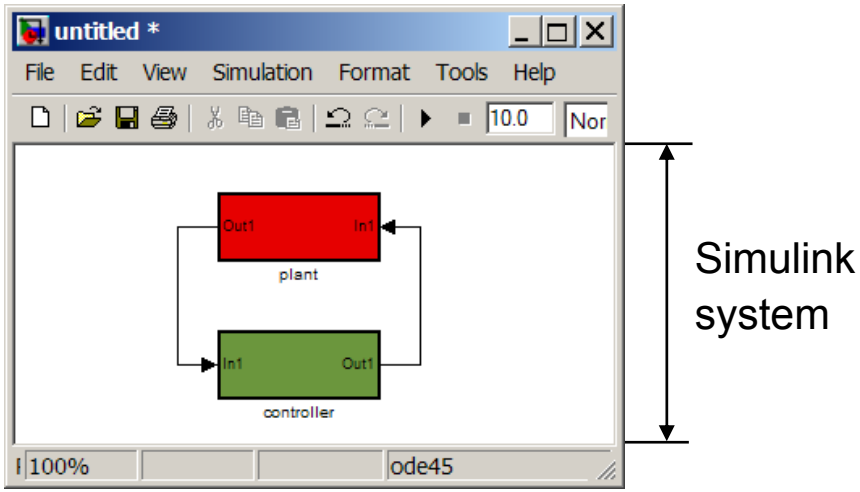
How to choose the needed functionality

- Using pre-defined blocks of the Simulink library
- Create your own functionality by using
  - S-Function blocks (writing your own function in C, Fortran, etc.)
  - Subsystem blocks



Subsystem block

- Every ‘container’ that holds blocks and signals is called a *system*



- Subsystem blocks structure Simulink models
- Communication to parent system by using Port blocks

Sample time

- *The functionality of a block is used to calculate the values of the output ports of a block based on the values of the input ports and the internal states of a block.*
- Every block has a *sample time.*
- *The sample time determines how often and when the functionality of a block is evaluated.*

Continuous vs. Discrete blocks

- Continuous blocks have an infinitesimal sample time
  - e.g. Integrator block, Derivative block

- Discrete block must have a sample time greater zero
  - Blocks can be configured by a sample time parameter
  - If sample time is -1, it is inherited either from the block connected to its input (inheritance) of its output (back inheritance).

Block updates

- A block update includes:
  1. Compute the block‘s outputs (for all blocks within a system)
  2. Compute the block‘s states (for all blocks within a system)
  3. Computes the time for the next time step (for all blocks within a system)

- If the functionality of an S-Function block has to be defined, all these steps have to be implemented

Atomic vs. Virtual Subsystem block

- The content of a Atomic Subsystem block is computed at once

- The content of a Virtual Subsystem block is flattened to to level of the parent system.

  →*Virtual Subsystems have no sample time*

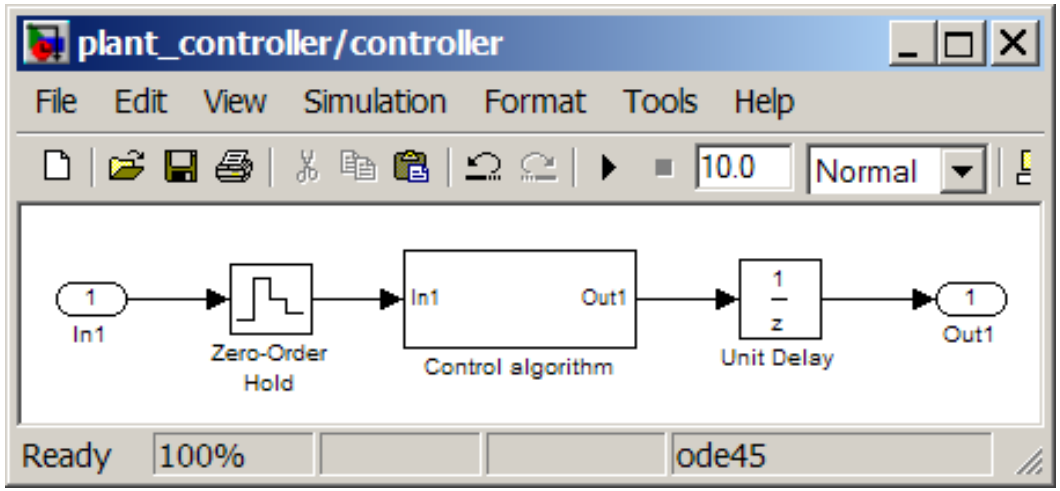
Direct Feedthrough vs. Non-Direct-Feedthrough

- Direct Feedthrough
  - The calculation of the output values of a block depends on the input values of the current sample time
  - e.g. Sum block, ZOH block

- Non-Direct-Feedthrough
  - The calculation of the output values of a block depends on the input values of a previous sample time and/or on the block‘s states.
  - e.g. Unit-Delay block, Constant block

Implementing the LET in Simulink

- One possible implementation of the LET:



- Zero-Order Hold block: Samples and hold the input value for the given sample time.
- Unit Delay block: Delays its input by the specified sample period.

Determing Block Update Order

- *Each block must be updated before any of the blocks whose direct-feed-through ports it drives.*
- *Blocks that do not have direct-feed-through inputs can be updated in any order as long as they are updated before any blocks whose direct-feed-through inputs they drive.*
- Attention: Block Update Order can be changed for some blocks if switching from Atomic to Virtual Subsystem blocks and vice versa!

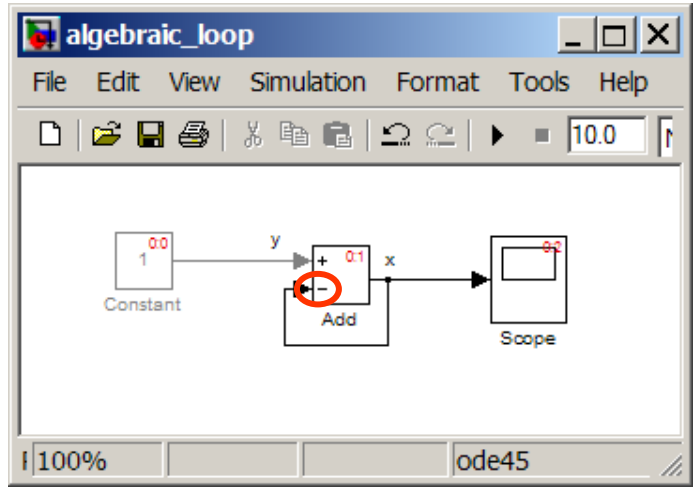
Algebraic loops

- The output of direct feedthrough blocks cannot be computed without knowing the input values.

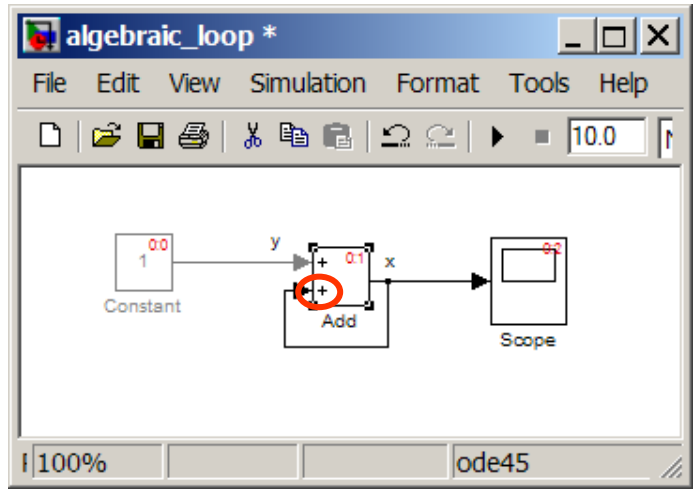
- An algebraic loop generally occurs when an input port with direct-feed-through is driven by the output of the same block, either directly, or by a feedback path through other blocks with direct-feed-through.

Example of an algebraic loop

- x = y – x
- → 2x = y
- → x = y/2



- BUT:
- x = y + x
- → y = 0
- → x = ???
- → **algebraic loop not solvable!**

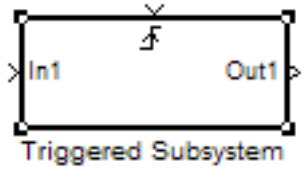


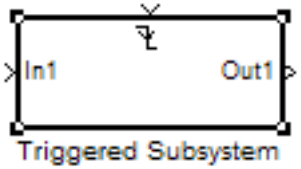
Conditionally Executed Subsystems

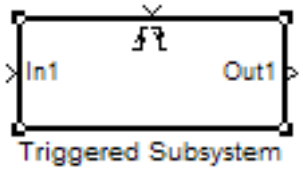
- A conditionally executed subsystem is a subsystem whose execution depends on the value of an input signal.
- Different types:
  - Enabled Subsystem
    - Executed if the control signal has a positive value
  - Triggered Subsystem
    - see next slide
  - Control Flow Subsystem
    - Executed if the control flow condition (if, while, for conditon) evaluated to true

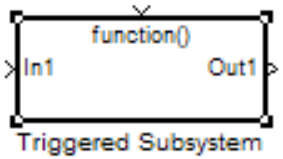
Triggered Subsystems

- Different types of trigger events:
  - The control signal has an rising edge
  - The control signal has an falling edge
  - The control signal has an rising or an falling edge
  - The control signal is a *function-call signal*









Function-call signal

- A function-call signal can either stem from a
  - Function-call generator block
  - S-Function block
    - →can be used to override Simulink‘s block update order!
    - →Using a S-Function to implement a E machine
  - Stateflow block
    - →Stateflow blocks are used to define Statecharts. E.g. Switching from one state to another can cause a function call.

Implementing a E machine for Simulink

- Using function call mechanism