Proceedings of the EuroGNC 2013, 2nd CEAS Specialist Conference on
Guidance, Navigation & Control, Delft University of Technology, Delft,
The Netherlands, April 10-12, 2013

Automatic Control Generation for Aircraft Taxi Systems Through Nonlinear Dynamic Inversion of Object-Oriented Model

Fabrizio Re

Abstract Within the framework of automatic ground propulsion systems for aircraft, a method is presented to generate Feedback Linearization based controllers in an automated way. A nonlinear on-ground aircraft model realized in the object-oriented language Modelica is inverted automatically and used as Feedback Linearizing core of a ground trajectory tracking system. The controller is completed by an outer linear loop. Issues in the model inversion process are discussed. In particular, robustness against parameter uncertainties must be assessed carefully. With this method, the study and development of automatic ground propulsion systems can be quickened because control laws can be obtained for different system architectures and different aircraft starting from the respective dynamic models, allowing easier and quicker assessment of these technologies and comparisons between different aircraft platforms.

1 Introduction

A number of airframers, suppliers and research institutions are investigating and designing innovative zero-emission taxiing systems that allow to move the aircraft on ground without generation of noise or pollution and with greater energetic efficiency than conventional taxi with jet engines. Proposed solutions include using electric motors to drive the landing gear wheels [10]. Along with these propulsion devices, some have proposed automatic ground control systems. While automatic flying controls are common to the point that they are taken for granted in present-day flight, ground movements are mostly accomplished by the pilot himself manually and the potential of automation in this area is often underestimated. Indeed, autonomous taxi would be beneficial in a number of aspects. Taxi may be more robust to exter-

Fabrizio Re
DLR German Aerospace Center, Institute of System Dynamics and Control, Oberpfaffenhofen, Germany, e-mail: fabrizio.re@dlr.de

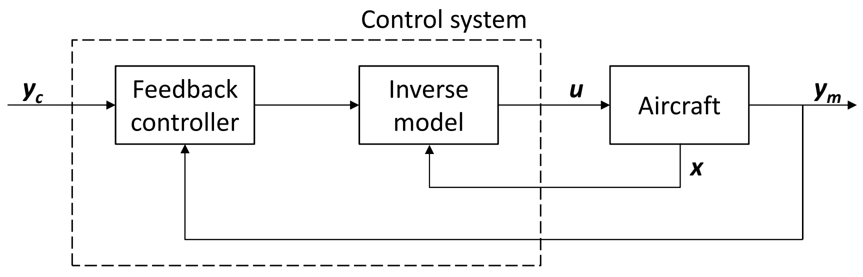


Fig. 1 Proposed control architecture for autonomous taxiing aircraft

nal disturbances (e.g. wind gusts). Also, the pilot can concentrate more on pre-flight and post-flight activities. In a future perspective, autonomous taxi might be part of a fully automated ground control at airport level, which would use the infrastructure capacity more efficiently and enhance the safety of ground movements by reducing the risk of human errors. Also, autonomous taxi can help optimize the use of electric taxiing systems from an energetic point of view by commanding convenient dynamics with regard to the taxi path, the airport traffic and other constraints.

In this paper, a method is presented to automatically generate inversion-based controllers starting from an object-oriented aircraft model. The issue to be addressed is that a ground vehicle (as a taxiing aircraft can be considered) is a nonlinear dynamic system. Simple linear control methods will fail to stabilize the system in complex situations like turns with large steering angles (e.g. U-turns) or with low grip. This can be solved in a conceptually simple way by using Feedback Linearization (FBL), also known as Nonlinear Dynamic Inversion (NDI). The core of the controller contains an inverse model of the plant that cancels out the plant nonlinearities, whereas the desired closed-loop behavior is imposed by an outer linear control loop (fig. 1). Here, the features of object-oriented modeling languages, such as Modelica [3], can be exploited to quickly obtain an inverse model from a given aircraft model. This is a very interesting feature in the research and engineering phase of such ground propulsion systems because it allows to easily adapt the plant and the controller to different aircraft architectures, making it possible to carry out comparisons and to assess the viability and performance of those systems in various conditions on different aircraft starting from their respective models. In section 2, the theoretical framework of the method will be presented first. Then in section 3, the steps needed to automatically generate the inverse model for control are illustrated. In section 4, a simplified aircraft model will be realized in Modelica and exported to Simulink to build a controlled plant model for the purpose of demonstrating the proposed method. Concluding remarks will be given in section 5.

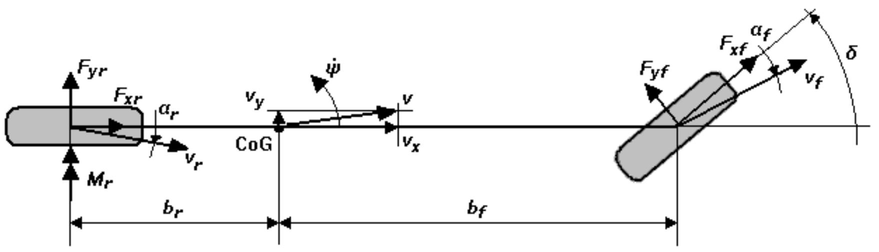


Fig. 2 One-track model ("bycicle model") of the on-ground aircraft

2 Theoretical background

The principle of the proposed method shall be illustrated in this chapter. The feedback linearization of a simplified, nonlinear on-ground aircraft model will be performed for this purpose.

2.1 On-ground aircraft model

A nonlinear one-track model ("bicycle model") will be used for the on-ground aircraft (Fig. 2). The vehicle features one steerable wheel at the front and one drivable/brakable wheel at the rear. Vertical dynamics is neglected, aerodynamics only include the aerodynamic resistance as function of the longitudinal speed (effects of aerodynamic surfaces are neglected). Moreover, the dynamics of the front tire will be neglected; its longitudinal slip and longitudinal force (along the wheel plane) will be considered zero at all times. With these assumptions, the equations of motion are:

$$\begin{aligned} m(\dot{v}_x - \dot{\psi} v_y) &= -F_{yf}\sin\delta + F_{xr} - F_{res} \\ m(\dot{v}_y + \dot{\psi} v_x) &= F_{yf}\cos\delta + F_{yr} \\ J_z\ddot{\psi} &= b_f F_{yf}\cos\delta - b_r F_{yr} \\ J_r\dot{\omega}_r &= M_r - F_{xr}R_r \end{aligned} \tag{1}$$

where:

- m is the aircraft mass
- J_z is the aircraft moment of inertia around the vertical axis
- v_x and v_y are the longitudinal resp. lateral velocity in body coordinates
- $\dot{\psi}$ is the yaw rate
- ω_r is the angular velocity of the rear wheel
- R_r and J_r are the rear wheel radius and moment of inertia
- M_r is the driving/braking moment on the rear wheel
- δ is the nose wheel steering angle

- $F_{xf},\ F_{yf},\ F_{xr},\ F_{yr}$ are the tire forces in longitudinal resp. lateral direction (subscript x and y) in the wheel local coordinates for the front resp. rear tire (subscript f and r)
- $F_{res} = F_{res}(v_x)$ summarizes the resistances, including rolling and aerodynamic resistance
- b_f and b_r are the distances of the front resp. rear wheel to the aircraft center of gravity.

A tire model is needed to link the tire forces to the aircraft kinematics. For this example, a simple linear tire model is chosen:

$$\begin{aligned} F_{xf} &= 0 \text{ (see assumptions)} \qquad & F_{yf} &= k_{yf}\alpha_f \\ F_{xr} &= -k_{xr}\sigma_r & F_{yr} &= k_{yr}\alpha_r \end{aligned} \tag{2}$$

with $k_{xf},\ k_{yf},\ k_{xr},\ k_{yr}$ constant stiffnesses of the tires, $\sigma_f,\ \sigma_r$ are the longitudinal tire slips, and $\alpha_f,\ \alpha_r$ are the tire slip angles. These quantities are defined as follows:

$$\begin{aligned} &\alpha_f = \delta - \arctan\left(\frac{v_y + b_f\dot{\psi}}{v_x}\right) \\ \sigma_r = \frac{v_x - \omega_r}{v_x} \qquad &\alpha_f = \arctan\left(\frac{-v_y + b_r\dot{\psi}}{v_x}\right) \end{aligned} \tag{3}$$

The system is completely described by equations 1, 2, and 3. A road vehicle with one steering axle and a given stable tire model only has two degrees of freedom. [5] Therefore, x_v and $\dot{\psi}$ are taken as state variables along with the internal state ω_r.

2.2 Feedback Linearizing controller

This well-known method, also called Nonlinear Dynamic Inversion (NDI), consists in finding a diffeomorphic coordinate transformation of a nonlinear system such that the inputs of the transformed system linearly map to the original outputs; this way, the system can be inverted easily and linear methods can be applied for stabilization and tracking. The approach for MIMO systems is described for example in [4] and is briefly summarized here for completeness.

A nonlinear system with n and n outputs is given,

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}) + \sum_{i=1}^{n} g_j(\mathbf{x})\, u_i \\ y_j &= h_j(\mathbf{x}) \qquad j = 1, 2, \ldots, n \end{aligned} \tag{4}$$

Using the Lie derivative notation $L_f h(x) = \frac{\partial h}{\partial x} f(x)$, each y_j is derived a number of times r_j with respect to time, which gives:

$$y_j^{(r_j)} = L_f^{(r_j)}(h_j) + \sum_{i=1}^{n} L_{g_i} L_f^{(r_j-1)}(h_j)\, u_i$$

until, for the r_j-th derivative, $L_{g_i}\left[L_f^{r_j-1}(h_j)\right] \neq 0$ holds for some i while $L_{g_i}\left[L_f^{k}(h_j)\right] = 0\ \forall k \leq r_j - 2$ and for every i. The quantity r_j is called relative degree of the output y_j. After this procedure, a non-singular *decoupling matrix J* is defined:

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} L_{g_1}[L_f^{r_1-1}(h_1)] & \dots & L_{g_n}[L_f^{r_1-1}(h_1)] \\ \dots & \dots & \dots \\ L_{g_1}[L_f^{r_n-1}(h_n)] & \dots & L_{g_n}[L_f^{r_n-1}(h_n)] \end{bmatrix}$$

and, after introducing the vector

$$\mathbf{l}(\mathbf{x}) = \begin{bmatrix} L_f^{(r_1)}(h_1) \\ \dots \\ L_f^{(r_n)}(h_n) \end{bmatrix}$$

the following equation holds:

$$\begin{bmatrix} y_1^{(r_1)} \\ \dots \\ y_n^{(r_n)} \end{bmatrix} = \mathbf{l}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\mathbf{u} = \mathbf{v} \tag{5}$$

where $\mathbf{v}$ is the $n \times 1$ vector of transformed inputs that are equal to the r_j-th derivative of each output y_j. If the system is inverted, i.e. if the inputs should be calculated that make it possible for the output to follow an assigned $\mathbf{y}_{\text{des}}$, this is accomplished by calculating

$$\mathbf{v}_{\text{des}} = \begin{bmatrix} y_{1,\text{des}}^{(r_1)} \\ \dots \\ y_{n,\text{des}}^{(r_n)} \end{bmatrix}$$

which implies that each $y_{j,\text{des}}$ must be derivable r_j times, and by inverting eq. 5:

$$\mathbf{u} = \mathbf{J}^{-1}(\mathbf{x})\left(\mathbf{v}_{\text{des}} - \mathbf{l}(\mathbf{x})\right) \tag{6}$$

The system has thus been linearized and decoupled, i.e. each output is controlled separately by one transformed input. Finally, it is possible to apply standard linear techniques to control the behavior of the system. For example, the closed-loop poles of each $v_j - y_j$ response may be changed by an appropriate choice of feedback parameters:

$$K_j(s) = a_{r_j-1}s^{r_j-1} + a_{r_j-2}s^{r_j-2} + \dots + a_0$$

which results in the following closed-loop transfer function:

$$G_j = \frac{1}{s^{r_j} + a_{r_j-1}s^{r_j-1} + a_{r_j-2}s^{r_j-2} + \cdots + a_0}$$

For the aircraft model showed in the previous chapter, the problem is to find the necessary inputs $[\delta,\ M_r]$ in order to follow a given trajectory $[v_{x,\text{des}},\ \dot{\psi}_{\text{des}}]$. It can be noted that the system in the form given by eq. 1 is not affine in control as required in eq. 4 since δ appears as argument of trigonometric functions. This can be overcome by considering $\hat{\delta} = \delta$ a system state and using its derivative $\dot{\delta}$ as input. This is also advantageous in reality because the command input of the steering system is the hydraulic servo-valve control current that is a function of $\dot{\delta}$ [6]. With this operation and after substituting eq. 2 and 3, the system becomes:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}$$
$$\mathbf{y} = \mathbf{h}\mathbf{x}$$

$$\mathbf{x} = \left[v_x,\ \dot{\psi},\ \omega_r,\ \hat{\delta}\right]^T, \qquad \mathbf{u} = \left[\dot{\delta},\ M_r\right]^T$$

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \dot{\psi}v_y - \frac{1}{m}\left[k_{yf}\sin\hat{\delta}\left(\hat{\delta} - \arctan\left(\frac{v_y+b_v\dot{\psi}}{v_x}\right)\right) - k_{xr}\frac{v_x-\omega_r}{v_x} - F_{res}(v_x)\right] \\ \frac{1}{J_z}\left[b_f k_{yf}\cos\hat{\delta}\left(\hat{\delta} - \arctan\left(\frac{v_y+b_v\dot{\psi}}{v_x}\right)\right) - b_r k_{yr}\arctan\left(\frac{-v_y+b_h\dot{\psi}}{v_x}\right)\right] \\ -\frac{F_{xr}R_r}{J_r} \\ 0 \end{bmatrix} \quad (7)$$

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \frac{1}{J_r} \\ 1 & 0 \end{bmatrix} \qquad \mathbf{h}(\mathbf{x}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

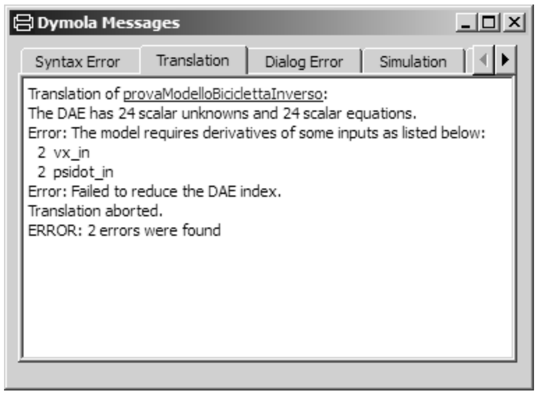
Model inversion involves differentiating the outputs as many times as necessary for the inputs $\dot{\delta}$ and M_r to appear. This calculation may be lengthy and tedious if performed manually on a complex model. The analytic result for this example will be omitted, however, focusing on the first two rows of $\mathbf{f}(\mathbf{x})$ corresponding to $\dot{v}_x$ and $\ddot{\psi}$, it can be intuitively noticed that deriving them once yields some terms multiplied by $\dot{\delta}$ and some others multiplied by $\dot{\omega}_r$ which in turn is linearly dependent on M_r. This means that both outputs v_x and $\dot{\psi}$ need to be differentiated twice in order for the inputs to appear. Therefore the relative degree is 2 for both outputs.

3 Automated Object-Oriented Model Inversion

Object-oriented modeling is a technique that is becoming more and more widespread in various engineering fields. Basic components of a system and relevant phenomena are modeled as classes containing sets of algebraic and differential equations interrelating the states, the inputs, and the outputs of each component or phenomenon. The classes are connected together to build the whole system by means of appropriate interfaces exchanging signal flows and states as needed. This method offers a number of advantages over other, currently still very common modeling techniques, e.g. signal flow modeling environments such as Simulink. Notably, the modular structure allows a more realistic modeling of real components and sub-assemblies; it allows a better understanding of the model parts; it makes it easier to modify or expand a system without major modifications of the model; and it is very suitable for building discipline-specific libraries.

The particular feature that is exploited in this work is that the causality of the system model is not determined in the modeling phase; inputs and outputs are specified only in the code generation phase and the executable simulation code will be generated accordingly in an automated way by a compiler. This improves the model flexibility dramatically, since a direct model and an inverse model can be generated easily from the very same system model with only minor modifications. This work used the open-source modeling language Modelica [3] and the simulation environment Dymola [1], which is based on Modelica and also features, along with a command layer, a graphic layer that allows to work with classes represented as blocks and to connect them in a signal-flow-like fashion. In model compilation, this software is capable of transforming the set of Differential Algebraic Equations (DAE) contained in the model into a state-space representation of Ordinary Differential Equations (ODE) that will be simulated afterwards. This transformation generally involves symbolic algorithms for index reduction (e.g. the Pantelides algorithm [2]) and selection of states (e.g. Mattson and Söderlind's "Dummy Derivative Method" [2], [9]).

With regard to the basic example in the previous chapter, a model class can be created containing the system equations (1), (2), and (3) and specifying the needed parameters (mass, inertia, tire stiffnesses, geometry). This model can now be compiled and simulated as a direct model by assigning the inputs δ and M_r. To compile an inverse model, v_x and $\dot{\psi}$ should be declared as inputs. This can be done in the command layer, or alternatively in the graphic layer by using the connectors *RealInput* and *RealOutput*, which already contain the necessary commands. Upon DAE reduction, the compiler differentiates the equations containing states; in this simple example, two states are directly assigned as inputs and no derivatives are available, hence the DAE cannot be transformed, and an error message similar to Fig. 3 is output. This informs the user that additional derivatives of the inputs are needed - in this example, two for v_x and two for $\dot{\psi}$. If the inputs are available as analytic functions, additional inputs for the needed input derivatives can be added and connected to the analytic derivatives available. Alternatively the needed derivatives of the highest order (in this example, the second derivatives) can be declared as inputs, and an appropriate number of integrators should be added to the system model. If the analytic derivatives are not available, a solution consists in using filters of the appropriate order to obtain the needed derivatives through a combination of the filter states. [11] In the real system, this equates to treating the commands so as to ensure that they are sufficiently smooth (i.e. derivable the necessary number of times). The filter dy-



```
Translation of provaModelloBiciclettaInverso:
The DAE has 24 scalar unknowns and 24 scalar equations.
Error: The model requires derivatives of some inputs as listed below:
  2 vx_in
  2 psidot_in
Error: Failed to reduce the DAE index.
Translation aborted.
ERROR: 2 errors were found
```

Fig. 3 Dymola message informing about missing input derivatives

namics should be chosen fast enough to be negligible compared to the dynamics of the whole system.

For the inversion-based FBL controller, the states appearing in the inverse model must be fed back from the system. The Dymola command

```
Advanced.TurnStatesIntoInputs = true
```

instructs the compiler to consider the states as system inputs. Afterwards, the model is compiled and simulation code is generated that may be simulated in Dymola or exported to other environments for convenience, e.g. to Simulink where a DymolaBlock exists for this purpose.

3.1 Challenges and Difficulties

Two important issues need to be addressed when applying the proposed method. Firstly, nonlinearities are only canceled out if the inverse model exactly matches the real plant. This is often difficult to achieve in reality mainly because of parameter uncertainties or approximate system modeling. While the controlled linearized system should perform adequately in theory, unsuppressed nonlinearities may result in poor performance and even instability of the real controlled plant. For this reason, along with a precise enough system model in the inversion-based part, the controller must be made robust against uncertainties. This is usually addressed in the linear, outer-loop part of the controller by applying well-known robust synthesis techniques.

Secondly, in the general case, if the system order n is greater than the sum of relative degrees r_j, then there exists some unobservable internal dynamics whose stability needs to be verified. A sufficient condition for local asymptotic stability of closed-loop MIMO systems is that the zero dynamics are asymptotically stable [4]. For the specific one-track model example considered, $n = \sum r_j = 4$, hence there is no internal dynamics.

If the model should become more complex, e.g. if more realistic, nonlinear tyre models were to be considered, then effects like actuator or tire saturation should be also taken into account in the control system. Generally speaking, for an efficient inversion it is expedient to proceed as follows with the system modeling [8]:

- The actuator dynamics are neglected within the inverse model. As a consequence, the actuator commands are calculated that correspond to the necessary actuator action once its dynamics has reached the steady state in the real plant;
- The actuator saturation limits are neglected; also, regarding the tire models, the force must be a strictly monotonic function of the slip. If this were not the case, there would be no, or no unique solution for certain system states and controller inputs, causing the inverse model simulation to crash.

Controlling these aspects must be a task of the outer control loop. So for example, actuator saturations and rate limitations can be detected and corrected for through methods like Pseudo-Control Hedging [7].

4 Application Example

An example of application of this technique for generation of controls is given. The on-ground aircraft system described in eq. (1) is modeled in the Modelica language in the Dymola environment and then compiled as a ”direct” system (with δ, M_r as inputs and v_x, $\dot{\psi}$ as outputs) and exported into Simulink. Additional outputs are the full system states. The quantities $\dot{v}_x$ and $\ddot{\psi}$ are chosen as more suitable controller inputs than the second derivatives that were shown to be needed in section 2.2, mainly because controlling accelerations is more intuitive and less sensitive to disturbance than controlling jerks. This means the inputs given to the controller need to be derivable in time once. For this reason, the aircraft model is inverted by declaring $\dot{v}_x$ and $\ddot{\psi}$ as inputs, and these are connected to two analytic first-order, 50 Hz lowpass filters that will provide the compiler with the state equations needed to differentiate the inputs. This model is also compiled and exported to Simulink. Additionally, saturation blocks and first-order, 1.59 Hz lowpass filters are added before the system inputs δ and M_r to model actuator saturations (maximum driving/braking torque, maximum steering rate) and actuator delays; finally, $\dot{\delta}$ is integrated into δ before going into the system input. The outer loop consists in one PD and one PID linear controller for closed-loop control of v_x resp. $\dot{\psi}$. The Simulink model is shown in fig. 4. The parameters m, J_z, b_f, b_r, k_{xr}, k_{yf}, k_{yr} used in the Modelica models correspond to a narrow-body airliner.

To test the controller, the input profiles have been chosen as follows: a deceleration ramp from the initial $v_x = 10$ m/s to 5 m/s in 1 s, followed by a sharp left turn bringing $\dot{\psi}$ from zero to 0.2 rad/s in 0.25 s, which results in a cornering radius of 25 m. The corresponding steering radius for the aircraft type considered is in a range around 30 degrees, hence this is a maneuver where linear approximations (e.g.

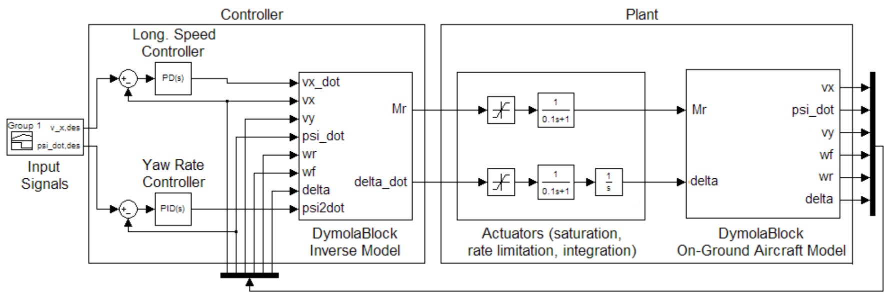


Fig. 4 Simulink Model of the controlled system

small-angle approximations) cannot hold. The closed-loop system response to the input signals is shown in fig. 5.

As stated previously, one important issue is to assess the controller robustness against parameter uncertainties. While robustness has not yet been investigated thoroughly for this work and will be addressed in a next stage, three simulations have been carried out with different parameters for a preliminary robustness test. A reliable estimation of the aircraft mass is usually available online in the flight computer, and moment of inertia as well as the position of the center of gravity can be also estimated. However, the tire parameters related to the tire-ground grip are critical for

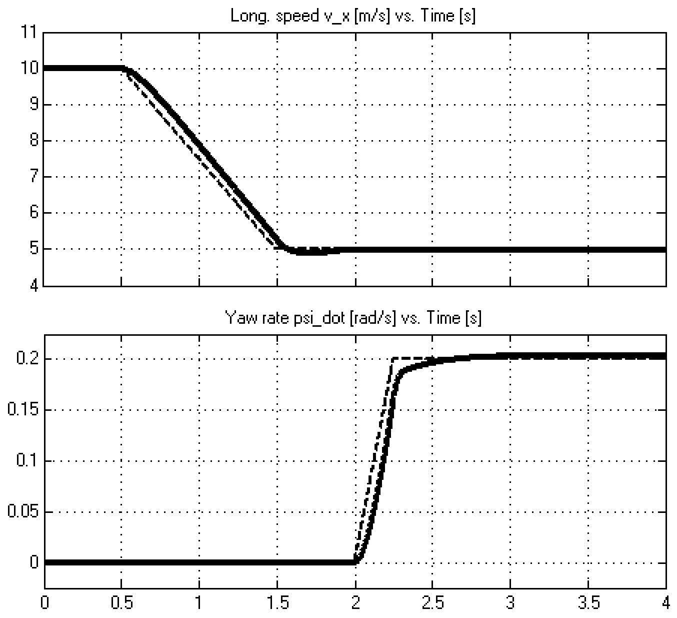


Fig. 5 Ramp response of the controlled system without parameter uncertainty (equal parameters in inverse and direct model). Dashed lines: Commanded outputs. Solid lines: actual outputs

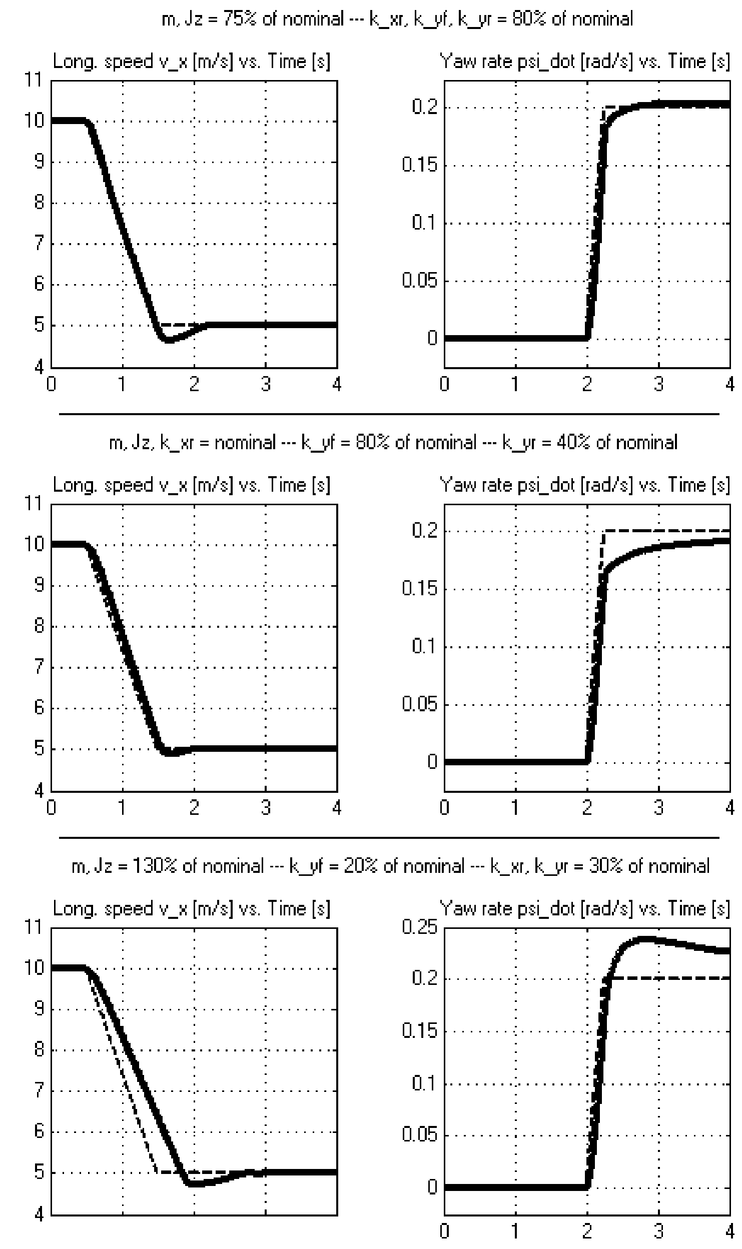


Fig. 6 Ramp response of three different simulations with parameter uncertainty. The parameters changed in each simulation are described above the respective diagrams. Dashed lines: Commanded outputs. Solid lines: actual outputs

correct tracking and stability itself and their estimation is not straightforward. Therefore, the controller must be robust against their variation within a certain range. For these tests, the parameters in the controller (inverse model) are left at their nominal value, whereas they are changed in the plant model (direct model). The outputs of these different simulations and the parameters used in each simulation are shown in fig. 6. The response remains stable even in low-grip situations and with incorrect mass or moment of inertia, although with a degraded performance e.g. in reaching the steady-state yaw rate, as well as in slowing down the aircraft in the last simulation due to the physically limited tire grip. The results suggest that the control is robust over a wide parameter range and may be further optimized, for example by tuning the PID parameters with multi-objective optimization algorithms or by using appropriate robust synthesis methods for linear systems.

5 Conclusion and Outlook

A method has been presented to obtain automatic inverse models of an on-ground aircraft for use in a ground controller based on Feedback Linearization. As an example, a simple one-track aircraft model has been written in an object-oriented modeling language, then automatically inverted and used as inner loop of the control system; the outer loop consisted in linear PID controllers. The functionality has been shown through simulation examples, as well as robustness in selected conditions. With this method, the study and development of automatic ground propulsion systems can be quickened because control laws can be obtained in a partially automatic way for different architectures and different aircraft starting from the respective dynamic models. More complex models can be used for inversion, including more realistic aerodynamics, actuator models and tire models, as long as the contained functions are invertible (e.g. the actuators in the inverse model do not saturate and their effect is monotonically dependent on their input). Multiple actuators can also be included in the inverse model as long as a control allocation strategy is provided. Robustness of the proposed controller structure is key in all controllers based on feedback linearization because of parameter uncertainties. In future stages of this work, the controller parameters may be optimized for robustness over a wide range of working conditions and parameter intervals; also, robust synthesis techniques may be applied. More generally, once a control architecture has been defined for an on-ground propulsion system, optimization algorithms may be investigated that would automate the entire control synthesis process, thus allowing a quick adaptation of the system to different aircraft platforms.

References

1. Dassaults Systemes' Dymola website (retrieved August 2010). URL

http://www.3ds.com/products/catia/portfolio/dymola
2. Cellier, F.E., Kofman, E.: Continuous System Simulation. Springer (2006). ISBN 978-0-387-26102-7
3. Fritzson, P., Bunus, P.: Modelica–A General Object-Oriented Language for Continuous and Discrete-Event System Modeling and Simulation. In: Proceedings of the 35th Annual Simulation Symposium (San Diego, California, April 14-18, 2002). DOI 10.1109/SIMSYM.2002.1000174
4. Henson, M.A., Seborg, D.E.: Nonlinear Process Control. Prentice Hall (1997)
5. Isermann, R.: Fahrdynamik-Regelung. ATZ/MTZ-Fachbuch. Vieweg+Teubner (2006)
6. Jeanneau, M.: The AIRBUS On-Ground Transport Aircraft Benchmark. In: D. Bates, M. Hagstrm (eds.) Nonlinear Analysis and Synthesis Techniques for Aircraft Control. Springer (2007)
7. Johnson, E.N., Calise, A.J.: Pseudo-Control Hedging: A New Method for Adaptive Control. In: Advances in Navigation Guidance and Control Technology Workshop (Redstone Arsenal, Alabama, USA, November 1-2, 2000)
8. Looye, G.: Rapid Prototyping Using Inversion-based Control and Object-oriented Modelling. In: D. Bates, M. Hagstrm (eds.) Nonlinear Analysis and Synthesis Techniques for Aircraft Control. Springer (2007)
9. Mattsson, S.E., Söderlind, G.: Index Reduction in Differential-Algebraic Equations Using Dummy Derivatives. SIAM Journal on Scientific Computing **14(3)**, 677–692 (1993)
10. Re, F.: Viability and State of the Art of Environmentally Friendly Aircraft Taxiing Systems. In: ESARS 2012 International Conference on Electrical Systems for Aircraft, Railway and Ship Propulsion (Bologna, Italy, October 16-18, 2012). DOI 10.1109/ESARS.2012.6387462
11. Thümmel, M., Looye, G., Kurze, M., Otter, M., Bals, J.: Nonlinear Inverse Models for Control. In: Proceedings of the 4th International Modelica Conference, pp. 267–279 (Hamburg, March 7-8, 2005)