Model based design (MBD) – a free tool-chain

Simon Mayr, (Gernot Grabmair)

Simon Mayr, simon.mayr@fh-wels.at, University of Applied Sciences Upper Austria , Austria

Projects
Projects dealing with Scilab/XCos

- **PROTOFRAME – Framework und frontend for semi-automated matching of real and virtual prototypes**
  - Work in progress

- MOdoPS – MOdel based Design by OPen Source
  - Project finished
  - Project result: Scilab/XCos example library

Overview

- Model based design (MBD)
- Code generation from XCos
- Example (cart and pendulum)
- Conclusion

Model based design (MBD)
Definition

- Mathematical and visual method applied in designing embedded software to address problems associated with
  - Complex control
  - Signal processing
  - Communication systems

Model based design (MBD)
Applications

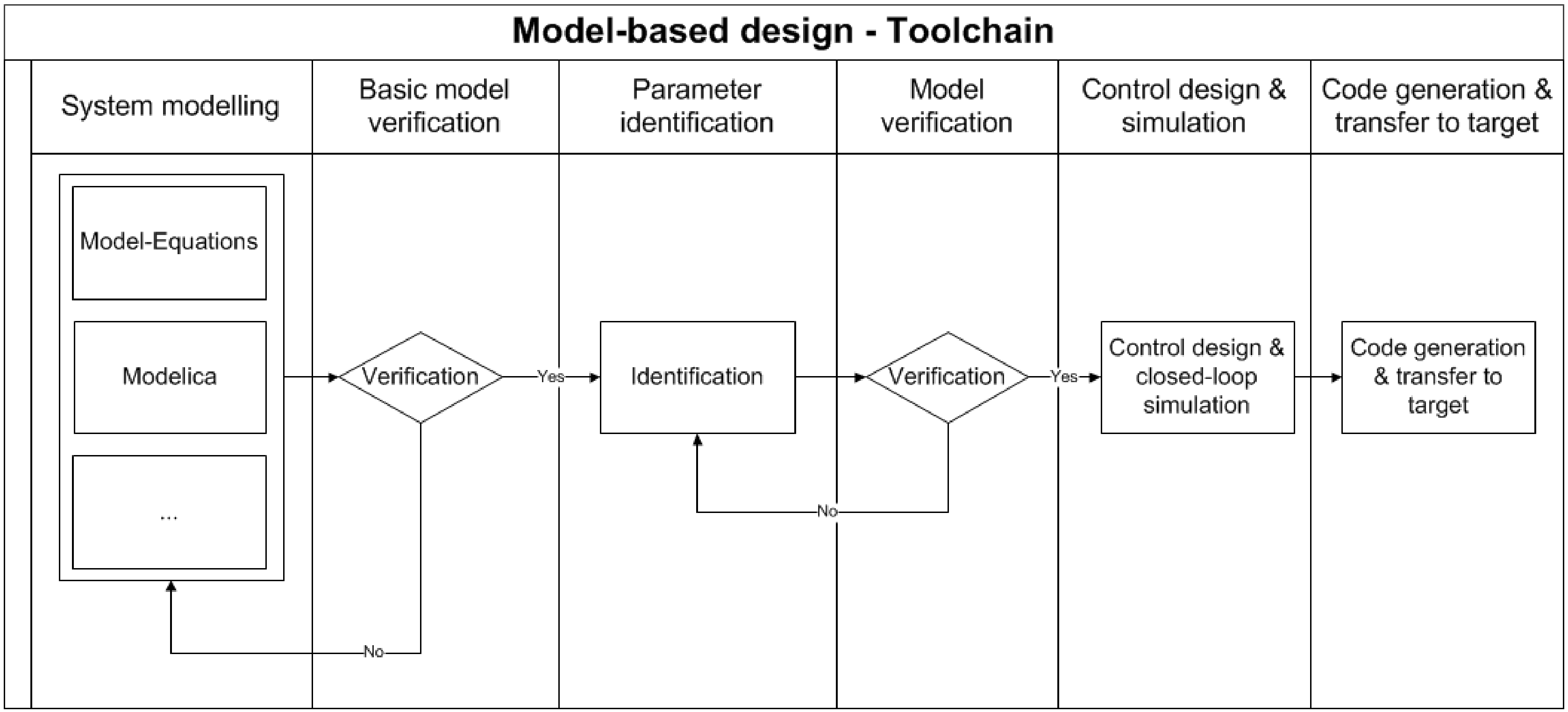
- Common fields of application are:
  - Motion control applications
  - Industrial equipment
  - Aerospace applications
  - Automotive applications
  - …

Model based design (MBD)

Main steps of model based (controller) design

- System modelling and basic model verification
- Parameter identification and model verification
- Control design and closed-loop simulation
- Code generation and transfer to target

Model based design (MBD)
Main steps of model based (controller) design



Model based design (MBD)
Advantages

- Faster and more cost-efficient development
- Errors in system design can be located and corrected in early stage of the project, when financial and time impacts of the system redesign are relatively small
- Extension and/or modification of an existing system is relatively easy

Model based design (MBD)
Common commercial tool-chains

- Typical examples of commercial tools are:
  - Matlab/Simulink
  - Dymola
  - …
- Advantages:
  - Advanced and well-proven software
  - Complete tool-chains
- Disadvantages:
  - Quite expensive
  - Unsuitable for small and medium-sized companies

Model based design (MBD)
Free tool-chain

- Scilab/XCos

- Advantages:
  - Plant modeling
  - Control design & simulation

- Disadvantages:
  - Code generation is not implemented

- Solution:
  - Use an external application to generate code from XCos diagram

Code generator for Scilab/XCos

State of the art

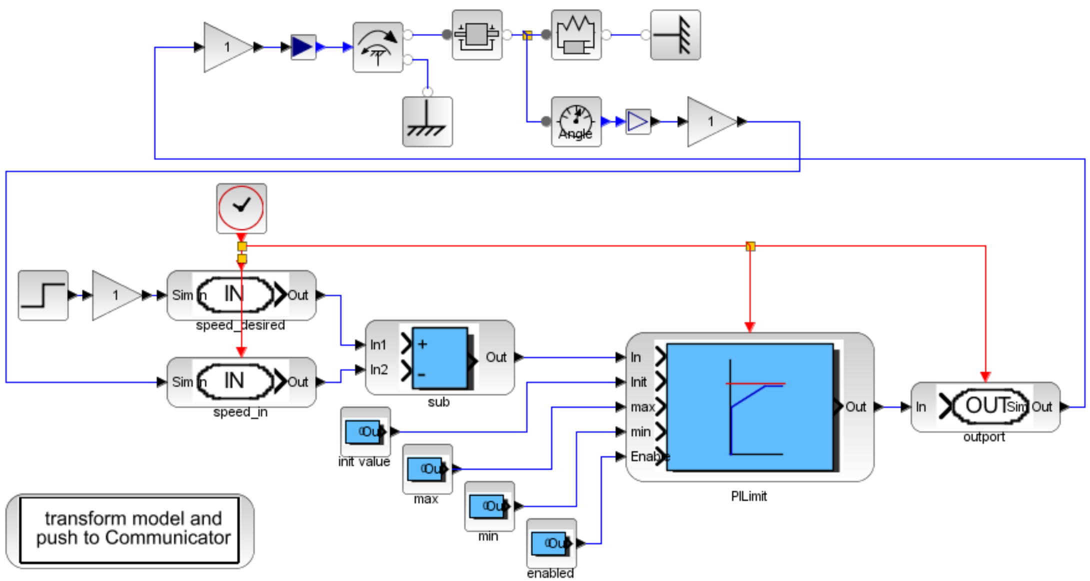
- Existing code generators for the outdated Scilab/Scicos:
  - RTAI [3]
  - Gene-Auto [4]
  - Scicos-FLEX [5]

- Code generators for Scilab/XCos:
  - Project-P [6]
  - **X2C from JKU-Linz (Upper Austria)** [2]

Code generator for Scilab/XCos
Code generator X2C

- The predecessor of X2C was developed more than 10 years ago at the JK-University Linz, Austria as a Simulink extension

- X2C natively includes into XCos and can be simulated in parallel with plant and controller



Code generator for Scilab/XCos

Code generator X2C

- X2C-blocks are full featured XCos-blocks extended with an parameter editor and the connection to the back-end for code generation
- In XCos simulation the X2C-blocks are implementing exactly the code which will run on the target
- Model transformation and code generation is executed by a simple mouse click. All non-X2C-blocks are ignored during this process.

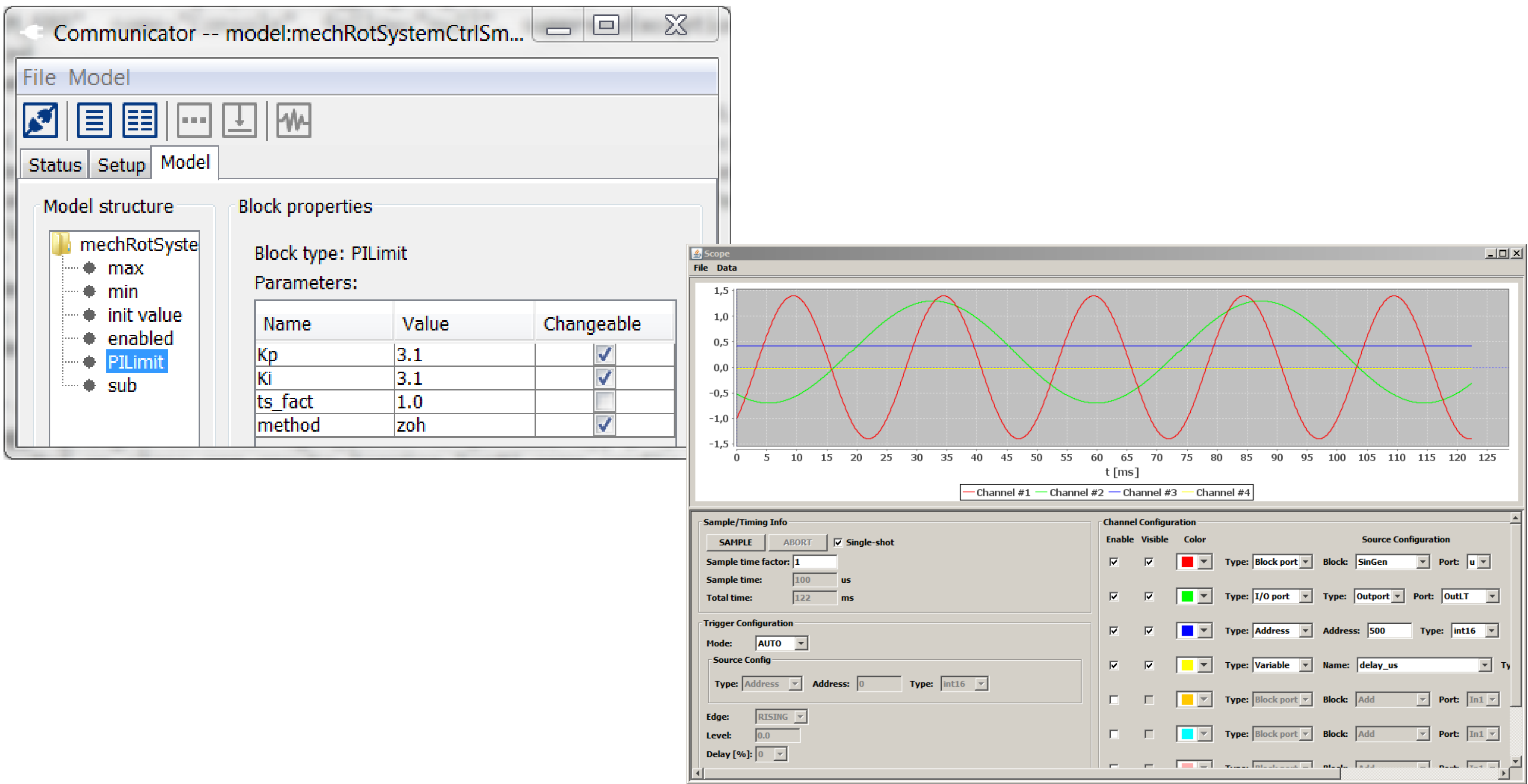
Code generator for Scilab/XCos
Code generator X2C

- The central tool is the so called „Communicator“. It‘s the interface between simulation environment and target.

- The Communicator features
  - Code generation
  - Change parameters in the model or in the communicator, and the parameters on the target are updated instantly
  - Scope (software oscilloscope)

Code generator for Scilab/XCos

Communicator and scope

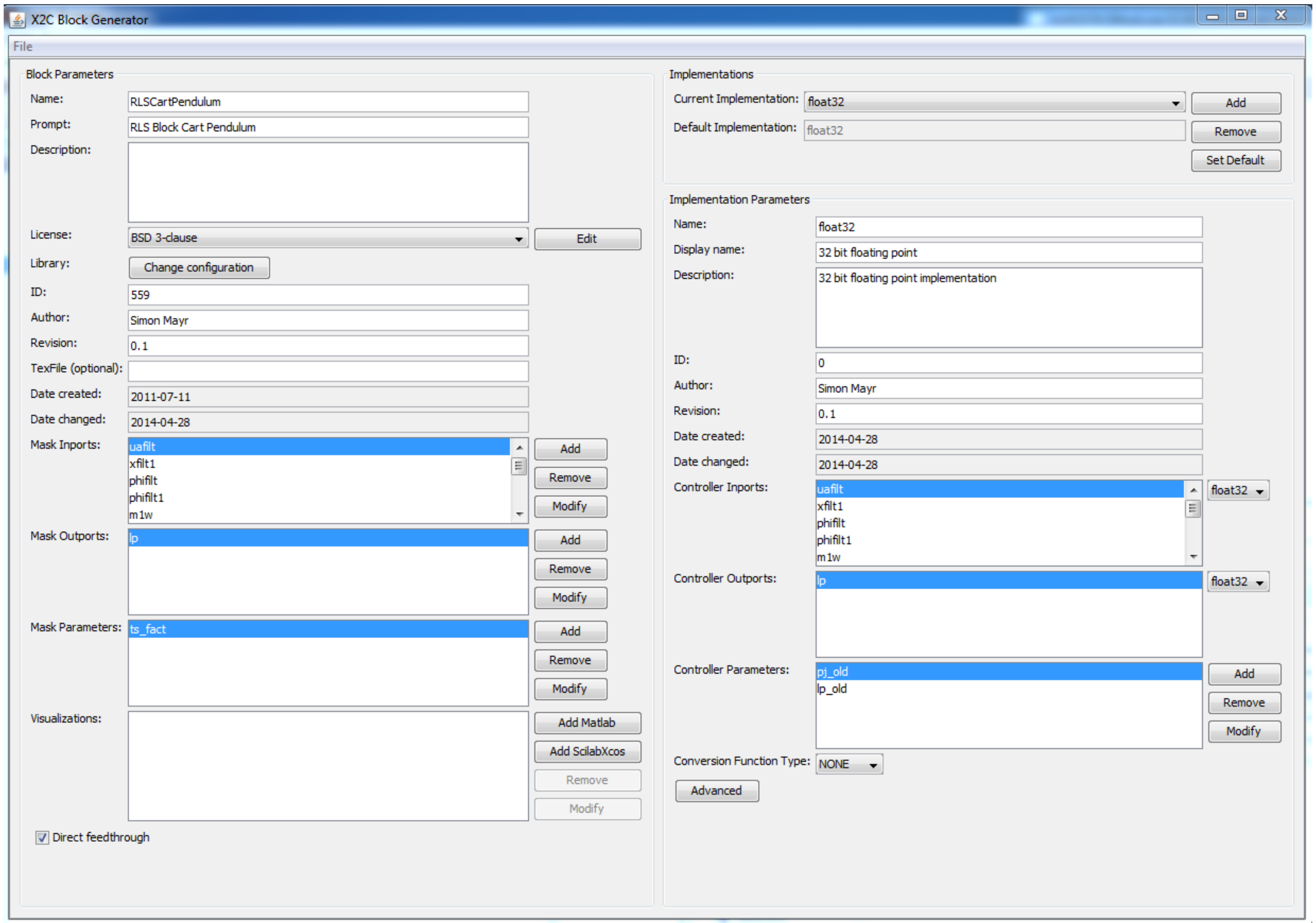


Code generator for Scilab/XCos
User defined X2C-blocks

- It’s possible to generate user defined X2C blocks easily with the help of a dedicated block generator
- Inputs, outputs, control parameters and data types are specified by the user
- This information is used to generate a code template automatically
- The behavior of the block is included by the user
- This blocks can be used for simulation and implementation on target

Code generator for Scilab/XCos

X2C-block generator



Code generator for Scilab/XCos

Code template and user code

```c
/* USERCODE-BEGIN:Description                                             */
/* Description: */
/* USERCODE-END:Description                                               */
#include "RLSCartPendulum_float32.h"

/* all used update functions to ram for c2000                             */
#if defined(__ALL_UPDATE_FUNC_2_RAM_C2000__)
     #pragma CODE_SECTION(RLSCartPendulum_float32_Update, "ramfuncs")
/* individual added update functions to ram for c2000                     */
#elif defined(__ALL_UPDATE_FUNC_SEPARATE_SECT_DEF_C2000__)
     #pragma CODE_SECTION(RLSCartPendulum_float32_Update, "RLSCartPendulum_float32_Update_Sect")
#endif

/* USERCODE-BEGIN:PreProcessor                                            */

/* USERCODE-END:PreProcessor                                              */

/*************************************************************************/
/** RLSCartPendulum_float32_Update                                      **/
/*************************************************************************/
void RLSCartPendulum_float32_Update(RLSCARTPENDULUM_FLOAT32 *pTRLSCartPendulum_float32)
{
/* USERCODE-BEGIN:UpdateFnc                                               */

/* USERCODE-END:UpdateFnc                                                 */

}

/*************************************************************************/
/** RLSCartPendulum_float32_Init                                        **/
/*************************************************************************/
void RLSCartPendulum_float32_Init(RLSCARTPENDULUM_FLOAT32 *pTRLSCartPendulum_float32)
{
     pTRLSCartPendulum_float32->ID = RLSCARTPENDULUM_FLOAT32_ID;
     pTRLSCartPendulum_float32->lp = 0;
/* USERCODE-BEGIN:InitFnc                                                 */

/* USERCODE-END:InitFnc                                                   */
}

/*************************************************************************/
/** RLSCartPendulum_float32_Load                                        **/
/*************************************************************************/
uint8 RLSCartPendulum_float32_Load(const RLSCARTPENDULUM_FLOAT32 *pTRLSCartPendulum_float32, uint8 data[])
{
     uint8 loadSize;

     data[0] = (uint8)((*(uint32*)&(pTRLSCartPendulum_float32->pj_old)) & 0x000000FF);
     data[1] = (uint8)((*(uint32*)&(pTRLSCartPendulum_float32->pj_old) >> 8) & 0x000000FF);
     data[2] = (uint8)((*(uint32*)&(pTRLSCartPendulum_float32->pj_old) >> 16) & 0x000000FF);
     data[3] = (uint8)((*(uint32*)&(pTRLSCartPendulum_float32->pj_old) >> 24) & 0x000000FF);
     data[4] = (uint8)((*(uint32*)&(pTRLSCartPendulum_float32->lp_old)) & 0x000000FF);
     data[5] = (uint8)((*(uint32*)&(pTRLSCartPendulum_float32->lp_old) >> 8) & 0x000000FF);
     data[6] = (uint8)((*(uint32*)&(pTRLSCartPendulum_float32->lp_old) >> 16) & 0x000000FF);
     data[7] = (uint8)((*(uint32*)&(pTRLSCartPendulum_float32->lp_old) >> 24) & 0x000000FF);
     loadSize = (uint8)8;
/* USERCODE-BEGIN:LoadFnc                                                 */
/* USERCODE-END:LoadFnc                                                   */
     return (loadSize);
}
```

```c
/* USERCODE-BEGIN:Description                                             */
/* Description: */
/* USERCODE-END:Description                                               */
#include "RLSCartPendulum_float32.h"

/* all used update functions to ram for c2000                             */
#if defined(__ALL_UPDATE_FUNC_2_RAM_C2000__)
     #pragma CODE_SECTION(RLSCartPendulum_float32_Update, "ramfuncs")
/* individual added update functions to ram for c2000                     */
#elif defined(__ALL_UPDATE_FUNC_SEPARATE_SECT_DEF_C2000__)
     #pragma CODE_SECTION(RLSCartPendulum_float32_Update, "RLSCartPendulum_float32_Update_Sect")
#endif

/* USERCODE-BEGIN:PreProcessor                                            */
// Inputs
#define UAFILT          (*pTRLSCartPendulum_float32->uafilt)
#define XFILT1          (*pTRLSCartPendulum_float32->xfilt1)
#define PHI             (*pTRLSCartPendulum_float32->phi)
#define PHIFILT         (*pTRLSCartPendulum_float32->phifilt)
#define PHIFILT1        (*pTRLSCartPendulum_float32->phifilt1)
#define M1W             (*pTRLSCartPendulum_float32->m1w)
#define D1W             (*pTRLSCartPendulum_float32->d1w)
#define MP              (*pTRLSCartPendulum_float32->mp)
#define BETA            (*pTRLSCartPendulum_float32->beta)
#define G               (*pTRLSCartPendulum_float32->g)
#define A1              (*pTRLSCartPendulum_float32->a1)
#define A0              (*pTRLSCartPendulum_float32->a0)

// Outputs
#define LP              (pTRLSCartPendulum_float32->lp)

// Control parameters
#define LP_OLD          (pTRLSCartPendulum_float32->lp_old)
#define PJ_OLD          (pTRLSCartPendulum_float32->pj_old)

/* USERCODE-END:PreProcessor                                              */

/*************************************************************************/
/** RLSCartPendulum_float32_Update                                      **/
/*************************************************************************/
void RLSCartPendulum_float32_Update(RLSCARTPENDULUM_FLOAT32 *pTRLSCartPendulum_float32)
{
/* USERCODE-BEGIN:UpdateFnc                                               */
float32 WJ, YJ, PJ_NEW;
YJ = 6*D1W*XFILT1 - 6*G*(M1W + MP)*PHIFILT - 6*BETA*UAFILT;
WJ = (PHI - PHIFILT - (A1/A0)*PHIFILT1)*(A0*(MP + 4*M1W));

PJ_NEW = PJ_OLD*(1-(PJ_OLD*WJ*WJ)/(1+WJ*WJ*PJ_OLD));
LP = LP_OLD+(PJ_OLD*WJ*(YJ-WJ*LP_OLD))/(1+WJ*WJ*PJ_OLD);

PJ_OLD = PJ_NEW;
LP_OLD = LP;

/* USERCODE-END:UpdateFnc                                                 */

}
```

Experiment
Cart and pendulum

- System modeling
- Plant simulation
- Parameter identification (pendulum length)
- Adaptive STC control
- Code generation
- Measurements

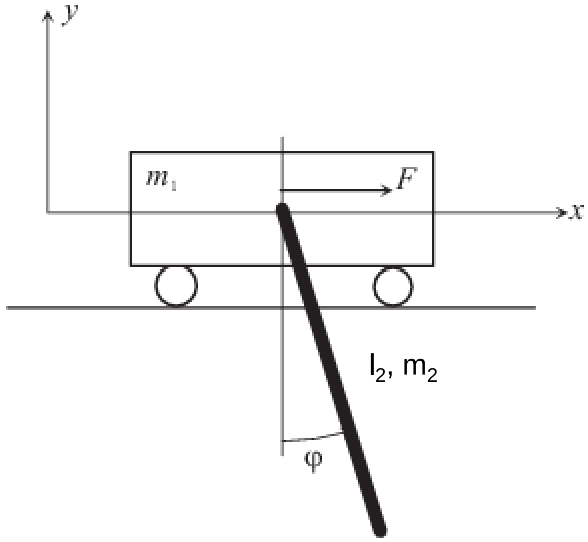
Plant modeling
Cart and pendulum

- Plant modeling can be either done by using mathematical terms (e.g. ODEs) or by using the Modelica-based Scilab-addon „Coselica“

- Plant modeling is shown by the cart and pendulum example.

m_1 ... vehicle mass
d_1 … linear friction coefficient (cart)
m_2 … pendulum mass
d_2 … linear friction coefficient (pendulum)
l_2 … pendulum length

x … distance (cart)
v ... velocity (cart)
φ … angle (pendulum)
ω … angular velocity (pendulum)



Plant modeling
Cart and pendulum

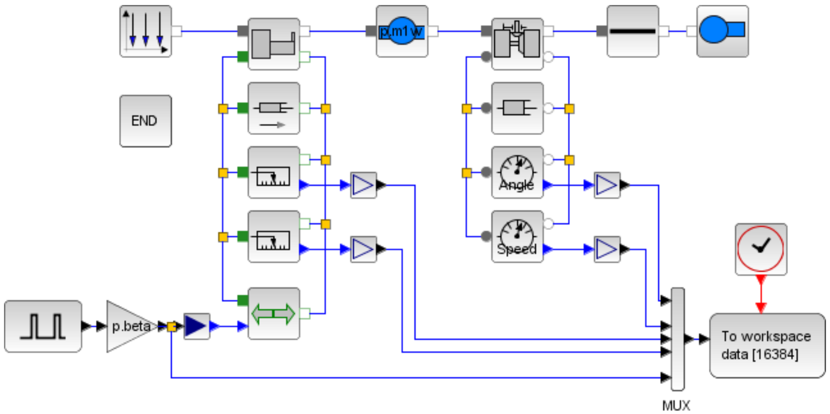
- Nonlinear system equations can be computed using the Lagrange formalism with the vector of generalized coordinates $\boldsymbol{q} = [x, \varphi]^T$ and $F = \beta u_A$. Furthermore static friction F_C is ignored, because it's compensated.

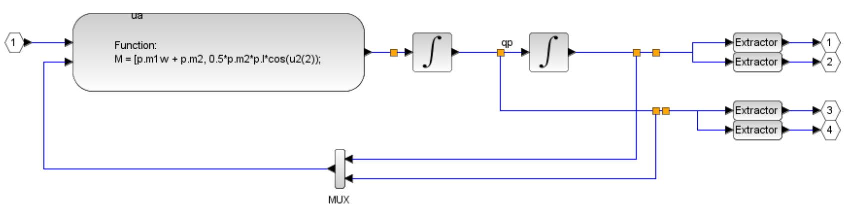
$$\begin{bmatrix} \tilde{m}_1 + m_2 & \frac{1}{2} m_2 l_2 cos\left(\varphi\right) \\ \frac{1}{2} m_2 l_2 cos\left(\varphi\right) & \frac{1}{3} m_2 l_2^2 \end{bmatrix} \cdot \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} m_2 l_2 sin\left(\varphi\right) \omega^2 - \tilde{d}_1 v + \beta u_A \\ -\frac{1}{2} m_2 g l_2 sin\left(\varphi\right) - d_2 \omega \end{bmatrix}$$

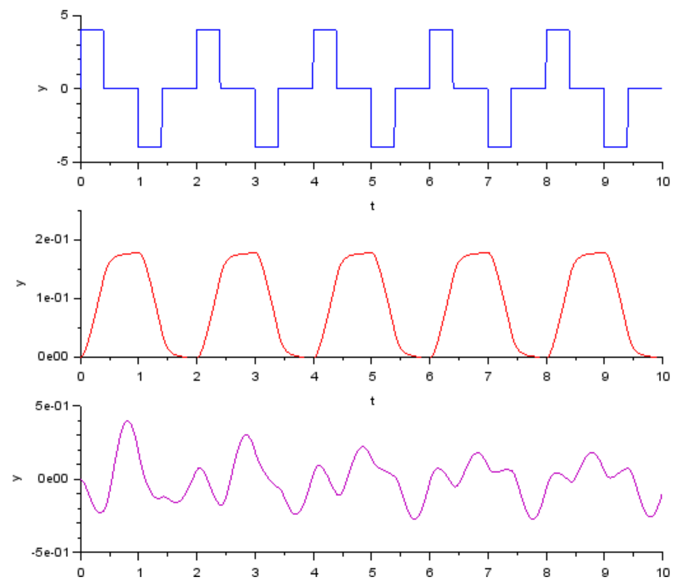
- The linearized model (around $q_S = [x_S, \varphi_S, v_S, \omega_S]^T = [0, k\pi, 0{,}0]^T$, k = 0,2,…) can be written as

$$\begin{bmatrix} \Delta\dot{x} \\ \Delta\dot{\varphi} \\ \Delta\dot{v} \\ \Delta\dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{3gm_2}{4\tilde{m}_1 + m_2} & -\frac{4\tilde{d}_1}{4\tilde{m}_1 + m_2} & \frac{6d_2}{l_2(4\tilde{m}_1 + m_2)} \\ 0 & -\frac{6g(\tilde{m}_1 + m_2)}{l_2(4\tilde{m}_1 + m_2)} & \frac{6\tilde{d}_1}{l_2(4\tilde{m}_1 + m_2)} & -\frac{12d_2(\tilde{m}_1 + m_2)}{m_2 l_2^2 (4\tilde{m}_1 + m_2)} \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta\varphi \\ \Delta v \\ \Delta\omega \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{4\beta}{4\tilde{m}_1 + m_2} \\ -\frac{6\beta}{l_2(4\tilde{m}_1 + m_2)} \end{bmatrix} \cdot u_A$$

Plant modeling
Coselica & ODE







Parameter identification

- Identifying the unknown (but constant) pendulum length l_2
- Assumption(s):
  - Pendulum friction is set to zero ($d_2 = 0$)
- The 4^{th} line of the linearized model is used for identification
- To get rid of the time derivatives, the system equation is transformed into the laplace-domain

$$s^2\hat{\varphi}l_2\left(4\tilde{m}_1 + m_2\right) = -6\beta\hat{u}_A - 6g\left(\tilde{m}_1 + m_2\right)\hat{\varphi} + 6\tilde{d}_1 s\hat{x}$$

Parameter identification

- We apply realizable stable filters $F_0(s)$ and $F_1(s) = sF_0(s)$ with free coefficients to the whole equation [1]

$$F_0(s) = \frac{\alpha_0}{s^2 + \alpha_1 s + \alpha_0} \qquad F_1(s) = sF_0(s)$$

- The inverse laplace transformation leads to one data line, linear in the unknown parameter (* indicates the convolution operator in time-domain)

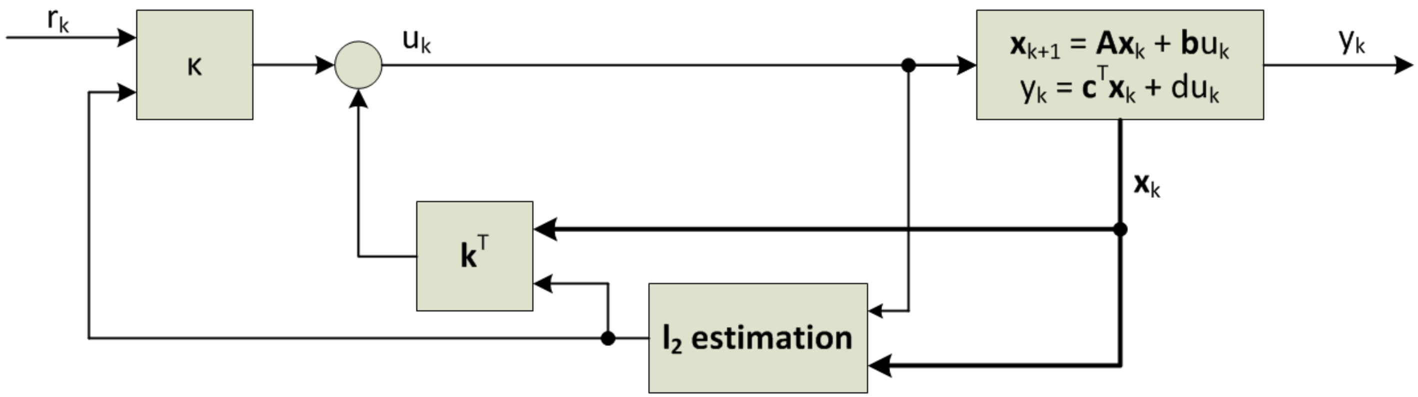
$$\left[\alpha_0 \left(\varphi - \tfrac{\alpha_1}{\alpha_0} F_1 * \varphi - F_0 * \varphi\right)(4\tilde{m}_1 + m_2)\right]\left[\theta_1\right] = -F_0 * u_A 6\beta - F_0 * \varphi 6g(\tilde{m}_1 + m_2) + F_1 * x 6\tilde{d}_1$$

- Estimation of the parameter using recursive least square algorithm

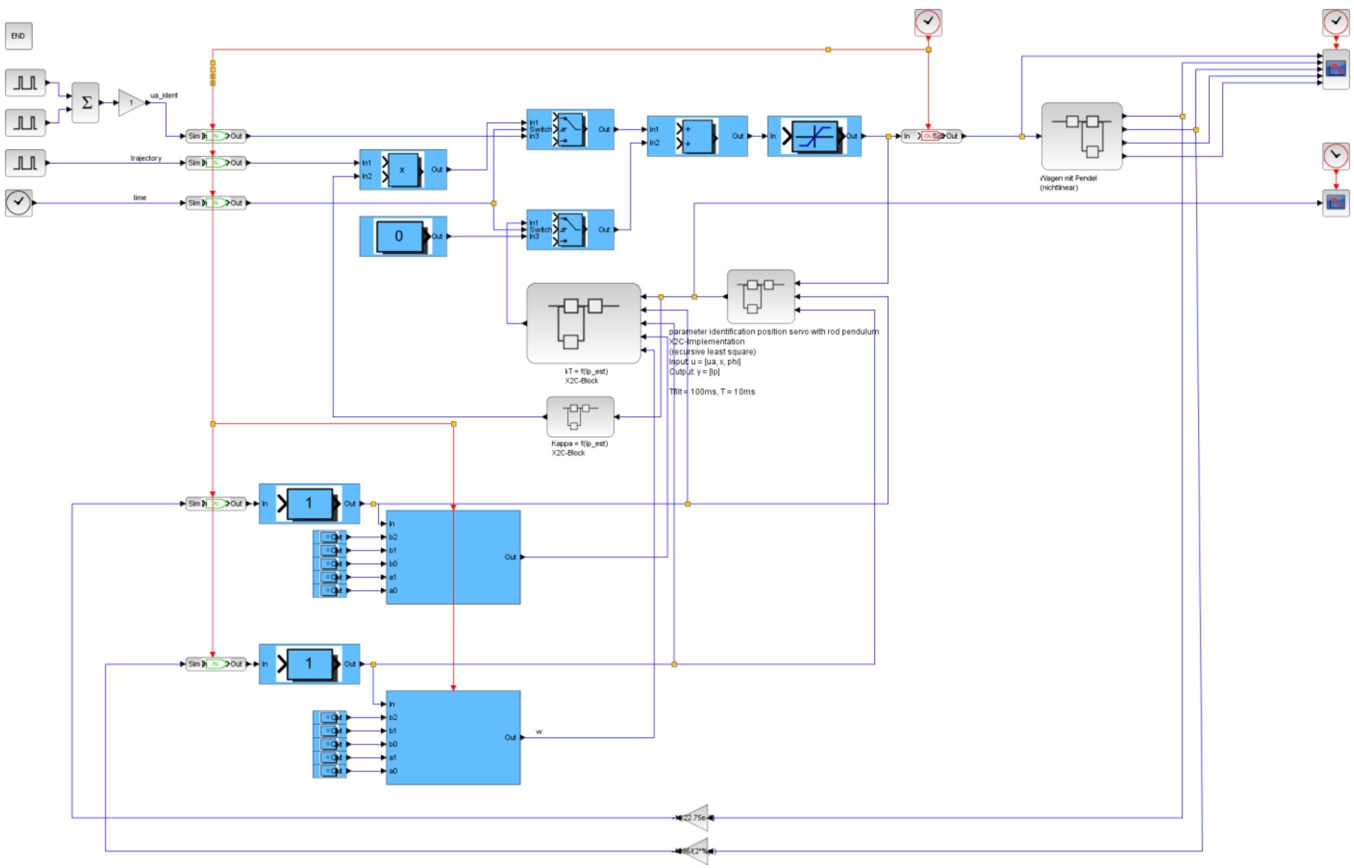
Adaptive STC control

- Design a linear state control law parameterized in pendulum length l2

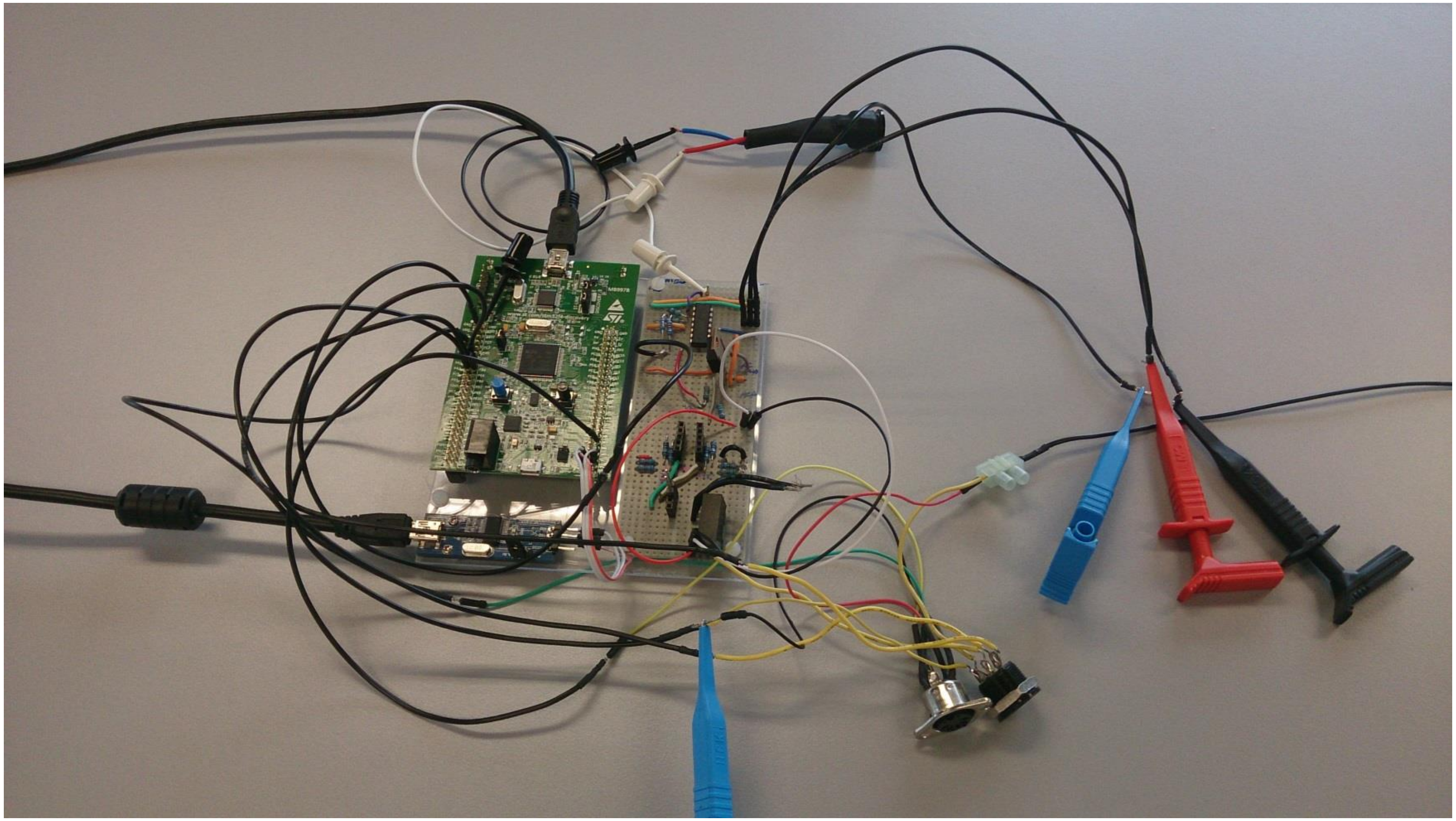
$$u_k = \mathbf{k}(\mathrm{l}_2)^{\mathrm{T}}\mathbf{x} + \kappa(l_2)r_k$$



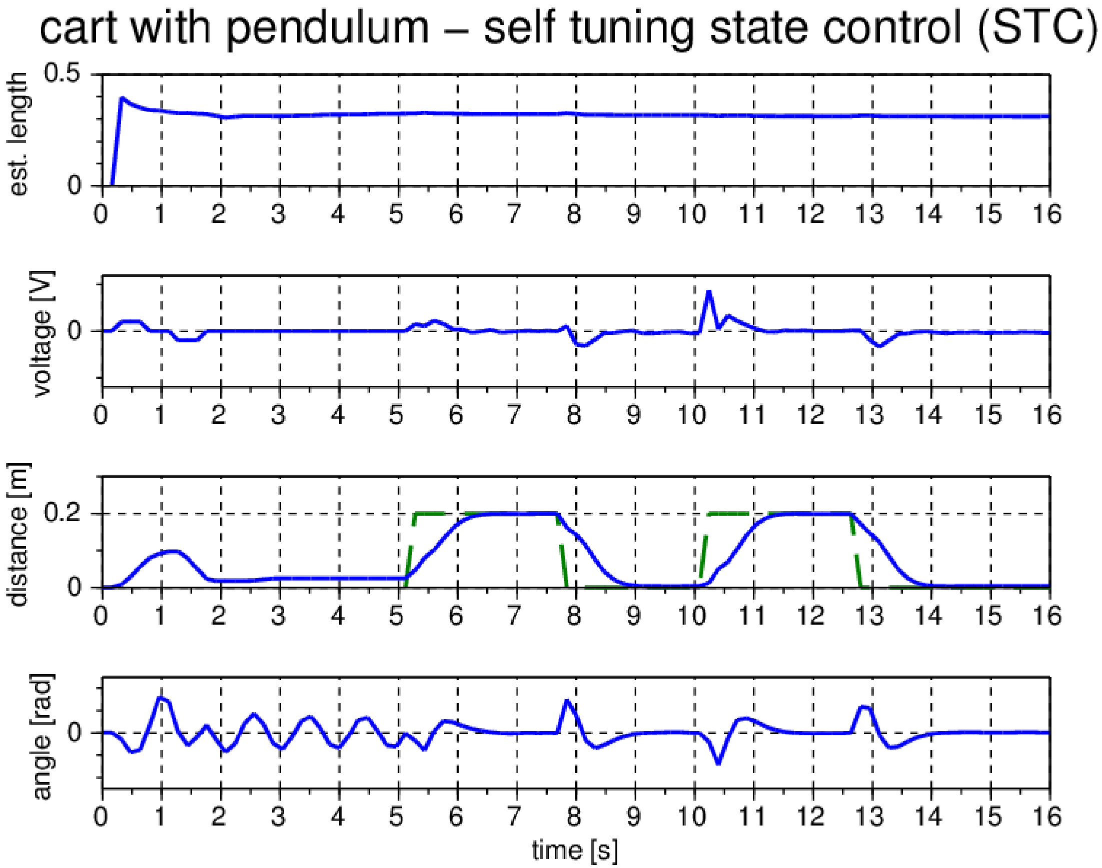
STC measurements



STC measurements
Discovery-board



STC measurements



Summary and outlook

- Complete free (or low cost if hardware is included) tool-chain based on Scilab/XCos

- Ongoing development is targeted towards
  - efficient handling of vectorized signal lines in X2C
  - more block libraries
  - Industrial targets
  - adaption of the FMI (functional mockup interface) for model exchange

Thank you for your attention!

References

- [1] JJE. Slotine, W. Li, Applied nonlinear control, Prantice-Hall, 1991
- [2] X2C in Scilab/XCos, 2013, http://www.mechatronic-simulation.org
- [3] Roberto Bucher, et al., RTAI-Lab tutorial: Scilab, Comedi and real-time control, 2006
- [4] Ana-Elena Rugina, et al., Gene-Auto: Automatic Software Code Generation for Real-Time Embedded Systems, DASIA 2008
- [5] Scicos-FLEX code generator, http://erika.tuxfamily.org/drupal/scilabscicos.html
- [6] Project-P code generator, http://www.open-do.org/projects/p/