D5.1.2 - Development of an FMI-based standardized interface for executing NMPC within virtual environments

WP 5.1 Nonlinear Model Predictive Control
WP 5 Optimized system operation

MODRIO (11004)

Version 2.0

Date 21/04/2016

Authors

Andreas Pfeiffer	DLR
Johan Akesson	Modelon

Executive summary

This document motivates and specifies the requirements for the Functional Mock-Up Interface FMI 2.0 to execute Nonlinear Model Predictive Control (NMPC) using this interface. The FMI defines interfaces for Model Exchange and Co-Simulation. To apply NMPC algorithms to FMI models both interfaces are not sufficient.

A prototype implementation of NMPC algorithms using Dymola's Optimization library causes some additional features of FMI for Co-Simulation, especially to get access to the models states also for the Co-simulation interface.

NMPC algorithms often rely on gradient based optimization methods. For successful application of these methods accurate derivatives of the optimization objective and constraint functions are essential. According to this request some additional requirements to FMI are analyzed with respect to partial derivatives of some of the FMI functions.

Contents

1. Nonlinear Model Predictive Control (NMPC)

Nonlinear Model Predictive Control is a control technique that is based on model information predicting into the future. A nonlinear dynamical model (e.g. a Modelica model) describes a physical system to be considered. The system behavior is influenced by some model input functions $u(t)$ (= the controls). The goal in each discrete controller step is to minimize some kind of objective function in order to determine an optimal trajectory for the controls in the near future of the current time. The optimization problem typically has some inequality and equality constraints, too.

1.1. Optimization problems for NMPC

The following general NMPC optimization problem is the considered problem class:

$$\min_{u,p,t_0,t_f} \phi\left(\bar{z},p,t_0,t_f\right) + \int_{t_0}^{t_f} L(\dot{x}(t),x(t),w(t),u(t),t,p)\,dt$$
$$F(\dot{x}(t),x(t),w(t),u(t),t,p) = 0, t \in \left[t_0,t_f\right], F \in C^2$$
$$F_0(\dot{x}(t_0),x(t_0),w(t_0),u(t_0),t,p) = 0, t \in \left[t_0,t_f\right], F_0 \in C^2$$
$$C_{eq}(\dot{x}(t),x(t),w(t),u(t),t,p) = 0,\ t \in \left[t_0,t_f\right], C_{eq} \in C^2$$
$$C_{ineq}(\dot{x}(t),x(t),w(t),u(t),t,p) \le 0, t \in \left[t_0,t_f\right], C_{ineq} \in C^2$$
$$H_{eq}\left(\bar{z},p,t_0,t_f\right) = 0, H_{eq} \in C^2$$
$$H_{ineq}\left(\bar{z},p,t_0,t_f\right) \le 0, H_{ineq} \in C^2$$
$$\bar{z} = \left[\dot{x}(t_1),x(t_1),w(t_1),u(t_1),\dots,\dot{x}(t_{n_p}),x(t_{n_p}),w(t_{n_p}),u(t_{n_p})\right]$$
$$x_{min} \le x(t) \le x_{max}, w_{min} \le w(t) \le w_{max}, u_{min} \le u(t) \le u_{max}$$

1.2. Numerical Methods

Several numerical methods and techniques can be applied to approximately solve a Nonlinear Model Predictive Control task. Three main categories of methods can be identified:

- Single Shooting Technique,
- Multiple Shooting Technique,
- Collocation Technique.

All of the methods apply numerical optimization algorithms, especially gradient based ones (e.g. Sequential Quadratic Programming SQP) to reach good convergence properties. The gradient based optimization algorithms need derivatives of the objective function and derivatives of the constraint functions of the optimization problem. The derivatives are with respect to the free optimization variables, like parameters or control functions.

Concerning the dynamical part of the equations, this means that the sensitivities of the model states with respect to model parameters have to be computed. Several methods for a numerical approximation of the sensitivities exist. A well-proven technique is the numerical solution of the sensitivity equations. Section 2 deals with the sensitivity analysis of hybrid systems to show the potential of this approach for NMPC also with hybrid dynamical systems.

1.3. Prototype Implementation of NMPC using FMI

A prototype implementation of nonlinear model predictive control (NMPC) based on the Optimization Library [6] in Dymola has been developed and is described in D5.2.3 [2].

The concept how to realize an NMPC in Modelica and Dymola is partially based on the process to generate nonlinear Kalman filters described in the D3.1.2 [1]. Basically, a Modelica plant model (the *Prediction Model*) has to be provided by the user. By the implemented process this model is automatically exported as FMU 2.0 for Co-Simulation with additional features described in [1] (Section 3.2). Then, the FMU is imported as Modelica package into Dymola by using a template package that provides defined interfaces to all functions of the FMU.

The goal is to have a Modelica model of the nonlinear model predictive controller that uses the imported FMU for repeated predictions for a given length of time inside of the optimization loop. The controller model calls an optimization routine to solve the trajectory optimization problem at every given sample time.

The implemented process automatically generates a Modelica package containing the imported FMU and the model predictive controller model within a system model. It means the system model can be directly used to test the NMPC.

2. Sensitivity Analysis of Hybrid Systems

From [7]: "Practical models mostly contain a lot of parameters to calibrate the model behavior in comparison to measurements on a test bench. In addition to the solution of the corresponding dynamical system also the sensitivity of the solution with respect to small variations of model parameters is of interest. From an analytical point of view the sensitivities are derivatives of a parameter depending solution […] with respect to the model parameters […]. Based on the sensitivity analysis for systems with smooth right hand sides we investigate the existence of sensitivities for discontinuous systems with structures that are typical of models in practical applications."

2.1. Hybrid ODE System

The following smooth ODE system is considered for $t_0 \le t \le t_1$:

$$\begin{aligned} \dot{x} &= f(x,d,u,p,t), & x(t_0) &= X(u,p,t_0), \\ y &= h(x,d,u,p,t), & d(t_0) &= D(u,p,t_0), \\ d &= d(t_0). \end{aligned} \tag{2.1}$$

An explanation of all used variables and functions is given in Sections 2.5 and 2.6. All of the functions may be implicitly defined in the model. But they have to be sufficiently smooth with respect to all variables, if sensitivity analysis shall be applied.

System (2.1) describes the hybrid model equations until a non-smoothness is appearing. This is indicated by zeros of switching functions. It is assumed that exactly one switching function q is being active (i.e. has a zero-crossing) at the same time: $q(x(t^*), d(t^*), u(t^*), p, t^*) = 0$. This time t^* we call an event.

At an event different kinds of discontinuous changes are possible:

a. The right hand side f of the ODE system may non-smoothly (e.g. discontinuously) change.
b. The continuous states x may change, i.e. they introduce discontinuity jumps (in Modelica by the reinit operator).
c. The continuous states change their meaning and / or the dimension. New states $\bar{x}$ are introduced that describe the model behavior beyond the event.

Because the changes in c. are the most general ones that include the changes in a. and b., the following considerations are according to case c.

For the time after the event the model equations shall be described by a smooth system for $t^* \le t \le t^{**}$:

$$\begin{aligned} \dot{\bar{x}} &= \bar{f}(\bar{x},\bar{d},u,p,t), & \bar{x}(t^*) &= \bar{X}(x,d,u,p,t^*), \\ \bar{y} &= \bar{h}(\bar{x},\bar{d},u,p,t), & \bar{d}(t^*) &= \bar{D}(x,d,u,p,t^*). \end{aligned} \tag{2.2}$$

The transitions from system (2.1) to system (2.2) are formulated by the functions $\bar{X}$ and $\bar{D}$. The procedure of changing from one smooth system to another one indicated by the zero-crossing of a switching function can be repeated finitely many times until the end time of the overall time interval.

2.2. Sensitivity Equations for Hybrid Systems

The nomenclature $\mathrm{F_z} := \frac{\partial \mathrm{F}}{\partial \mathrm{z}}$ is used for a Jacobian matrix that can also be vector or a scalar depending on the dimensions of F and z. Parts of the following equations are according to [8], [3] and [5]. We consider the hybrid system (2.1) + (2.2) from the last paragraph and formulate the sensitivity equations for system (1), $t_0 \le t \le t^*$:

$$\begin{aligned} &\dot{x}_p = f_x x_p + f_d d_p + f_u u_p + f_p, \quad x_p(t_0) = X_u u_p + X_p, \\ &y_p = h_x x_p + h_d d_p + h_u u_p + h_p, \quad d_p(t_0) = D_u u_p + D_p, \\ &d_p = d_p(t_0). \end{aligned} \tag{2.3}$$

In this formulation the case that the input variables depend on the parameters is also included. If there is no dependency of u on p, then $u_p = 0$ applies.

For an event the sensitivity of the switching time t^* can be computed by:

$$t_p^* = -(q_x x_p + q_d d_p + q_u u_p + q_p)(q_x f + q_u u_t + q_t)^{-1}$$

under the assumption that $q_x f + q_u u_t + q_t \neq 0$ holds for $t = t^*$.

The sensitivity equations for system (2.2), $t^* \le t \le t^{**}$ are as follows:

$$\begin{aligned} &\dot{\bar{x}}_p = \bar{f}_{\bar{x}} \bar{x}_p + \bar{f}_{\bar{d}} \bar{d}_p + \bar{f}_u u_p + \bar{f}_p, \\ &\bar{y}_p = \bar{h}_{\bar{x}} \bar{x}_p + \bar{h}_{\bar{d}} \bar{d}_p + \bar{h}_u u_p + \bar{h}_p, \\ &\bar{d}_p = \bar{d}_p(t^*), \\ &\bar{x}_p(t^*) = \bar{X}_x x_p + \bar{X}_d d_p + \bar{X}_u u_p + \bar{X}_p + \left(\bar{X}_x f + \bar{X}_u u_t + \bar{X}_t - \bar{f}\right) t_p^*, \\ &\bar{d}_p(t^*) = \bar{D}_x x_p + \bar{D}_d d_p + \bar{D}_u u_p + \bar{D}_p + (\bar{D}_x f + \bar{D}_u u_t + \bar{D}_t) t_p^*. \end{aligned} \tag{2.4}$$

Especially remarkable are the update formulas at the event $t = t^*$ for the sensitivities of the continuous states $\bar{x}$ and the discrete variables $\bar{d}$.

2.3. Special Cases

Sensitivity analysis of hybrid systems is generally more complicated than solving the hybrid system itself. There are different additional aspects to be taken into account:

- If $q_x f + q_u u_t + q_t = 0$ holds at the event, then the sensitivities cannot be directly computed. Further complicated analysis is necessary.
- More than one switching function may be active, see [7] for an example and [5] for the analysis. In this case further structural analysis of the equations is necessary. One result can be that the sensitivities do not exist for the time $t > t^*$.
- Smoothness of the switching function q can not be guaranteed for all practical models, see [7] for an example and [5] for the analysis of such systems.

2.4. Conclusion

The problem class of hybrid systems that can be solved numerically is greater than the problem class of hybrid systems for those sensitivities exist or can be computed numerically robust. Always, there will be the problem which systems shall be supported by the sensitivity analysis. One suggestion is to implement the following constitutive steps to enlarge the problem class:

1. Smooth system (2.3).
2. Hybrid system (2.3) + (2.4) with possibly discontinuous right hand side, i.e. $\bar{X} := x$, no discrete variables and no inputs.
3. Allowing inputs.
4. Allowing discrete variables.
5. Allowing discontinuities in states, i.e. $\bar{X} \neq x$.

2.5. List of Variables

x	Vector of continuous states
d	Vector of discrete real variables
u	Vector of real input variables
p	Vector of continuous parameters for that sensitivities shall be computed
t	Time
y	Vector of continuous algebraic variables
t_0	Start time
t^*	Time of first switching event
t^{**}	Time of second switching event or end time of overall time interval
$\bar{x}$	New vector of continuous states after first switching event
$\bar{d}$	New vector of discrete real variables after first switching event
$\bar{y}$	New vector of algebraic continuous variables after first switching event

2.6. List of Functions

f	Right hand side of ODE system
h	Evaluation function of continuous algebraic variables
X	Function to compute initial values of continuous states
D	Function to compute initial values of discrete real variables
q	First active switching function
$\bar{f}$	Right hand side of new ODE system after first switching event
$\bar{h}$	Evaluation function of new algebraic variables after first switching event
$\bar{X}$	Function to compute values of new states at first switching event
$\bar{D}$	Function to compute values of discrete variables at first switching event

3. Interface requirements for FMI to execute NMPC

The discussed methods and prototypes in the previous sections result in the following requirements for FMI to execute NMPC.

3.1. Partial Derivatives

According to the conclusion in Section 2.4 in a first step the smooth continuous system is considered. To solve the sensitivity equations of the smooth continuous system (2.3), the FMI interface [4] has to be extended in order to get access to the following partial derivatives:

- f_d, f_p,
- X_u, X_p,
- h_d, h_u, h_p,
- D_u, D_p.

3.2. Extended Co-Simulation Interface

The prototype implementation of NMPC based on the Optimization Library in Dymola, see Section 1.3, showed, that FMI 2.0 is not fully sufficient to be used for NMPC algorithms. The same requirements necessary for state estimation algorithms discussed in [1] are also suitable for NMPC. Therefore, the requirements are cited in the following:

The standard FMI 2.0 for Co-Simulation interface allows integrating the following dynamical equations

$$\begin{aligned} &\dot{x} = f(x,u,t), \quad x(t_0) = x_0, \\ &y = h(x,u,t) \end{aligned}$$

from sample time t_{k-1} to the next sample time t_k with the FMI function `fmi2DoStep(...)`. In standard co-simulation the continuous-time states x of a model are hidden in the co-simulation slave. However, for NMPC algorithms the states need to be explicit and it must be possible to reset the states at sample instants.

Especially,

- the continuous-time states shall be reported in the `modelDescription.xml` file under element `ModelStructure`,
- it shall be possible to explicitly set the continuous-time states with `fmi2SetReal(..)` before `fmi2DoStep(..)` is called,
- it shall be possible to inquire the actual values of all variables with `fmi2GetReal(..)` after `fmi2SetReal(..)` was called, without an `fmi2DoStep(..)` in between,
- when importing an FMU for Co-Simulation into a Modelica simulation environment, the tool should generate the Modelica code optionally according to a given template package. This package serves as interface to access the needed FMI functionality from a Modelica model or function. More details according this requirement can be found in D3.1.2 [1].

4. References

[1] Deliverable D3.1.2: *Standardized interfaces for continuous-time models as needed for state and parameter estimation in the MODRIO tool chains*, ITEA2 project MODRIO (11004), Version 2.0, 2014.

[2] Deliverable D5.2.3: *Prototype for optimization toolchain in Dymola-Optimization*, ITEA2 project MODRIO (11004), Version 1.0, 2016.

[3] Galan, S., W. F. Feehery and P. I. Barton: *Parametric sensitivity functions for hybrid discrete/continuous systems.* Applied Numerical Mathematics, 31(1):17–47, 1999. http://dx.doi.org/10.1016/S0168-9274(98)00125-1

[4] Modelica Association Project "FMI": *Functional Mock-up Interface for Model Exchange and Co-Simulation*, Version 2.0, July 25 2014. www.fmi-standard.org

[5] Pfeiffer, A.: *Numerische Sensitivitätsanalyse unstetiger multidisziplinärer Modelle mit Anwendungen in der gradientenbasierten Optimierung.* Fortschrittberichte VDI, Reihe 20, Nr. 417, VDI Verlag, Düsseldorf, 2008. http://elib.dlr.de/54759

[6] Pfeiffer, A.: *Optimization Library for Interactive Multi-Criteria Optimization Tasks*. Proc. of 9th International Modelica Conference, pp. 669-679, Munich, Germany, Sept. 2012.

[7] Pfeiffer, A. and M. Arnold: *Sensitivity Analysis of Discontinuous Multidisciplinary Models: Two Examples.* In Thomsen, P.G. and H. True (editors): *Non-smooth Problems in Vehicle Systems Dynamics.* Proceedings of the Euromech 500 Colloquium, pp. 239–251, Springer, 2010. http://link.springer.com/chapter/10.1007/978-3-642-01356-0_21

[8] Rozenvasser, E. N.: *General sensitivity equations of discontinuous systems.* Automat. Remote Control, pp. 400–404, 1967.