



Proceedings
of the 4th International Modelica Conference,
Hamburg, March 7-8, 2005,
Gerhard Schmitz (editor)

M. Thümmel, G. Looye, M. Kurze, M. Otter, J. Bals
DLR Oberpfaffenhofen, Germany
Nonlinear Inverse Models for Control
pp. 267-279

Paper presented at the 4th International Modelica Conference, March 7-8, 2005,
Hamburg University of Technology, Hamburg-Harburg, Germany,
organized by The Modelica Association and the Department of Thermodynamics, Hamburg University
of Technology

All papers of this conference can be downloaded from
<http://www.Modelica.org/events/Conference2005/>

Program Committee

- Prof. Gerhard Schmitz, Hamburg University of Technology, Germany (Program chair).
- Prof. Bernhard Bachmann, University of Applied Sciences Bielefeld, Germany.
- Dr. Francesco Casella, Politecnico di Milano, Italy.
- Dr. Hilding Elmqvist, Dynasim AB, Sweden.
- Prof. Peter Fritzson, University of Linkping, Sweden
- Prof. Martin Otter, DLR, Germany
- Dr. Michael Tiller, Ford Motor Company, USA
- Dr. Hubertus Tummescheit, Scynamics HB, Sweden

Local Organization: Gerhard Schmitz, Katrin Prölb, Wilson Casas, Henning Knigge, Jens Vasel, Stefan Wischhusen, TuTech Innovation GmbH

Nonlinear Inverse Models for Control

Gertjan Looye, Michael Thümmel, Matthias Kurze, Martin Otter, Johann Bals
 DLR Institute of Robotics and Mechatronics, Oberpfaffenhofen, Germany
 {Getjan.Looye, Michael.Thuemmel, Matthias.Kurze, Martin.Otter, Johann.Bals}@DLR.de

Abstract

A general technique to design advanced controllers for non-linear systems is described, using component oriented modeling and symbolic algorithms as used for Modelica models. Starting point are linear design techniques that use linear inverse models as a core part of the controller structure. Starting from such a structure, the approach is to replace the linear inverse model with a nonlinear one, resulting in a controller that is applicable over the full operating range of the (nonlinear) plant. It is shown that nonlinear inverse models may be automatically generated from the plant model in Modelica.

1 Introduction

The subject of this article is the systematic design of controllers for *nonlinear* systems, based on inversion of the plant model. Traditional design techniques require the nonlinear plant model to be linearised around a stationary operating point, after which linear methods may be applied to synthesize a controller. In order to make this controller work over the full operating range of the plant, robust design techniques and/or gain scheduling are applied. The first approach may considerably reduce achievable performance if the plant dynamics vary strongly over the operating range, whereas the latter may involve designing many controllers at a grid of operating points and finding an interpolation scheme in between them.

In linear design, inversion of plant dynamics is sometimes used to compensate for coupled input / output responses, or as an easy way to impose specific dynamic behavior of the closed-loop system [7]. Provision is that the linear plant model is minimum phase and, for some structures, stable. In a nonlinear context, the application of model inversion additionally provides compensation of nonlinear dynamic

behavior of the plant. This is exploited in design techniques such as feedback-linearization [19].

The design approach in this article starts from any controller structure that is based on a linear inverse model of the plant. This model is replaced with a nonlinear inverse one, resulting in a controller that is valid for the full operating range of the plant. In case the plant model is available in Modelica, it will be demonstrated that inversion can be performed automatically, exploiting symbolic algorithms and code generation features of a Modelica simulation environment. This allows for a highly automated design process that directly results in nonlinear controllers that work in all operating conditions of the plant, avoiding the need for gain scheduling.

This article is structured as follows. First general aspects of nonlinear inverse models are reviewed, as well as the possibility to derive these automatically from Modelica. In section 4, a number of common controller structures are discussed, for which the described design approach is applied. Next, a design example will be discussed. In section 6 a number of common problems in deriving and applying nonlinear inverse models will be described, as well as possible solutions or workarounds.

2 Inversion of Nonlinear Models

The goal is to use a nonlinear plant model in a controller in order that the nonlinearities of the plant are directly taken care of in the control system. For linear systems, several control structures are known where an inverse plant model is part of the controller. A single-input-single-output plant might be described as transfer function

$$y = \frac{n(s)}{d(s)}u \quad (1)$$

where “u” is the plant input, “y” is the plant output, “n(s)” is the numerator and “d(s)” is the denominator

of the transfer function. In section 4 several control structures will be investigated where an *inverse model*

$$u = \frac{d(s)}{n(s)} y \quad (2)$$

is part of the controller. Basically, the plant and the inverse plant model “cancel” each other due to the connection structure and by additional control blocks a desired transfer function of the closed loop system can be achieved. Although, this seems to be quite a “brute” force method, it will be shown that by appropriate adaptations practically useful control systems can be designed.

The essential idea is as follows:

1. Take any control structure for linear systems that utilizes a *linear inverse* plant model.
2. Replace the linear inverse plant model by a more detailed *nonlinear inverse* plant model.
3. Determine the remaining part of the control system by appropriate techniques, e.g., by tuning controller coefficients via parameter optimization.

Several different controller structures according to this technique will be discussed in section 4. The difficult part is issue (2): The nonlinear plant model should be constructed in a convenient way and the inverse model should be directly derived from the plant model. It turns out that Modelica is very well suited for this approach because Modelica is designed to model complex systems, and since Modelica tools, like Dymola [5], can generate nonlinear inverse models *automatically*:

A continuous Modelica model is primarily mapped to a DAE (= set of Differential Algebraic Equations) of the form:

$$\mathbf{0} = \mathbf{f}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{y}, \mathbf{u}) \quad (3)$$

where $\mathbf{x}(t)$ are variables that appear differentiated in the model, $\mathbf{y}(t)$ are algebraic and $\mathbf{u}(t)$ are known input functions of time t . It is possible to transform system (3) to the following state space form, at least numerically:

$$\begin{bmatrix} \dot{\mathbf{x}}_1 \\ \mathbf{x}_2 \\ \mathbf{y} \\ \mathbf{w} \end{bmatrix} = \mathbf{f}_1(\mathbf{x}_1, \mathbf{u}) \quad (4)$$

where \mathbf{x}_1 and \mathbf{x}_2 form vector \mathbf{x} such that the subset vector \mathbf{x}_1 is the *state vector* and contains the independent variables of \mathbf{x} . The new vector \mathbf{w} contains higher order derivatives of $\dot{\mathbf{x}}$ and of \mathbf{y} that appear

when differentiating equations of $\mathbf{f}(\cdot)$ and that are treated as algebraic variables. For the computation of $\mathbf{f}_1(\cdot)$, it might be necessary to solve linear and/or non-linear algebraic systems of equations. The equations to be differentiated can be determined with the algorithm of Pantelides [15]. The selection of the state variables \mathbf{x}_1 can be performed with the “dummy derivative method” of Mattsson and Söderlind [16]. Both algorithms are, for example, available in the Modelica simulation environment Dymola [5].

An *inverse* model of the DAE (3) is constructed by exchanging the meaning of variables: A subset of the *input* vector, \mathbf{u}_{inv} , with dimension n_{inv} , is treated no longer as known but as *unknown*, and n_{inv} previously unknown variables from the vectors \mathbf{x} and/ or \mathbf{y} are treated as *known* inputs. The result is still a DAE which can be handled with the same methods as any other DAE. Examples are given in the following sections. This technique of constructing non-linear inverse models has been first applied in [17][18]. An inverse model can only be used in a controller if the DAE of the inverse model has a *unique solution* and if it is *stable*. For linear systems the latter requirement means that the plant must be a minimum phase system. In section 6 it is discussed how to proceed if these requirements are not fulfilled.

Since the transformation from (3) to (4) might differentiate equations, the known inputs of the *inverse* model may be differentiated too, i.e., the derivatives of these inputs must exist and must be provided analytically up to a certain order. These derivatives can be provided if, e.g., the inputs are available as analytic functions that can be differentiated sufficiently often, or by a desired reference model that in combination with the inverse DAE results in a DAE that does not require derivatives of inputs. Often, the reference model is selected as a filter such that a combination of the filter states yields the needed derivatives. For linear systems, this approach is well known. Take for example the following linear system with one zero and two poles:

$$y = \frac{s+1}{(s-2) \cdot (s+3)} u \quad (5)$$

The inverse model is constructed as

$$u = \frac{(s-2) \cdot (s+3)}{(s+1)} \cdot \frac{1}{(Ts+1)} y \quad (6)$$

In order that the inverse model is causal (i.e., can be implemented as an algorithm), additional poles have to be added until the degree of the denominator is larger or, at least, as large as the degree of the numerator. For this reason, a filter $1/(Ts+1)$ has been connected in series, making the combined transfer

function causal. Another possibility is not to control y , but one of its derivatives instead:

$$\dot{y} = s \cdot y = \frac{s(s+1)}{(s-2) \cdot (s+3)} \cdot u$$

The transfer function is proper now and may be inverted:

$$u = \frac{(s-2) \cdot (s+3)}{s(s+1)} \dot{y}$$

Connecting this controller with the plant (5) in series results in integrator behavior. A simple feedback loop may be added to place the integrator pole at a desired location.

3 Constructing Inverse Models

With a Modelica simulation environment, such as Dymola, the practical derivation of inverse models is straightforward, even for complex systems:

1. Define the plant as Modelica model and include input and output signals of the plant over which the inversion shall take place.
2. If necessary, provide a reference model or input filter of appropriate relative degree. The relative degree may be known from physical knowledge of the plant dynamics, or can be automatically derived by Dymola as described below.
3. Connect the “u1” inputs of a “Modelica.Blocks.Math.TwoInputs” block to the plant outputs, the “u2” inputs of this block to the outputs of the reference model, and the input of this model to an input signal connector (Modelica.Blocks.Interfaces.RealInput) that defines the desired plant outputs.

A typical example is shown in Fig. 1. On the left side the plant model with one input and one output is

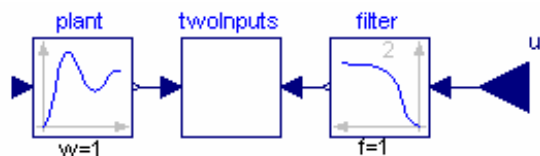


Fig. 1. Definition of inverse model with Modelica

present. On the right side, a filter is used as reference model. The output of the filter should be connected to the output of the plant. This is not directly possible, because signal connectors can only be connected according to block diagram semantic and in block diagrams it is not allowed to connect two output signals with each other. For this reason the “TwoInputs” block is used. It has two inputs $u1$ and $u2$ and

is described by the equation “ $u1 = u2$ ”. If the filter order is too low the DAE is not causal and Dymola prints an error message of the following form (Dymola version 5.3b and later):

Error: The model requires derivatives of some inputs as listed below:

Order of input derivative	
4	u1
2	u2
3	u3

Error: Failed to reduce the DAE index

In the second column the Modelica names of the input signals are listed that need to be differentiated according to the differentiation order of the first column. The numbers in the first column are therefore the minimum order of the corresponding filters.

If the inversion is to be based on a time derivative of the output, a sufficient number of integrators needs to be added, instead of increasing the filter order.

There is always a filter order / number of integrators for which the system will translate. The higher the filter order, the more problems will occur when applying it in a control system. In such cases, one might remove dynamic elements from the plant and try it again. One might even use a *stationary plant* model.

4 Example Controller Structures

In this section different controller structures will be discussed that follow the general approach outlined in section 2.

4.1 Inverse Model in Feedforward Path

Different variants of linear controllers with two structural degrees of freedom are known. The most general form for linear, single-input/single-output systems has been proposed and analyzed by Kreiselmeier [12]. According to the approach sketched in section 2, the generalization using nonlinear inverse models is shown in Fig. 2. This structure has been applied in [22] to the control of robots and has been successfully validated with hardware experiments. In flight control, the “model following approach”, see for example [2], is a special case of this structure whereby the reference model is known as the “command block” providing state references for the inverse model as well as the feedback controller.

In Fig. 2 the multi-input/multi-output plant has inputs u , measured signals y_m and outputs y_c that are primarily controlled. In many cases $y_c \in y_m$. For this controller structure the number of inputs must be

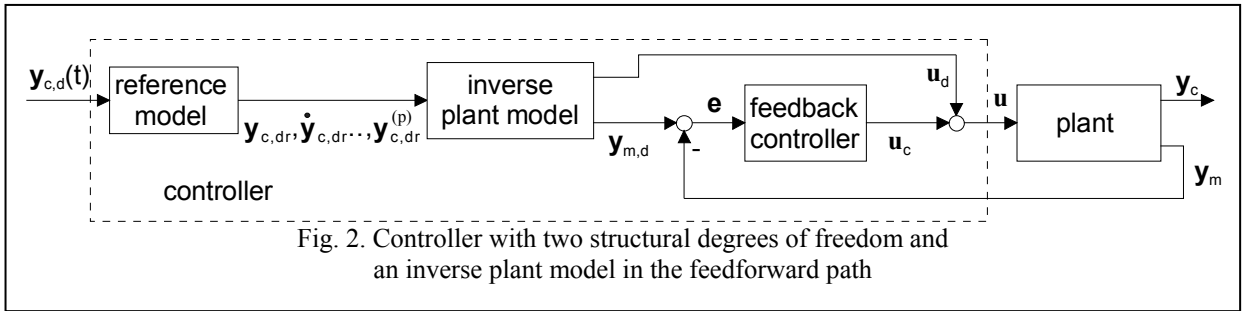


Fig. 2. Controller with two structural degrees of freedom and an inverse plant model in the feedforward path

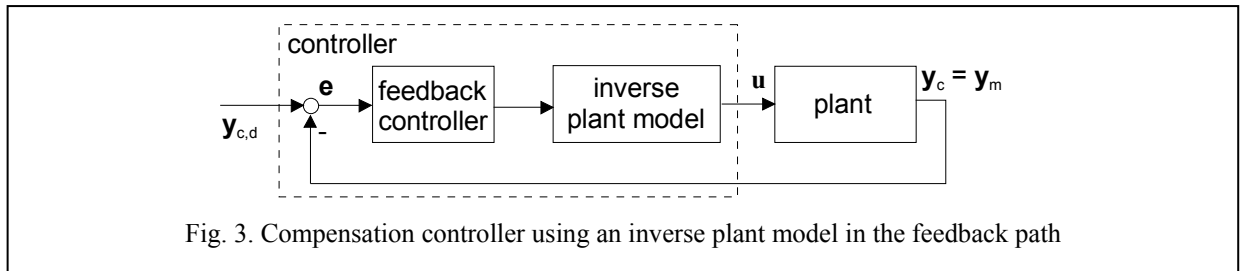


Fig. 3. Compensation controller using an inverse plant model in the feedback path

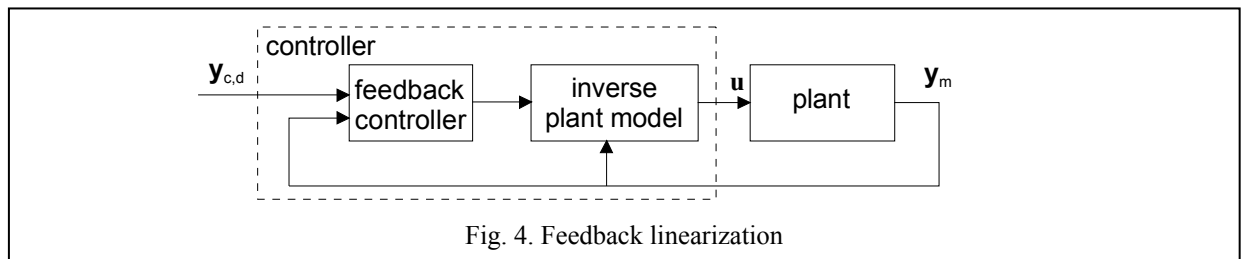


Fig. 4. Feedback linearization

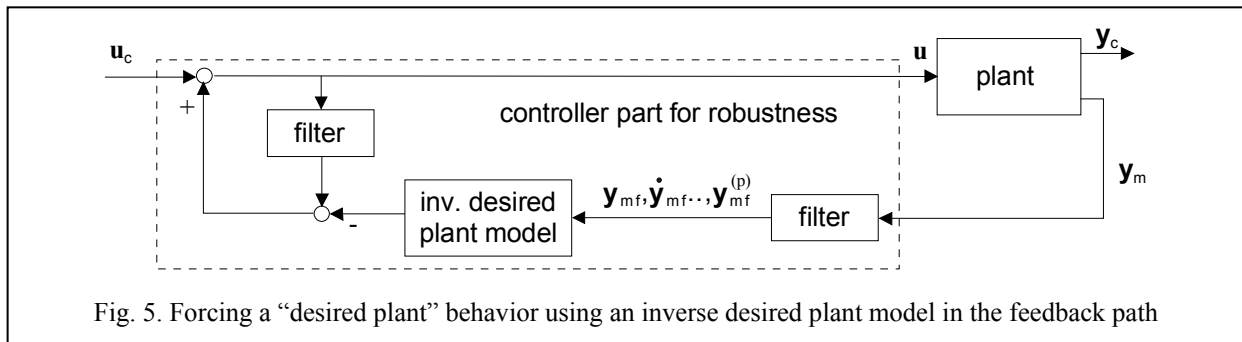


Fig. 5. Forcing a "desired plant" behavior using an inverse desired plant model in the feedback path

identical to the number of controlled variables: $\dim(\mathbf{y}_c) = \dim(\mathbf{u})$. In this case the inverse plant model with (known) inputs \mathbf{y}_c and unknown outputs \mathbf{u} is used in the feedforward path of the controller to compute the desired actuator inputs \mathbf{u}_d to the plant.

A "reference model" defines the desired dynamic behavior of the closed loop system. It is often most convenient to use a filter, since the filter is parameterized by just the cut-off frequency, once the filter order and the filter type is fixed, and because a filter provides the "optimal" reference model with transfer function "1" below the cut-off frequency. There are also other useful choices of the reference model, see for example [2] for in-flight simulation.

The outputs $\mathbf{y}_{c,dr}$ of the reference model are the inputs to the inverse plant model. By solving a DAE system (3) or the symbolically transformed system (4), the

inverse plant model computes the desired measurement signals $\mathbf{y}_{m,d}$ and the desired plant inputs \mathbf{u}_d . A feedback controller is used to stabilize the overall system and to improve robustness. This might be a simple PID like controller.

It can be shown that the feedback controller has no effect, as long as the plant and the inverse plant models are identical, the plant and the inverse plant models are stable and both start at the same initial conditions. In this case the "reference model" determines the input/output behavior, i.e., it is the transfer function of the closed loop system. If these assumptions are not fulfilled, a control error occurs and the controller has to stabilize the system and cope with the imprecise inverse plant model and its initial conditions.

The structure in Fig. 2 has several advantages:

- The two controller parts (inverse plant model with reference model and feedback controller) can be designed independently from each other.
- The controller structure can be applied to unstable plants provided the inverse model is stable, see section 6.1.
- Since the inverse plant model is in the feedforward path, the calculation of \mathbf{u}_d and of $\mathbf{y}_{m,d}$ might be performed *offline* if possible, so that hard real-time requirements for the solution of the inverse plant model are not present.

The disadvantage of this structure is that for some applications the feedback controller may still have to be scheduled as a function of the operating conditions.

Inverse model-based feedforward control will be demonstrated at hand of an example in section 5.

4.2 Compensation Controller

The disadvantage of the inverse feedforward controller can be avoided by moving the inverse model into the feedback path. This is shown in Fig 3. The feedback controller now only “sees” the combined inverse and plant model. The structure is a generalization of the *linear compensation controller* described in Föllinger [10], page 266. For linear plant models the “feedback controller”, see Fig. 3, must have a relative degree that is equal or larger than the relative degree of the plant, in order that the system is proper. For single-input/single-output systems, a useful “feedback controller” is

$$u_c = \frac{1}{r(s)-1} \cdot e \quad (7)$$

Under the assumption that the desired and the actual plant behavior is identical, the inverse and the actual plant model “cancel” each other and the transfer function from $y_{c,d}$ to y_c is identical to $1/r(s)$, i.e., $r(s)$ of the feedback controller defines the “desired” closed loop behavior. Note that it is assumed that y_c is measurable (in this case, $y_c = y_m$). Alternatively, the procedure as described in section 2 may be applied: integrators are added to the inverse model input before designing the feedback controller.

This structure has the disadvantage that it can be applied to stable plants only. Also the inverse plant model needs to be stable. For a linear plant model this is obvious, since otherwise an unstable pole/zero cancellation occurs, resulting in an internally unstable system. For multi-input/multi-output systems it is nearly always possible (also for unstable plants) to construct the inverse of a *stationary* desired plant model. Once the control error \mathbf{e} has reached a sta-

tionary value, the inverse plant model leads to a decoupled control loop. In other words, the different outputs might be controlled independently from each other by simple PID-like single-input/single-output controllers and the stationary inverse plant model is used to decouple the control loops from each other.

4.3 Feedback linearization

A complete theory to use nonlinear plant models as the controller kernel is “feedback linearization” (in aerospace applications also known as Nonlinear Dynamic Inversion, NDI), see for example Isidori [11] and Enns et. al. [7]. The basic structure is given in Fig 4. The principal difference compared with the compensation and feedforward controllers (Fig. 2,3) is that the states in the inverse model are obtained from the actual plant, via measurement and estimation. Contrary to the compensation controller, the methodology can also be applied to unstable plants.

When deriving feedback linearizing control laws manually, the outputs to be controlled are differentiated until an analytical relation with a control input is found [19]. To this end “Lie” algebra is used. The number of required differentiations is the so-called relative degree of the specific output. If the system model is available in Modelica, the derivation of the control laws can be automated using a similar procedure as described in section 2. However, instead of a filter of appropriate relative degree, a set of integrators is added (see section 2):

$$y_i = \frac{1}{s^{p_i}} v_i$$

where v_i is the i^{th} new model input, corresponding with the i^{th} output (with relative degree p_i). The desired dynamic behavior of the closed-loop system is then imposed by application of an additional feedback law, like for example:

$$v_i = k_{0,i}(y_{md,i} - y_{m,i}) - k_{1,i}(y_{m,i}^{(1)}) \dots - k_{(p_i-1),i}(y_{m,i}^{(p_i-1)}) \quad (8)$$

Note that this feedback law requires availability of the $(p_i-1)^{\text{th}}$ derivative of the controlled output. This derivative may be obtained from measurements or, less favorably, from the computed value in the inverse model. In aerospace applications first or no time derivatives are usually required, since relative degrees of controlled variables tend to be low (1 or 2). One reason for this is that control laws are designed in the form of multiple cascaded loops [7]. In case the inverted model exactly represents the true system, the closed loop system becomes:

$$y_{m,i} = \frac{k_{0,i}}{s^p + k_{(p-1),i}s^{p-1} + \dots + k_{1,i}s + k_{0,i}} y_{md,i}$$

Note that the coefficients may be selected to match the reference model in Fig. 2.

In case time derivatives of the desired output $y_{c,d,i}$ are available, the relative degree (i.e. phase lag of the response) of this linear closed loop system may be reduced, provided that these are not too fast as to require too large control inputs.

An important disadvantage of feedback linearization is that the state vector of the plant must be fully available from measurement and/or estimation.

Automatic generation of feedback linearization control laws in Modelica will be illustrated in section 5. This procedure has been applied for an automatic landing system, see [13], and manual control laws for a fighter aircraft, see [21]. The software code for the automatic landing system was automatically generated with Dymola and successfully flight tested on a small passenger jet [3], see the figure below that shows one of the automatic landing tests.



4.4 Robust Controllers

All previous controller structures require that the plant model used as inverse system in the controller match the real plant “sufficiently” accurate. The controller structure in Fig. 5 uses an inverse model to achieve a more robust design. It was developed for linear systems with the goal to enhance robustness against disturbances and model errors, see [14][23] [1]. This structure is called “disturbance observer” in the literature although the name is misleading since it is actually an additional structural degree of freedom for a controller. It can be designed independently from the main control loop. In Fig. 5 the generalization for nonlinear systems is shown: One important part is an inverse model of a desired plant behavior in the feedback path. Additionally, the same filter is present at two places. The standard disturbance observer uses a linear model for the inverse plant model. A nonlinear desired plant model provides more freedoms, since it might be impossible that a physical system can be forced to have the same

linear behavior in its whole operating range. Note, there is the requirement that the number of measurement signals is identical to the number of plant inputs: $\dim(\mathbf{y}_m) = \dim(\mathbf{u})$.

For a single-input/single-output system where all parts are linear, the transfer function from u_c to y_m is given by:

$$y_m = \frac{1}{\frac{1-F(s)}{P(s)} + \frac{F(s)}{P_{des}(s)}} \cdot u_c \quad (9)$$

where $F(s)$ is the filter, $P(s)$ is the plant and $P_{des}(s)$ is the desired plant transfer function in the feedback loop. For low frequencies, $F(s) \approx 1$ and therefore $y_m \approx P_{des}(s) \cdot u_c$. For high frequencies, $F(s) \approx 0$ and then $y_m \approx P(s) \cdot u_c$. The effect of the disturbance observer is therefore, that it enforces the desired plant behavior for low frequencies. In other words, if there are modeling errors or disturbances then the disturbance observer enforces a desired plant behavior below the cut-off frequency of the filter, i.e., the controller designed for the desired plant will usually work considerably better.

The disturbance controller is usually combined with other controller structures. For example, by combining it with the structure from section 4.1, a controller with 3 structural degrees of freedom is obtained:

- An inverse plant model from \mathbf{y}_c to \mathbf{u} in the feedforward path is used for *command following* and for providing the desired measurements $\mathbf{y}_{m,d}$.
- An inverse plant model from \mathbf{y}_m to \mathbf{u} in the feedback path is used to make the closed loop system *robust* against model errors and disturbances.
- The feedback controller in the feedback loop is used to *stabilize* the system.

5 Example application

In this section the feedforward and feedback linearization controller structures as discussed in the previous section will be illustrated on the following example (the plant description is from Föllinger [9], page 279):

A substance A is flowing continuously into a mixing reactor. Due to a catalyst, the substance reacts and splits into several base substances that are continuously removed. The reaction generates energy and therefore the reactor is cooled with a cooling medium. The cooling temperature $T_c(t)$ in [K] is the primary actuation signal. Substance A is described

by its concentration $c(t)$ in [mol/l] and its temperature $T(t)$ in [K] according to the following DAE:

$$\begin{aligned} \gamma &= c \cdot k_0 \cdot e^{-\varepsilon/T} \\ \dot{c} &= -a_{11} \cdot c - a_{12} \cdot \gamma + a_{13} \\ \dot{T} &= -a_{21} \cdot T + a_{22} \cdot \gamma + a_{23} + b \cdot T_c \end{aligned} \quad (10)$$

with

$$\begin{aligned} k_0 &= 1.24 \cdot 10^{14} & a_{11} &= 0.00446 & a_{21} &= 0.0303 \\ \varepsilon &= 10578 & a_{12} &= 0.0141 & a_{22} &= 2.41 \\ b &= 0.0258 & a_{13} &= 0.00378 & a_{23} &= 1.37 \end{aligned}$$

For the given input $T_c(t)$ these are 1 algebraic equation for the reaction speed $\gamma(t)$ and two differential equations for $c(t)$ and $T(t)$. The concentration $c(t)$ is the signal to be primarily *controlled* ($= y_c$) and the temperature $T(t)$ is the signal that is measured ($= y_m$).

5.1 Inverse Model in Feedforward Path

The *inverse* plant model is constructed from (10) by assuming that the variable to be controlled, i.e., the concentration $c(t)$, is a known time function. By inspection or by using the Pantelides algorithm [15] it turns out that the first two equations of (10) have to be differentiated:

$$\begin{aligned} \dot{\gamma} &= \left(\dot{c} + \frac{\varepsilon \cdot c}{T^2} \dot{T} \right) \cdot k_0 \cdot e^{-\varepsilon/T} \\ \ddot{c} &= -a_{11} \cdot \dot{c} - a_{12} \cdot \dot{\gamma} \end{aligned} \quad (11)$$

(10) and (11) are the *inverse model* of (10). A filter with an n^{th} order pole on the negative real axis is used as “reference model”. Since the second derivative of the input appears ($= \ddot{c}$), at least a filter of order 2 is needed, such as:

$$c = \frac{1}{(s/\omega + 1)^2} \cdot c_{des} \quad (12)$$

with c_{des} the desired concentration, $\omega = 2\pi f$ and f the cut-off frequency of the filter. A state space description of the filter is given by:

$$\begin{aligned} \dot{x} &= (c_{des} - x) \cdot \omega \\ \dot{c} &= (x - c) \cdot \omega \end{aligned} \quad (13)$$

The needed second derivative of c is obtained by differentiating the second equation of (13):

$$\ddot{c} = (\dot{x} - \dot{c}) \cdot \omega \quad (14)$$

Equations (10), (11), (13), (14) are the DAE of the *inverse model* of (10) with a *prefilter* of order 2, i.e., these are the connected blocks labeled as “inverse plant model” and as “reference model” in Fig. 2. It

turns out that this DAE has two states. One possibility is to use the filter states $\{x, c\}$ as state vector \mathbf{x}_1 of the overall system. Here, the original plant states $\{c, T\}$ are used as state vector \mathbf{x}_1 . Transforming the equations to the state space form (4) results in the following *sequence of assignment* statements to compute the derivative $\{\dot{c}, \dot{T}\}$ of the state vector and of the output T_c as function of $\{c, T\}$

$$\begin{aligned} \gamma &:= c \cdot k_0 \cdot e^{-\varepsilon/T} \\ \dot{c} &:= -a_{11} \cdot c - a_{12} \cdot \gamma + a_{13} \\ x &:= c + \dot{c} / \omega \\ x' &:= (c_{des} - x) \cdot \omega \\ c'' &:= (x' - \dot{c}) \cdot \omega \\ \gamma' &:= (c'' + a_{11} \cdot \dot{c}) / a_{12} \\ \dot{T} &:= \frac{T^2}{\varepsilon \cdot c} \cdot \left(\frac{\gamma'}{k_0 \cdot e^{-\varepsilon/T}} - \dot{c} \right) \\ T_c &:= (\dot{T} + a_{21} \cdot T - a_{22} \cdot \gamma - a_{23}) / b \end{aligned} \quad (15)$$

For notational clarity, the time derivatives of variables that are treated as purely *algebraic* variables ($=$ “dummy derivative method”) are denoted with an apostrophe, such as γ' . Equations (15) are a set of differential equations in state space form: Given the desired concentrations c_{des} , it is possible by numerical integration to compute the desired cooling temperature T_c ($= \mathbf{u}_d$ in Fig. 2) and the desired substance temperature T ($= \mathbf{y}_{m,d}$ in Fig. 2). The latter is compared with the measured substance temperature forming the control error \mathbf{e} as input to the feedback controller.

Even for this rather simple system, the derivation of the nonlinear feedforward controller is not so easy. Such a manual derivation becomes impractical if the plant model consists of hundreds or of thousands of equations as it is usual in complex Modelica models. It is now demonstrated how to derive this nonlinear feedforward controller in an *automatic* way:

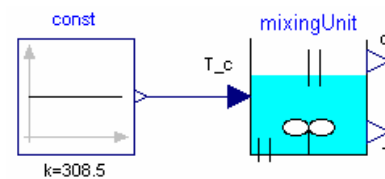


Fig 6. Modelica model of mixing unit with constant cooling temperature T_c

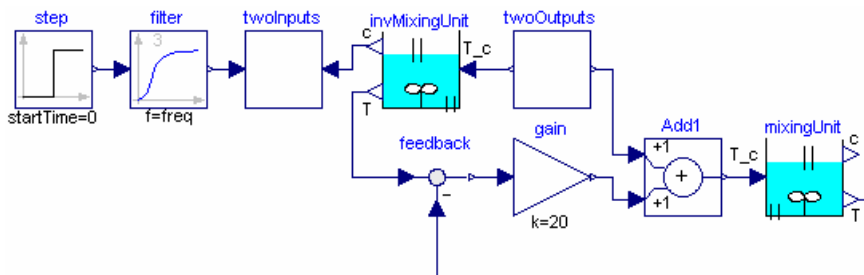


Fig 9. Control system with nonlinear feedforward path for mixing unit

In Fig. 6 a Modelica model of the mixing unit is shown. The constant input is the cooling temperature; the outputs are the concentration c and the temperature T of the substance. This model contains just the equations (10). Simulation results of this model are shown in Fig. 7. As can be seen, the system is unstable at this operating point

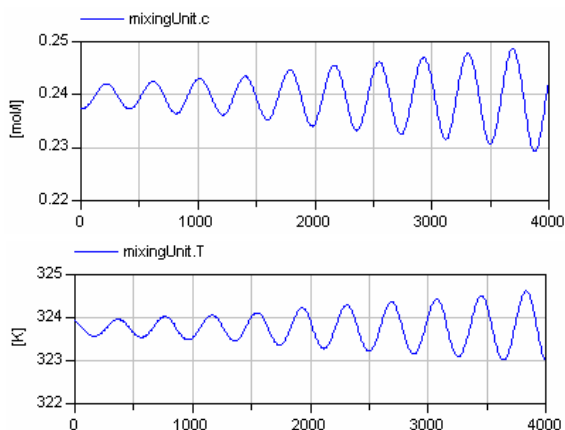


Fig. 7. Simulation results of mixing unit for $c(t_0) = 0.237$ mol/l, $T(t_0) = 323.9$ K, $T_c(t) = 308.5$ K

In Fig. 8 the inverse model of the mixing unit is constructed by connecting the input “ c_{des} ” via a filter to the “ c ” output of the mixing unit, i.e., the concentration c is treated as known input signal.

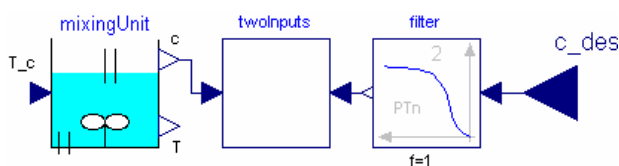


Fig 8. Inverse model of mixing unit

When this system is translated without the filter, Dymola reports that the second derivative of c_{des} is needed. In a second step, the filter is included with order = 2 and Dymola translates without an error. Afterwards, the inverse model is connected with the plant model according to Fig. 2. The result is shown in Fig. 9. In order to not have a jump in the cooling temperature, a filter order of 3 instead of 2 is actually used. The cut-off frequency of the filter is set to 1/300 Hz. It turns out that a simple P controller is

sufficient to stabilize the system. A controller gain of 20 is selected.

Simulation results are shown in Fig. 10 for a jump of $c_{des} = 0.492$ to 0.237 . The straight lines correspond to the nominal case, where the plant and the inverse plant model have the same parameters. The result is a good control behavior. The dashed lines correspond to the case where the parameters of the plant are 50 % higher as the parameters of the inverse plant model to check the robustness of the design (only parameter ε was not changed because the result is very sensitive to it). As can be seen, the result is still satisfactorily. For an actual design, it is useful to perform a Monte Carlo simulation by varying all model parameters and initial conditions of the plant systematically in order to determine how robust the control system is

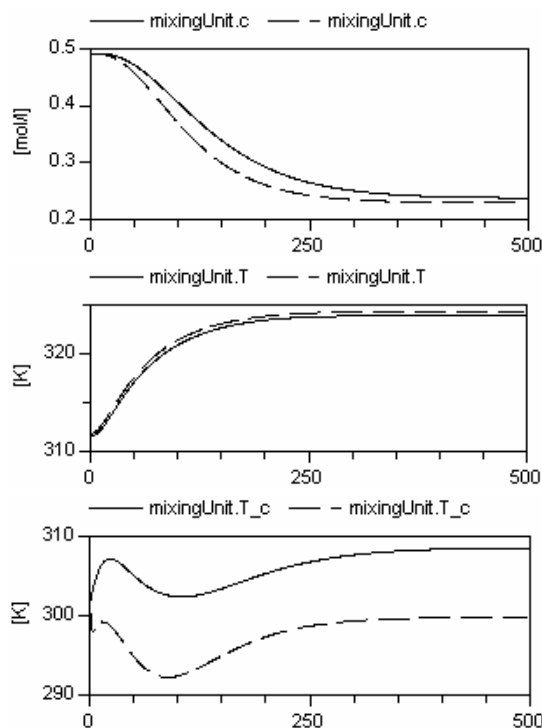


Fig. 10. Simulation results of mixing unit of Fig. 8 Straight line: same model parameters for plant and inverse plant model. Dashed line: model parameters of plant are enlarged by 50 %

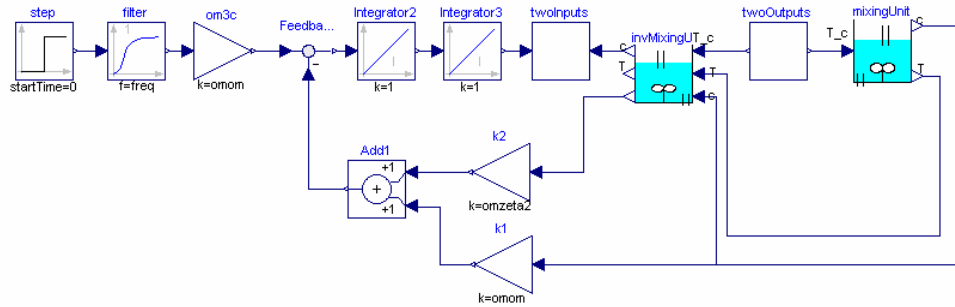


Fig 12. Closed loop system of mixing reactor and feedback linearization controller

5.2 Feedback linearization

The compensation control scheme and the feedback linearization cannot be applied directly to the example plant, since the concentration c is not measurable. One possibility is to use model knowledge in combination with estimation (e.g. a Kalman filter), but this is beyond the scope of this example. For this reason it is assumed that the concentration is measurable. In Modelica, design of feedback linearization and the compensation controllers start in the same way. This time two integrators are added, instead of an input filter, see Fig. 11.

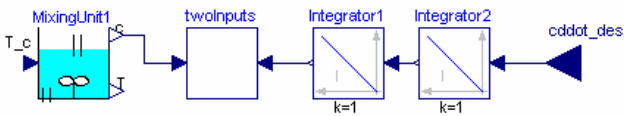


Fig 11. Inverse model of mixing unit for feedback linearization (compare with Fig. 8)

In the case of the compensation controller, the inversion work is done. For feedback linearization, the states in the inverse plant model must be replaced with measured ones. This can be performed by setting the flag

`Advanced.TurnStatesIntoInputs = true`

before translation to transform all states into inputs in the generated code. This code can be incorporated with the export feature of Dymola in another environment, such as Simulink from Mathworks. Currently, it is not possible to import this transformed system in Modelica again. Dynasim plans to support this in the future. For the example, the differentiated equations (11) are added manually to the model, and \ddot{c} and the plant states (c, T) are selected as input variables (in case of complex models this manual derivation is not practical). The design is finished by adding the feedback controller. In case of feedback linearization, a usual choice is:

$$c'' = k_1(c_{des} - c) - k_2\dot{c}$$

whereby c is available from measurement and \dot{c} is computed or obtained from differentiation. By choosing

$$k_1 = (2\pi f)^2, \quad k_2 = 2(2\pi f),$$

with $f = 1/300$ Hz, exactly the same closed loop dynamics is obtained as with the input filter of second order in section 5.1. Starting from the ideal response

$$\frac{1}{r(s)} = \frac{k_1}{s^2 + k_2s + k_1}$$

the feedback controller may also be shaped as:

$$T_c = \frac{1}{r(s) - 1} s^2 = \frac{k_1 s^2}{s^2 + k_2 s}$$

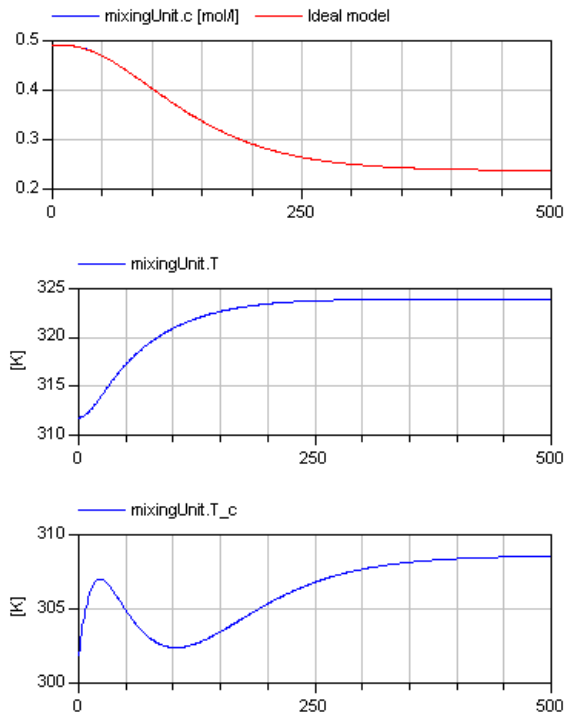


Fig 13. Closed-loop and ideal step response of the mixing reactor

whereby in the numerator s^2 has been added, since the input of the inverse model is effectively the second time derivative of c_{des} . Fig. 13 shows the response of the closed loop system to the same command as in Fig. 10. The command input has been smoothed with a first order filter (as in Fig. 8). The over-all closed-loop system is depicted in Fig. 12.

6 Difficulties with Inverse Models

When constructing inverse models for industrial systems, it is often the case that the generated inverse models do not work as expected. In this section, the major reasons are discussed and it is explained how to circumvent such problems.

6.1 Unstable inverse models

Usually, it is required that the inverse model is a *stable system*. For example, in the structure of Fig. 2, the inverse model is in the feedforward path and if it would not be stable, the overall system would be unstable as well. For linear single-input/single-output systems this situation is well known and can be easily analyzed. For example, take the following linear plant model:

$$y = \frac{s-1}{(s-2) \cdot (s+3)} u \quad (16)$$

The inverse model together with a reference model is

$$u = \frac{(s-2) \cdot (s+3)}{(s-1) \cdot (Ts+1)} y \quad (17)$$

As can be seen, the inverse model is unstable, because the plant has an unstable zero. In other words, for linear systems the plant must be a *minimum phase system* in order that the inverse model is stable. For a general DAE no stability proof exists. Therefore many simulations have to be performed with the inverse DAE to check whether it is stable in the desired operation region. For certain classes of DAEs, it might be possible to prove that the inverse model is stable. An alternative is to linearize the plant model around several stationary operating points and check whether the transmission zeros are stable. Of course, none of these checks can guarantee that the inverse DAE is stable for simulations or stationary points that have not been analyzed.

If the inverse plant is unstable, only *approximate* inverse plant models can be used for the design. For linear single-input/single-output systems this can be achieved by removing unstable zeros before invert-

ing the plant. E.g., in the example above, the approximate inverse plant model would be:

$$u = \frac{(s-2) \cdot (s+3)}{(Ts+1)^2} y \quad (18)$$

For a non-linear plant, one might choose other outputs of the plant as inputs to the inverse model, since this might change the stability behavior of the inverse plant, see for example [20]. Alternatively, the plant might be modified before inversion. These advices are demonstrated by the crane example in Fig. 14.

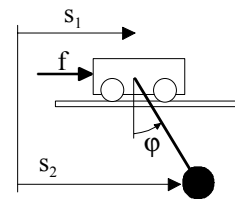


Fig 14. Crane consisting of horizontal moving crab and a load on a rope

The crane consists of a horizontally moving crab and a rope on which the load is attached. For simplicity, the load is modeled as a mass point. The crab is driven by the external force “f”. The horizontal position of the crab “s1” and its derivative “v1” are measured. The goal is to move the load to a specified horizontal position “s2”.

For a non-linear disturbance observer, the inverse model from s1 to f is needed, since s1 is measured. The system is first linearized around the stationary position where the rope hangs vertically down ($\varphi = 0$). The transfer function from f to s1 has 2 conjugate complex zeros on the imaginary axis, signaling an undamped oscillation of the inverse model. This can be improved by including linear damping ($= d \cdot \dot{\varphi}$) in the revolute joint for the inverse plant used in the controller. If the damping constant d is large enough, the two zeros on the imaginary axis are moved to the negative real axis. The disturbance observer is able to force the plant (that does have low damping) moving in such a way as if there would be high damping.

The major goal is to position the load, i.e., to control the horizontal position “s2” of the load. Therefore, the feedforward control should use the inverse model from s2 to f. The transfer function from f to s2 of the linearized model has no zeros and a relative degree of 4. Constructing the inverse model from the non-linear plant model requires, however, a filter of order 2 instead of 4 as suggested by the linearized model. Simulating the inverse model results in a division by zero if $\varphi = 0^\circ$ or $\varphi = 180^\circ$. To summarize, the structure of the inverse model equations is different

at these two points and at $\varphi \neq 0^\circ$ and $\varphi \neq 180^\circ$ (the DAE index is 5 for $\varphi = 0^\circ$ and $\varphi = 180^\circ$ and the DAE index is 3 otherwise). Since the division by zero occurs when computing $\ddot{\varphi}$, the plant model should be changed to compute $\ddot{\varphi}$ in a different way. This can be accomplished by taking the inertia of the load into consideration (previously it was neglected). With a non-zero inertia, the transfer function from f to s_2 of the linearized plant has 2 conjugate complex zeros on the imaginary axis and a relative degree of 2. Again, by introducing damping in the revolute joint, these two zeros are moved to the negative real axis. Note, that the inverse model is very insensitive with respect to the newly introduced load inertia.

To summarize, for the crane example the inverse plant models from s_1 to f and from s_2 to f can be constructed by inverting a modified plant that has a load inertia and additionally damping in the revolute joint. A simpler alternative is also available: Before inversion, the angle φ is fixed to 0° and therefore $s_1 = s_2$, and the plant to be inverted is described by the equations (m_{crab} is the mass of the crab and m_{load} is the mass of the load):

$$(m_{\text{crab}} + m_{\text{load}}) \cdot \ddot{s}_1 = f \quad (19)$$

which can be easily inverted. This example demonstrates that it might be necessary to slightly modify the original plant model in order that the inverse model of the plant can be used in a controller.

6.2 Equations that cannot be inverted

A plant may have equations that cannot be inverted. Examples are time delays, backlash, friction, hysteresis. This can be fixed by *approximating* the problematic elements in such a way that the resulting equation leads to a unique inverse.

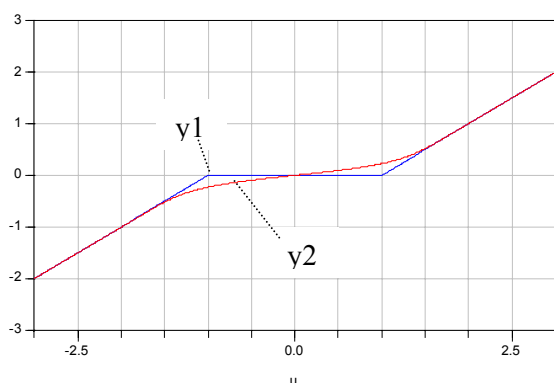


Fig. 15. Backlash (y_1) and approximate backlash (y_2)

A typical example is shown in Fig. 15. The original backlash characteristic $y_1 = f_1(u)$ is not invertible

because for $y_1 = 0$, there are an infinite number of solutions ($u = -1 \dots +1$). In Fig. 15 an approximation $y_2 = f_2(u)$ is shown that is strict monotonic and therefore the inverse function has a unique solution.

It might also occur that tables have to be inverted. Formally, a table in one dimension is defined as a function $y = f(u)$. Inversion of this function means to solve a non-linear equation. This can be often quite easily avoided by providing already the inverse tabulated values $u = g(y)$ in the plant before inverting the plant model. The advantage is that the solution is faster and more robust. This problem was, e.g., encountered in [21], where control surface effectiveness of a military jet tended to have a local maximum as a function of the deflection. This was solved by adapting tables and internal limitations of control commands.

The inverse plant model may have also other singularities at particular operating points or regions that prevent an inversion, e.g., due to divisions by zero, singular linear or singular non-linear systems. The reason is that the corresponding inverse model has no or infinitely many solutions in particular points or in particular regions of the state space. Again, one remedy is to change the plant model before the inversion, e.g., by neglecting dynamic elements or by approximating components with functions that are less problematic to invert.

6.3 Actuator limits

Every control system is inherently limited by constraints in the actuator or other parts of the plant and therefore the question arises how to cope with these restrictions. When inverting a plant model, such constraints have to be removed before the inversion. Otherwise no unique solution of the inverse exists anymore, because there are infinitely many solutions when an actuator is in one of its limits. As a result, usually only the trivial action is possible to add appropriate limiters to the outputs of inverse models. This will only help for short-time violations of the constraints because the control system is effectively switched off when the actuators are in their limits.

The most effective way to cope with actuator constraints in any control system is to adapt the desired control signals, such as $y_{c,d}(t)$ in Fig. 2. In the most general case this means to solve a trajectory optimization problem, i.e., to determine actuator signals $u(t)$ such that the plant outputs $y_c(t)$ have a desired behavior, e.g., reach the desired position in minimum time or with minimum energy, without violating the plant constraints. The result is used as $y_{c,d}(t)$. A typical example can be found in [8]. Note, if the plant is

unstable and the inverse plant model is stable, it might be considerably simpler to solve the trajectory optimization problem with the inverse plant instead with the plant model. Usually, trajectory optimization problems are difficult to solve and therefore highly simplified plant models are used.

Take for example the crane model from section 6.1. The basic requirement is to move the crab from position $s_1=a$ to $s_1=b$ in a short time. The plant model is simplified by fixing the angle to $\varphi = 0^\circ$ resulting in equation (19). Based on (19), the actuator limit $|f| \leq f_{\max}$ can be directly transformed into a limit of the acceleration:

$$|\ddot{s}_1| \leq f_{\max} / (m_{\text{crab}} + m_{\text{load,max}}) \quad (20)$$

Together with limits on the maximum speed, due to the maximum speed of the motor, $|\dot{s}_1| \leq \dot{s}_{1,\max}$, and the requirement to move in minimum time from a to b it is straightforward to construct the analytic solution of the desired movement $s_{1,d}(t)$. This solution is, e.g., available via the block `Modelica.Blocks.Sources.KinematicPTP`. Note, the plant model used for the trajectory optimization problem and for the inverse plant model in the feedforward path according to Fig. 2 are identical here. In such a case, the feedforward controller can be removed and can be replaced by the result of the trajectory optimization: $s_{1,d}$ and $f_{1,d} = (m_{\text{crab}} + m_{\text{load,max}}) \cdot \ddot{s}_{1,d}$. For the trajectory optimization problem an f_{\max} should be used that is, say, 10 % - 20 % smaller as the actual limit in order to provide some margin for the feedback controller.

If the desired control variables $y_{c,d}(t)$ are not known in advance but generated online, e.g., by an operator, online optimization techniques have to be used: The operator request is reduced such that the plant constraints are fulfilled in the next sample time instant. A well known measure in flight control is the so-called daisy-chain. In case a control input saturates, a secondary, redundant control input is brought in that provides the remaining required control power. In [21] for example, lateral deflection of the thrust vector is used to yaw the aircraft in case the rudder saturates.

6.4 Real-time implementation

If inverse plant models are part of the controller, linear and non-linear systems of equations as well as non-linear differential equations might have to be solved in every sampling interval of the controller. The techniques developed for hardware-in-the-loop

simulations can be also applied for such an application. The methods described in [6] are available in Dymola [5] with the Dymola real-time option and can be applied by selecting the appropriate options when translating the inverse model (Simulation / Setup / Realtime / Inline integration method). Only fixed step integrators can be used for a real-time application. Via simulations, the appropriate step size of the integrator has to be determined.

6.5 Robustness

As already mentioned in section 5, the use of inverse model equations gives rise to robustness issues, since any mismatch between the inverted model equations and the actual plant will leave part of the nonlinearities and couplings uncompensated. The usual approach is to provide robustness to model uncertainty via the (linear) feedback controller (Fig. 2,3,4) or the filter (Fig. 5). This can be done by application of a robust control synthesis technique [2], or by robust parameter tuning in a classical structure, e.g. using multi-model techniques and enforcing sufficient stability margins [13].

Tolerances on parameters in the model also appear in the inverse model equations. In [13] it has been shown that these parameters may be very effectively used as additional tuning parameters in multi-objective optimization. The result is a model that is basically inverted at a location in the parameter space that provides the highest level of robustness.

7 Summary

Several control structures have been discussed that are based on non-linear inverse plant models. These structures are attractive since it is possible to cope directly with operating point dependencies. The difficult part to construct an inverse model can be performed automatically even for complex systems: The plant is modeled with Modelica, inputs and outputs are exchanged and a Modelica simulation environment, such as Dymola, generates automatically the appropriate C code for the inverse plant model, including real-time integration algorithms. The generated code can be easily embedded into Simulink from Mathworks using the corresponding Dymola export option. Via Mathworks Realtime-Workshop, the code can be finally downloaded to different target processors.

The presented controller structures can be used in all types of areas such as control of robots, vehicles, aircrafts, satellites, ships, motors, air conditioning

systems. The most important requirement is that an appropriate plant model is available. Then, the inverse modeling approach is in principle fully automatic, although the practical application is usually more difficult. The essential issues have been discussed in section 6 and also possible remedies

References

- [1] Ackermann J., Blue P., Bünte T., Güvenc L., Kaesbauer D., Kordt M., Muhler M., and Odenthal D. (2002): **Robust Control: The Parameter Space Approach**. Springer-Verlag.
- [2] Adams, R.J., Banda, S. **Robust Flight Control Design Using Dynamic Inversion and Structured Singular Value Synthesis**. IEEE Transactions on Control Systems Technology, 1(2):80-92, June 1993.
- [3] Bauschat, M., Mönnich, W., Willemsen, D., and Looye, G. **Flight testing Robust Autoland Control Laws**. In Proceedings of the AIAA Guidance, Navigation and Control Conference, Montreal CA, 2001.
- [4] Duda H., Bouwer G., Bauschat J.M., Hahn K.-U. (1997): **A Model Following Control Approach**. In "Robust Flight Control: A Design Challenge" by J.-F. Magni, S. Bennani and J. Terlouw (editors), Springer Verlag, pp. 116 – 124.
- [5] Dynasim (1994): **Dymola – Users Manual** (<http://www.dynasim.com>)
- [6] Elmqvist H., Mattsson S.E., Olsson H. (2002): **New Methods for Hardware-in-the-Loop Simulation of Stiff Models**. 2nd International Modelica Conference, March 18-19, DLR Oberpfaffenhofen, Germany, pp. 59-64. Download: <http://www.Modelica.org/-Conference2002/papers.shtml>.
- [7] Enns, D., Bugajski, D., Hendrick, R., and Stein, G.. **Dynamic Inversion: An Evolving Methodology for Flight Control Design**. In AGARD Conference Proceedings 560: Active Control Technology: Applications and Lessons Learned, pages 7-1 – 7-12, Turin, Italy, May 1994.
- [8] Franke R., Rode M., and Krüger K. (2003): **On-line Optimization of Drum Boiler Startup**. 3rd Int. Modelica Conference, Linköping, Nov. 3-4, pp. 287 – 296. Download: <http://www.Modelica.org/-Conference2003/papers.shtml>.
- [9] Föllinger O. (1998): **Nichtlineare Regelungen I**, Oldenbourg Verlag, 8. Auflage.
- [10] Föllinger O. (1994): **Regelungstechnik**. Hüthig Verlag, 8. Auflage.
- [11] Isidori A. (1995): **Nonlinear Control Systems**. 3rd Edition, Springer Verlag.
- [12] Kreisselmeier G. (1999): **Struktur mit zwei Freiheitsgraden**. Automatisierungstechnik at 6, pages 266-269.
- [13] Looye G. (2001): **Design of Robust Autopilot Control Laws with Nonlinear Dynamic Inversion**. Automatisierungstechnik at 49-12, p. 523-531.
- [14] Ohnishi K. (1987): **A new servo method in mechatronics**. Trans. Japanese Society of Electrical Engineering, vol 107-D, pp. 83-86.
- [15] Pantelides C.C. (1988): **The consistent initialization of differential-algebraic systems**. SIAM Journal of Scientific and Statistical Computing, pp. 213-231.
- [16] Mattsson S.E., Söderlind G. (1993): **Index reduction in differential-algebraic equations using dummy derivatives**. SIAM Journal of Scientific and Statistical Computing, pp. 677-692.
- [17] Mugica F., Cellier F.E. (1994): **Automated synthesis of a fuzzy controller for cargo ship steering by means of qualitative simulation**. Proceedings of the European Simulation MultiConference (ESM'94), Barcelona, Spain, pp. 523-528,
- [18] Otter M., Cellier F.E. (1996): **Software for Modeling and Simulating Control Systems**. The Control Handbook, by W.S. Levine (editor), CRC Press, pp. 415 – 428.
- [19] Slotine, J.E, Li, W. **Applied Nonlinear Control**. Prentice Hall, Englewood Cliffs, N.J., 1991.
- [20] Snell, A. **Decoupling of Nonminimum Phase Plants and Application to Flight Control**, AIAA-2002-4760 AIAA Guidance, Navigation, and Control Conference and Exhibit, Monterey, California, 2002.
- [21] Steinhäuser R., Looye G., Brieger O. (2004): **Design and Evaluation of a Dynamic Inversion Control Law for X-31A**. Proc. 6th ONERA-DLR Aerospace Symposium, Berlin, June 22-23, pp. 25-33.
- [22] Thümmel M., Otter M., Bals J. (2001): **Control of Robots with Elastic Joints based on Automatic Generation of Inverse Dynamics Models**. IEEE/RSJ Conference on Intelligent Robots and Systems, Oct. 29- Nov. 3rd, Hawaii, U.S.A.
- [23] Umeno T., Hori Y. (1991): **Robust speed control of dc servomotors using modern two degrees-of-freedom controller design**. IEEE Trans. Ind. Electron, 38-5, pp. 363-368.