Integrated Modeling, Control Synthesis and Code Generation with Modelica and Dymola

Johan Åkesson
Department of Automatic Control, Lund University

Hilding Elmqvist

Ulf Nordström
Dynasim, Lund
Project in Automatic Control
FRT090

Johan Åkesson
Department of Automatic Control
Faculty of Engineering
Lund University
Project in Automatic Control

- Advanced level course 7.5 ECTS units
- Course runs for seven weeks
- Team effort
  - Collaborative problem solving
- Get practical experience
  - Work in the lab
- Apply course knowledge
  - Modeling and identification
  - Control design and implementation
Project in Automatic Control
2009

• 25 students
  • Mostly from Lund but some exchange students
  • Several disciplines
    • Engineering physics
    • Applied mathematics
    • Computer sciences
    • Chemical engineering

• Two groups working with Lego/Dymola
  • Five students in each group
Lego with Modelica/Dymola

- Build the NXTway two-wheel robot
- Physical modeling with Modelica
  - Multi-body dynamics
- Model calibration
  - Experiments
  - Dymola Calibration module
- Control design
  - Derive simple model
  - Develop control scheme
Lego with Modelica/Dymola

- Automatic generation of fixed point controller code
  - No C programming
  - Software in the loop evaluation
- User interaction
- Deployment on NXTway
- Animation in Dymola
  - Real-time animation
- Get in touch with industry
  - Tutorials held by Dynasim personnel
Lego Dymola Groups
Course plan

• w1. Form groups and planning
  • Tuesday March 17th – group announcement
  • Friday March 20th – project plan dead line

• w2.-w3.
  • Tutorial
  • Weekly meetings with project supervisors

• w4.-w7
  • Weekly meetings with project supervisors
  • Presentation and demo in w7.
Lego Dymola Tutorials

1. Introduction to Modelica (AC)
2. Multi-body modeling (Dynasim)
   • Wheel models (by Martin Otter)
3. Code generation with Dymola (Dynasim)
   • Modelica_Embedded
Modelica/Dymola Modeling

Hilding Elmqvist
MultiBody modeling

- Bodies and Joints
- Automatic 3D animation
Example – Double pendulum

- World system (= Inertial system)
- Rotational joint
- 1D rotational flange for drive
- 1D rotational damper
- Bodies
Wheel Model

- The MultiBody library does not contain any wheel models.
- A separate library IdealWheels contains a wheel set where each wheel can be driven separately.

This is thus suitable for modeling of LEGO Mindstorms robots.
Body with two wheels

- The separate model WheelSetAndBody contains a body with two wheels.
- The wheels are driven by two sinusoidal torques with different frequencies.
Resulting animations
Linearization

\[ x \] \[ z \] \[ \text{world} \]

\[ x \] \[ z \] \[ \text{world} \]

\[ \text{tau}_1 \] \[ \text{tau}_2 \]

\[ \text{torque}_1 \] \[ \text{torque}_2 \]

\[ n=(0,1,0) \] \[ r=(0,0,0.2) \]

\[ \text{revolute} \] \[ \text{bodyBox} \]

\[ \text{wheelSet} \] \[ \text{tau} \]

\[ \text{tau}_1 \] \[ \text{tau}_2 \]

\[ \text{a} \] \[ \text{b} \] \[ \text{r} \]

\[ \text{tau}_1 \] \[ \text{tau}_2 \]

\[ \text{a} \] \[ \text{b} \]

\[ \text{revolute} \] \[ \text{bodyBox} \]

\[ \text{wheelSet} \] \[ \text{tau} \]

\[ \text{tau}_1 \] \[ \text{tau}_2 \]

\[ \text{a} \] \[ \text{b} \]

\[ \text{revolute} \] \[ \text{bodyBox} \]

\[ \text{wheelSet} \] \[ \text{tau} \]

\[ \text{tau}_1 \] \[ \text{tau}_2 \]

\[ \text{a} \] \[ \text{b} \]
Linear model

Inputs
- fileName
- showEigenValues

Outputs
- Inputs
- States

States
- "tau1", "tau2"
- "wheelSet.x", "wheelSet.y", "wheelSet.phi",
- "wheelSet.thetal", "wheelSet.theta2", "wheelSet.der_thetal", "wheelSet.der_theta2",
- "revolute.phi", "revolute.w"

Commands

```plaintext
= Modelica_LinearSystems.StateSpace(
  A =
  [0, 0, 0, 0, 0, 0.05, 0.05, 0, 0;  
   0, 0, 0, 0, 0, 0, 0, 0, 0;  
   0, 0, 0, 0, 0, (-0.2), 0.2, 0, 0;  
   0, 0, 0, 0, 0, 1, 0, 0, 0;  
   0, 0, 0, 0, 0, 0, 0, 0, 0;  
   0, 0, 0, 0, 0, 0, 0, 0, 0;  
   0, 0, 0, 0, 0, 0, 0, 0, 0;  
   0, 0, 0, 0, 0, 0, 0, 0, 0;  
   0, 0, 0, 0, 0, 0, 0, 0, 0;  
   0, 0, 0, 0, 0, 0, 0, 0, 0;  
   0.945581386124056, 10.4163334752606, (-0.945581386124056), 10.4163334752606,

  B =
  [0, 0;  
   0, 0;  
   0, 0;  
   0, 0;  
   0, 0;  
   0.4163334752606, (-0.945581386124056), 10.4163334752606;  
   0, 0;  
   3.34261838440113, 3.34261838440117],

  C =
  [1,  1;  
   1,  1;  
   1,  1]

  D =
  ["tau1", "tau2"]
)
```
Animation with LEGO shapes

- Dymola supports visualization of shapes described by DXF, STL and VRML (subset) files.
- LEGO building blocks are described in .DAT format: [http://www.ldraw.org](http://www.ldraw.org)
- LEGO parts are available at: [http://www.ldraw.org/Downloads-req-viewdownload-cid-1.html](http://www.ldraw.org/Downloads-req-viewdownload-cid-1.html)
- Electric Mindstorms NXT shape: [http://www.ldraw.org/cgi-bin/ptdetail.cgi?f=parts/53788.dat](http://www.ldraw.org/cgi-bin/ptdetail.cgi?f=parts/53788.dat)
Ldraw to DXF Converter

- Download parts directories from:
- Mindstorms parts are among the unofficial parts
- See dat2dxf\readme.txt for instructions on how to organize files
LEGO blocks resource

Unofficial File parts/53788.dat

File Header:
- Electric Minstorms NXT (Complete Shortcut)
- Name: 53788.dat
- Author: Kevin Clague [kclague]
- !LDRAW_ORG Unofficial_Part
- !LICENSE Redistribution under CCAL version 2.0: see CReadme.txt
- BFC CERTIFY CCW
- !HISTORY 2006-03-30 [kclague] Initial design
- !HISTORY 2007-12-30 [Phil] Complete redesign, use stone colors
- // Battery Lid - Dark Stone

Status:
- 7 subfiles aren't certified. (C:SSSSSSSSX)
- Size: 1153 bytes
Dymola animation of NXT (shape 53788)
Tutorial
Modelica and Dymola for System Design
Model Building and Calibration

H. Elmqvist
Dynasim AB, Lund
Calibrate engine map parameter and friction
Modelica_EmbeddedSystems and Code generation for Lego Mindstorms NXT

Ulf Nordström
Contents

- Overview
- Modelica_EMBEDDEDSystems
- Fixed point Code Generation
- Lego Mindstorms
Modelica_EMBEDDEDSYSTEMS

Key Components

- Communication
- Configuration
  - Target
  - Task
  - Subtask
Modelica_EmbeddedSystems

Key Components

Communication

Input and output reside in the same base-rate task
decoupleReal specifies boundary between subtasks

communicateReal in Unnamed

Communication using ports
Inter-task communication or using external I/O
Modelica_EmbeddedSystems

Key Components

Configuration

Target

Task

Subtask

![Configuration Screen Shot](image_url)
Modelica_EmbeddedSystems

Key Components

Configuration

Target

Task

Subtask
Modelica_EmbeddedSystems

Key Components

Configuration

Target

Task

Subtask

- Task in which subtask is running (synchronization between subtasks is automatic due to equation sorting)
- Identifier of subtask (unique within the task)
- Type of subtask (Continuous, Periodic, Triggered, Disabled)
- Sample period
- Sample offset factor
- Integration method
- Step size for fixed step integration method
Use pull-down menu to select target/task/subtask"
Modelica_EmbeddedSystems

Scenarios

- **Model In the Loop**
  - Quantization
  - Communication delay

- **Software In the Loop**
  - Task decomposition
  - Effects of fixed point arithmetics

- **Embedded**
Modelica_EmbeddedSystems

Scenarios

- Model In the Loop
  - Quantization
  - Communication delay

![Modelica_EmbeddedSystems Diagram]
Modelica_EmbeddedSystems

Scenarios

- Software In the Loop
  - Effects of fixed point arithmetics

![Graph showing real and fixed point signals with reference](image)
Modelica_EmbeddedSystems

Scenarios

- Embedded
  - Change target identifier to "lego"
  - Will be discussed in the Lego Mindstorms section
Use "extend from" to create new configurations/scenarios from the same base model

- Use modifiers to change attributes
Modelica_EMBEDDEDSYSTEMS

Example – Software In the Loop

Set Hidden.DecomposeInTasks = true

Task decomposition
- Reference subtask
- Feedback subtask
- Plant subtask
Modelica_EmbeddedSystems

Example – Software In the Loop

Task decomposition
- Reference subtask
- Feedback subtask
- Plant subtask
Modelica_EmbeddedSystems

Example – Software In the Loop
FixedPoint Code Generation

- Setup/Configuration
  - Mapping annotation
- Restrictions
FixedPoint Code Generation

Setup/Configuration – mapping annotation

FixedPoint variables must be annotated

- min
- max
- resolution

```modelica
Modelica.Blocks.Sources.Ramp ramp(
  height(
    min=0,
    max=100) = 100 annotation (mapping(resolution=0.001)),
  duration(
    min=0,
    max=50) = 10 annotation (mapping(resolution=0.001)),
  y(min=0, max=100) annotation (mapping(resolution=0.01)),
```
FixedPoint Code Generation

Setup/Configuration – mapping annotation

Declaration

```plaintext
/* output Modelica.Blocks.Interfaces.RealOutput ramp.y(min = 0.0, max = 100.0) 
annotation(mapping(resolution = 0.01)); */
int ramp_yFF;  /* Q[7, 0] */

/* parameter Modelica.SIunits.Time ramp.duration(min = 0.0, max = 50.0) = 10 
annotation(mapping(resolution = 0.001)); */
int ramp_durationFF = 320;  /* Q[6, 5] */

/* parameter Real ramp.height(min = 0.0, max = 100.0) = 100 
annotation(mapping(resolution = 0.001)); */
int ramp_heightFF = 1600;  /* Q[7, 4] */

Q[nQ, nQ] = [integer bits, fractional bits]
```

Generated code

```plaintext
/* ramp.y = ramp.offset+(if time < ramp.startTime then 0 else (if time < 
   ramp.startTime+ramp.duration then (time-ramp.startTime)*ramp.height/
   ramp.duration else ramp.height)); */
ramp_yFF = (((ramp_offsetFF << 9) + (((timeFP0_0 < (ramp_startTimeFP << 5)) ? (0 
  << 9) : (((timeFP0_0 < (ramp_startTimeFP + ramp_durationFP) << 5)) ? (((
  timeFP0_0 - (ramp_startTimeFP << 5)) * ramp_heightFP) / ramp_durationFP) : (ramp_heightFP << 5)))))) >> 9;
```
FixedPoint Code Generation

Setup/Configuration – mapping annotation

Use parameter dialog to input annotations as modifiers

![Parameter Dialog Example](image-url)
FixedPoint Code Generation

Setup/Configuration – enable FixedPoint

FixedPoint will be enabled when the subtask belongs to a target with "floatingpointUnit = false"
Lego Mindstorms

- **Framework**
  - dsmodel and dymosim
  - dymola_wrapper in nxtOSEK
- **Setup/Configuration**
- **Limitations**
- **Example**
Lego Mindstorms

Framework

declarations.c
equations.c
dymola_wrapper.c

Code generation

Code generation

dsmodel.c

Send internal variables over bluetooth

Build

Store and plot

dymosim
Lego Mindstorms

Framework – dymola_wrapper.c

```c
#include "kernel.h"
#include "kernel_id.h"
#include "ecrobot_interface.h"
#include "target_port.h"

/* OSEK declarations */
DeclareCounter(SysTimerCnt);
DeclareTask(Task1);

/* Include fixedpoint variable declarations */
#include "declarations.c"

/* Task1 executed every 50msec */
TASK(Task1)
{
    /* map system time to fixedpoint time */
    timeFP0_0 = (int)1024*systick_get_ms()/1000;

    /* Include fixedpoint equations */
    #include "equations.c"

    TerminateTask();
}
```

Generated by Dymola
Lego Mindstorms

Setup/Configuration

Configuration for Lego target and bluetooth communication
Translate and compile Dymola model
- CD to "../nxtOSEK/samples/dymola"
- Press translate button in Dymola

Compile dymola_wrapper using Cygwin
- CD to "../nxtOSEK/samples/dymola"
- Compile with "make all"
- Start Lego and download executable with "/ramboot.sh"
Lego Mindstorms

Setup/Configuration

Starting Lego and dymosim for Bluetooth communication and program execution
Lego Mindstorms

Setup/Configuration

- Results are automatically stored in Dymola
Thank you for your attention and Good Luck with your projects