Symbolics for control : Maple used in solving the exact linearization problem

Symbolics for Control: MAPLE used in Solving the Exact Linearization Problem

Bram de Jager

Faculty of Mechanical Engineering, WH 2.137
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven
The Netherlands

Abstract

In this paper the problem of exact linearization of dynamic systems by application of a suitable controller is discussed. Algorithms that are instrumental in solving this problem are implemented in the so-called NonLinCon package, based on the symbolic computation program MAPLE. Four examples are presented to illustrate the use of NonLinCon. They show that this package is a useful tool for the automated design of changes of coordinates and state-feedbacks that solve the exact linearization problem. The limited capability to solve partial differential equations is a bottleneck. Recommendations to improve NonLinCon and to make MAPLE more suitable for implementations like this one are given.

1 Introduction

The usefulness of industrial products depends on how well they comply with their specifications. These specifications are becoming more and more tight, due to an increased expectation level of the consumer and due to the accumulated knowledge and experience of the producer, resulting in high quality goods at a relatively low price. These market pull and technology push trends in the economy are important, as the following quote illustrates, "Consumption—to repeat the obvious—is the sole end and object of all economic activity" (John Maynard Keynes).

Control systems play an important role in some of these goods. Examples where control systems are used to attain goals that were not within easy reach without them are abundant, to name a few: engine control, (semi) active suspension control, hydraulic steering, and automatic brake systems for motor vehicles, laser beam focusing and radial servos for compact disk players, fuzzy controllers for home appliances, advanced tracking controllers for manipulators, stabilizing controllers for inherently unstable airplanes (fly-by-wire), etc.

To expand the current knowledge base, there is an ongoing research effort:

1. to increase the performance of control systems for enhanced quality by using advanced mathematical theories and more computing power,

Computer Algebra in Industry 2 Edited by A.M. Cohen, L. van Gastel and S.M. Verduyn Lunel

2. to shrink the control system design cycle for a reduced time to market, using advanced mathematical algorithms and computer-aided-design tools.

For linear systems an abundant number of theories, algorithms, and design tools is available. However, to produce high quality goods at low costs, these programs are not always sufficient, because they cannot cope well with non-linear systems. These tools are not adequate because they can only manipulate numbers, i.e., data entry, number crunching, and data visualization are their main forte. Symbolic computation programs are an alternative tool because they can manipulate both numbers and symbols, e.g., manipulate mathematical expressions.

To use these programs effectively they must be:

- without a steep learning curve;
- able to solve most problems, preferably without too much user programming;
- adaptable and expandable, to fit the user's needs closely;
- cost effective.

When a tool does not fulfil these requirements, it is not of much value for the practising engineer.

The question is: Do current symbolic computations programs satisfy these requirements and do they offer a viable alternative for numeric computation and paper and pencil work? We try to answer this question with a case study, where several mathematical algorithms, that are useful in the analysis and design of non-linear control systems, are implemented in the package NONLCON using the symbolic computation program MAPLE as computing substrate.

The use of symbolic computation programs for control purposes is investigated by several researchers. Some linear control problems are handled by REDUCE, see [1, 2, 3]. REDUCE has also been used for the design of some non-linear observers via observer normal forms [4].

The use of multidimensional Laplace transform for the analysis of a specific class of non-linear systems is advocated by Barker et al. [5] who implemented this method in MACSYMA. Zeitz et al., [6, 7] use the program MACNON, based on MACSYMA, to analyse observability and reachability, and to design observers and controllers for non-linear systems. In the paper [6] they discuss ten observer design methods ranging from a working point observer to an extended Kalman filter. They also implemented some controller design methods in MACNON. Their package is mainly used for teaching. Results for larger scale problems are published in [8, 9]. For linear systems some work using MACSYMA is reported in [10]. Blankenship [11, 12] used MACSYMA to design output tracking controllers for non-linear systems

via feedback linearization and (left) invertability with his implementation CONDENS. He used MATHEMATICA also, and provides a control toolbox for this platform. Some MATHEMATICA notebooks, e.g., COSY_PAK, are developed to mimic MATLAB tool boxes, primarily with the aim to get a more powerful visualization and a possible integration of symbolic capabilities, although those capabilities are not fully exploited now. A symbolic toolbox for MATLAB, using the oem kernel of MAPLE, is commercially available.

To study stability properties of a restricted class of single-input single-output non-linear systems MAPLE was used in [13]. The use of MAPLE for several problems in non-linear control is presented in [14]. Problems reported in this paper, e.g., with solving partial differential equations, are partly resolved in [15]. They describe a MAPLE package, here called NONLCON (a successor of the ZERODYN package presented and used in [14, 16]), that can compute, e.g., the zero dynamics and provide solutions to exact linearization problems.

In the present paper we illustrate the use of this package by using it for some textbook and practice-oriented problems. Contrary to [16, 17], where attention is focused on the computation of the zero dynamics and input–output exact linearization, here the focus is on state space exact linearization.

The main contribution of this work is a further assessment of the suitability of a symbolic computation program for the analysis and design of control systems. Other goals are to supply feedback to the developers of these programs, and to solve a semi-industrial control problem. Compared with [15] this paper differs mainly by presenting results obtained for some examples in the book by Isidori [18] and for the semi-industrial problem in the control of a spacecraft. It also assesses explicitly the areas in the implementation of the package and the underlying symbolic computation program that are problematic and limit its usefulness.

The paper is structured as follows. First, Section 2 presents, and makes some remarks on, the specific problem that has to be solved in the case study. Then, Section 3 discusses the mathematical details of the problem. Section 4 follows with a solution of the problem, an investigation of the mathematical tools needed, and algorithms used. The implementation of these algorithms in MAPLE is given a short treatment in Section 5, where the NONLCON package is described. Section 6 gives four examples, illustrating the use of the NONLCON package. Finally, Section 7 presents the conclusions and discusses the objectives for future research.

2 The Exact Linearization Problem

From several problems in non-linear control, where symbolic computations are likely to be of some profit, we discuss the state space exact linearization problem.

The exact linearization problem is of longstanding interest in control theory. In essence, it is the problem of modifying a non-linear dynamical system such that, after the modification, it behaves like a linear one, so powerful design methods for linear systems can again be employed.

To be able to make a system behave like a linear one, some modifications of the system are needed. Because, in our setup, the system itself is not allowed to be changed, the only possible modifications are the judicious manipulation of control signals, i.e., signals that act on the system and can be influenced from the outside, and a change of coordinates.

Examples of control signals are valve settings, that can influence flow rates or heat inputs, and electrical currents or voltages, that can influence the torque exerted by motors. Most of these control inputs are generated by control devices.

Generation of control signals is done by a control law, where information of the system is used to generate the control input. For an overview of the problem set-up, see Figure 1.

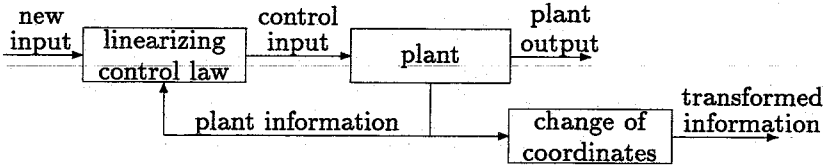

Figure 1: Standard control problem set-up.

This figure should be interpreted as follows. The non-linear system, called the *plant*, has some *input* signals, that can be manipulated, and gives the values of some *output* signals, e.g., measured temperature, position, or speed. The *control law* processes information of the plant (or feeds it back) to generate the control input. The controlled system can be influenced by a new input and be observed by the plant output. The goal is to get a linear (dynamical) relation between the new input and the transformed information of the plant, i.e., expressed in appropriate coordinates.

In a more complete control system design, the exact linearization is often only a subordinate goal, to make it possible to use other design methods for attaining additional goals. It is also possible to consider a more limited goal, where only the behaviour between the new input and the output of the plant is required to be linear. We will not consider this case, although the conditions to reach this goal are more easily satisfied.

3 The Mathematical Formulation

In the presentation of the mathematics, we closely follow the work of Isidori [18]. We start with a non-linear model of a square plant, and assume that the plant can be described adequately by a set of non-linear differential equations, affine in the input u, and without direct feed-through from input to output

$$\dot{x} = f(x) + g(x)u, \qquad y = h(x), \tag{1}$$

where the *state* vector $x \in \mathbb{R}^n$, containing all necessary information of the plant, the *input* vector $u \in \mathbb{R}^m$, and the *output* vector $y \in \mathbb{R}^m$, so the number of inputs is equal to the number of outputs, i.e., the plant is square. This assumption is for convenience only and makes a simplified presentation possible. The vector field f is a smooth one, g has m columns g_i of smooth vector fields, and h is a column of m scalar-valued smooth functions h_i. The assumptions that the model is affine in the input u, i.e., that u enters linearly in (1), and that there is no direct feed-through from u to y, i.e., $h(x)$ is not an explicit function of u, can often be circumvented by an appropriate redefinition or augmentation of the state x, the input u, or the output y, and is therefore not very restrictive.

Not all systems can be described with differential equations of the type of (1), e.g., sometimes it is convenient to include derivatives $u^{(k)}$ of the input u in the model equations. Then, a more general model is needed. For an illustration how this can be done see [19], where a general controller canonical form is introduced.

The modifications we allow for the exact linearization are *state-feedback*, i.e., a feedback based on the explicit knowledge of the value of the state vector $x(t)$ of the plant, and a (local) *change of coordinates* in the state space. The type of control law used is restricted to *static* state-feedback. In a static state-feedback the value of the input vector u at time t depends on the state $x(t)$ and a new *reference* input vector $v(t)$. We assume this dependency to be of the form

$$u = \alpha(x) + \beta(x)v \tag{2}$$

because it does not change the structure of (1). Here α_i and β_{ij} are smooth functions. In a *dynamic* state-feedback the value of $u(t)$ depends on $x(t)$, a new input $v(t)$, and an *auxiliary state* vector $\zeta(t) \in \mathbb{R}^k$. This dependence is of the form

$$\begin{aligned} u &= \alpha(x,\zeta) + \beta(x,\zeta)v \\ \dot{\zeta} &= \gamma(x,\zeta) + \delta(x,\zeta)v \end{aligned}$$

because, again, it does not change the structure of (1). The components α_i, β_{ij}, γ_i and δ_{ij} are smooth functions. See Figure 2 for an overview of the set-up for a static state-feedback.

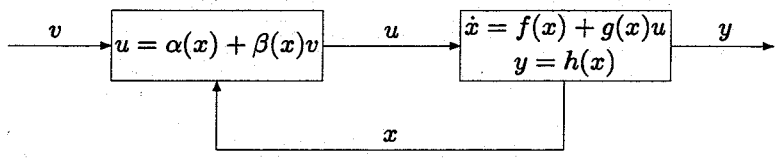

Figure 2: Control structure for static state-feedback.

Although for linear systems a linear change of coordinates $z = Tx$ in the state space $\mathbb{R}^n$, with T a non-singular matrix, is usually adequate, for non-linear systems it is more appropriate to allow for a non-linear change of coordinates

$$z = \Phi(x). \tag{3}$$

It is required that the Jacobian $\partial\Phi/\partial x$ of the transformation vector Φ is, at least locally, invertible for Φ to qualify as a change of coordinates.

We can now state our problem, which we will call the *state space exact linearization problem*: under which conditions is it possible to transform the system (1) to a linear and controllable one by state-feedback (2) and a change of coordinates (3)? The linearity property should be established between the new input v and the transformed state z. This problem has been solved, see, e.g., [18], and our goal is to test the conditions and to derive explicit equations for the feedback and the change of coordinates for specific plants.

The next section gives the conditions, and contains also some remarks on how the state-feedback and the change of coordinates can be computed.

4 The Solution of the Problem

Here the conditions for solving the exact linearization problem are given. To state the solution more easily, we define the so-called *distributions*:

$$\begin{aligned} G_0 &= \text{span}\{g_1, \ldots, g_m\} \\ &\vdots \\ G_i &= \text{span}\{\text{ad}_f^k g_j : 0 \le k \le i, 1 \le j \le m\} \end{aligned}$$

for $i = 0, \ldots, n-1$. Before defining the adjoint ad, we first give definitions for the Lie derivative, $L_f\lambda = \frac{\partial\lambda}{\partial x} f(x)$, where λ is a scalar-valued function of x, and the Lie product

$$[f, g_i] = \frac{\partial g_i}{\partial x} f - \frac{\partial f}{\partial x} g_i,$$

where f and g_i are vector fields. In terms of the Lie product ad is defined recursively as $\mathrm{ad}_f^k g_i = [f, \mathrm{ad}_f^{k-1} g_i]$ with $\mathrm{ad}_f^0 g_i = g_i$.

We now state the conditions for a solution of the state space exact linearization problem [18, Theorem 5.2.4].

Theorem 1 *Suppose a system*

$$\dot{x} = f(x) + g(x)u, \quad x \in \mathbb{R}^n, \quad u \in \mathbb{R}^m$$

with $\operatorname{rank} g(x^\circ) = m$ *is given. There exists a solution for the state space exact linearization problem if and only if:*

(1) G_i *has constant dimension near* x° *for each* $0 \leq i \leq n-1$,

(2) G_{n-1} *has dimension* n,

(3) G_i *is involutive for each* $0 \leq i \leq n-2$.

Here, *involutive* means that the distribution is closed under the Lie product, i.e., the dimension of the distribution G_i does not change when a vector field, generated by the Lie product of each combination of two of the vector fields in G_i, is added to the distribution.

Remark 1 For $m = 1$ the situation is easier, because condition (1) implies (1), and then condition (1) for $i = n-2$ implies (1) for $0 \leq i \leq n-3$.

Remark 2 For linear systems the conditions are equivalent with conditions for the controllability of the system.

When the conditions for the exact linearization problem are fulfilled, the state-feedback and change of coordinates that realize the linearization are still to be determined. It can be shown that, when the three conditions mentioned above are fulfilled, there exist solutions $\lambda_i(x), i = 1, \ldots, m$, for the following partial differential equations

$$L_{g_j} L_f^k \lambda_i(x) = 0, \quad \text{for } 0 \leq k \leq r_i - 2 \text{ and } 0 \leq j \leq m. \tag{4}$$

Also $\sum_{i=1}^m r_i = n$, where the set of integers $\{r_1, \ldots, r_m\}$ is called the *relative degree* vector. The m functions λ_i can be computed, based on a constructive proof of Theorem 1.

Using the functions λ_i, the change of coordinates and state-feedback that solve the state space exact linearization problem are given by

$$z = \Phi(x) \quad = \quad \begin{bmatrix} \lambda_1(x) \\ \cdots \\ L_f^{r_1 - 1} \lambda_1(x) \\ \vdots \\ \lambda_m(x) \\ \cdots \\ L_f^{r_m - 1} \lambda_m(x) \end{bmatrix}$$

$$\begin{aligned} \alpha(x) &= -A^{-1}(x)b(x) \\ \beta(x) &= A^{-1}(x) \end{aligned}$$

with the $m \times m$ nonsingular matrix A and the $m \times 1$ column b given by

$$A(x) = \begin{bmatrix} L_{g_1} L_f^{r_1-1}\lambda_1(x) & \cdots & L_{g_m} L_f^{r_1-1}\lambda_1(x) \\ \vdots & \ddots & \vdots \\ L_{g_1} L_f^{r_m-1}\lambda_m(x) & \cdots & L_{g_m} L_f^{r_m-1}\lambda_m(x) \end{bmatrix} \tag{5}$$

$$b(x) = \begin{bmatrix} L_f^{r_1}\lambda_1(x) \\ \vdots \\ L_f^{r_m}\lambda_m(x) \end{bmatrix}. \tag{6}$$

Using the state-feedback (2), the transformed state $z = \Phi(x)$ depends linearly on the new input v.

An analysis of these formulae shows that a symbolic computation program should be able to compute the Lie derivative and Lie product, perform matrix vector multiplication, compute a matrix inverse, etc. These computations are relatively easy. The main problem is in the computation of the functions λ_i, where partial differential equations have to be integrated. Although the solutions λ_i are known to exist, the actual computation can be complicated. Our approach is to use a constructive proof of the Frobenius' theorem. We will discuss this in the next section.

Remark 3 A completely different algorithm to solve the exact linearization problem, avoiding the integration step, also exists; see [20]. This algorithm is not implemented in NONLICON.

5 Computer Algebra Solution

The algorithm, described in the previous section, to compute the solution for the exact linearization problem, is included in NONLICON. Besides the state space exact linearization problem, other problems, like the computation of the zero dynamics and of the input–output linearizing feedback, both for systems with and without a well-defined relative degree, are included in this package. The structure of the implementation is sketched in Figure 3.

To solve the state space exact linearization problem we use the functions `outputfunc`, `reldeg`, `transform`, and `statelin`. In `outputfunc` the functions λ_i are computed. They should give the system a full order relative degree, i.e., the sum of the vector relative degree components is equal to the systems order: $\sum_{i=i}^{m} r_i = n$. In `reldeg` the vector relative degree is computed, using f, g, and $h = \lambda$. Matrix A and column b are results of

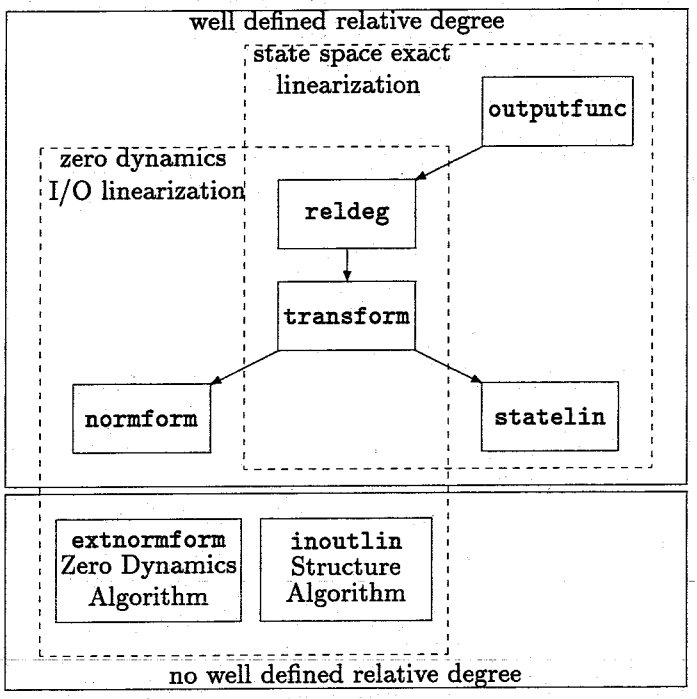

Figure 3: Structure of NONLICON.

the relative degree computation. Then `transform` computes the required transformation $z = \Phi(x)$, also used to bring the system in a standard (normal) form, and its inverse Φ^{-1}. From the computed A and b `statelin` derives the state-feedback $u = \alpha(x) + \beta(x)v$. The control law expressed in the new coordinates z can also be computed using the inverse transformation Φ^{-1}. For more information about NONLICON, especially for other parts of the package, see [15, 21, 22].

In the following we focus on a specific problem that appeared during the implementation of the algorithm presented in Section 4. This problem presents the major bottleneck for the symbolic solution of the exact linearization problem.

In the previous section we remarked that the main problem was to compute the functions λ_i, and that the integration of partial differential equations was a final step in this computation. In MAPLE almost no facilities are available to solve partial differential equations. Therefore the following route was chosen.

The partial differential equations we would like to solve are from the "completely integrable" type, so, based on Frobenius' theorem, we know

that a solution for the partial differential equations exists. To compute the solutions Frobenius' theorem itself is of no help. This problem was solved by computing the solutions with an algorithm based on a constructive proof of Frobenius' theorem. The procedure is as follows.

We use the property that the solution of our type of partial differential equation can be constructed by composing the solutions of related sets of ordinary differential equations; see the constructive proof of Frobenius' theorem in [18]. Because MAPLE provides some facilities to solve sets of ordinary differential equations, with the `dsolve` command, the problem seems solved. However, the `dsolve` command is not very powerful, and is often unable to present a solution, although this solution is known to exist. Therefore the `dsolve` procedure was extended in an ad hoc way, so a larger class of problems could be handled. In `extdsolve` a recursive procedure to solve sets of differential equations was implemented, starting from the "shortest" (assumed to be the simplest) equation, substituting the solution in the remaining equations, and so on. No effort was spend in trying to detect a (block) triangular dependency structure in the set of differential equations, that would be a more rigorous option. See also [23].

Nevertheless, the computation of the functions λ_i is often unsuccessful, especially for more complicated systems, so NONLINCON cannot finish the computations. This part of NONLINCON should therefore be considered as experimental. It seems unlikely that another symbolic computation program will improve this situation. A possible solution could be to use the alternative algorithm of [20].

6 Examples

To illustrate the use of NONLINCON and assess its usefulness, we consider four examples, one for a system with $m = 1$, i.e., a system with one input and output signal. The second example is for a system with multiple input and output signals and this is more complicated. These two examples are contrived ones. The first is taken from [18, Example 4.2.2], the second from [18, Example 5.2.1]. The last two examples are for the attitude control of a spacecraft.

Example 1. The model of the system is

$$\dot{x} = \begin{bmatrix} x_3(1+x_2) \\ x_1 \\ x_2(1+x_1) \end{bmatrix} + \begin{bmatrix} 0 \\ 1+x_2 \\ -x_3 \end{bmatrix} u.$$

To check whether this system can be transformed into a linear and controllable one via state-feedback and a change of coordinates, we have to compute the functions $\mathrm{ad}_f g(x)$ and $\mathrm{ad}_f^2 g(x)$ and test the conditions of Theorem 1. See also Remark 1.

Appropriate calculations show that

$$\begin{aligned} \mathrm{ad}_f g(x) &= \begin{bmatrix} 0 \\ x_1 \\ -(1+x_1)(1+2x_2) \end{bmatrix} \\ \mathrm{ad}_f^2 g(x) &= \begin{bmatrix} (1+x_1)(1+x_2)(1+2x_2) - x_1 x_3 \\ x_3(1+x_2) \\ -x_3(1+x_2)(1+2x_2) - 3x_1(1+x_1) \end{bmatrix}. \end{aligned}$$

At $x = 0$, the matrix

$$\begin{bmatrix} g(x) & \mathrm{ad}_f g(x) & \mathrm{ad}_f^2 g(x) \end{bmatrix}_{x=0} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

has rank 3 and therefore condition (2) (and also (1)) of Theorem 1 is satisfied. It is easy to check that the product $[g, \mathrm{ad}_f g](x)$ is of the form

$$[g, \mathrm{ad}_f g](x) = \begin{bmatrix} 0 \\ * \\ * \end{bmatrix}$$

and therefore also condition (3) is satisfied, because the matrix

$$\begin{bmatrix} g(x) & \mathrm{ad}_f g(x) & [g, \mathrm{ad}_f g](x) \end{bmatrix}$$

has rank 2 for all x.

A function $\lambda(x)$ that solves the equation

$$\frac{\partial \lambda}{\partial x} \begin{bmatrix} g(x) & \mathrm{ad}_f g(x) \end{bmatrix} = 0$$

is given by

$$\lambda(x) = x_1.$$

We check this result and observe that

$$\begin{aligned} L_g \lambda(x) &= 0 \\ L_g L_f \lambda(x) &= 0 \\ L_g L_f^2 \lambda(x) &= (1+x_1)(1+x_2)(1+2x_2) - x_1 x_3 \\ L_g L_f^2 \lambda(0) &= 1. \end{aligned}$$

Locally around $x = 0$, the system will be transformed into a linear one by the state-feedback

$$u = \frac{-x_3^2(1+x_2) - x_2 x_3 (1+x_2)^2 - x_1(1+x_1)(1+3x_2) + v}{(1+x_1)(1+x_2)(1+2x_2) - x_1 x_3}$$

and the change of coordinates

$$z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \Phi(x) = \begin{bmatrix} x_1 \\ x_3(1+x_2) \\ x_3x_1 + (1+x_1)(1+x_2)x_2 \end{bmatrix}.$$

The edited, log of a MAPLE session shows that these results can be reproduced by NON≡CON. From the functions supplied by NON≡CON we only need `outputfunc` for the computation of the λ_i, `statelin` for the linearizing state-feedback u, and `normform` for the change of coordinates $\Phi(x)$. The conditions given for the singularity of the matrix A in (5) define states for which the linearizing state-feedback is not well defined.

```
* finding functions for exact linearization, (outputfunc)*
the conditions for which the matrix A turns out to be singular are:

                                                    2              2
        1 + 3 x[2] + x[1] + 3 x[1] x[2] + 2 x[2]  + 2 x[1] x[2]
{{x[3] = ----------------------------------------------------------},
                                  x[1]

{x[1] = 0, x[2] = -1}, {x[1] = 0, x[2] = -1/2}}

the function(s) lambda that fulfil the demands are: [ x[1] ]

* exact linearization of the state input equations, (statelin)*
the exact linearizing feedback: (u)

                2       2                                  2
(- v[1] + x[3]  + x[3]  x[2] + x[3] x[2] + 2 x[3] x[2]

         3                    2                    2
   + x[2]  x[3] + x[1] + x[1]  + 3 x[1] x[2] + 3 x[1]  x[2])

  /                                                 2           2
 / (x[3] x[1] -1 -3x[2] -x[1] -3x[1] x[2] -2x[2]  -2x[1] x[2] )
/

* transformation to the normal form, (transform)*
                                                          2           2
x[1], x[3] + x[3]x[2], x[3]x[1] + x[2] + x[1]x[2] + x[2]  + x[1]x[2]
```

It is easy to check that these results coincide with the previous ones.

Example 2. The model of the system is

$$\dot{x} = \begin{bmatrix} x_2 + x_2^2 \\ x_3 - x_1x_4 + x_4x_5 \\ x_2x_4 + x_1x_5 - x_5^2 \\ x_5 \\ x_2^2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \cos(x_1 - x_5) \\ 0 \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} u_2.$$

This system satisfies the conditions of Theorem 1. We have to construct two functions λ_i. This is worked out in [18] and the result is

$$\lambda_1 = x_1 - x_5, \qquad \lambda_2 = x_4.$$

The following log of a MAPLE session, with some edits, shows that this result (and more) can also be obtained by NONLINCON.

```
* finding functions for exact linearization, (outputfunc)*
the conditions for which the matrix A turns out to be singular are:

{{x[1] = 1/2 Pi + x[5]}}

the function(s) lambda that fulfil the demands are:

                [-x[1] + x[5], x[4]]

* exact linearization of the state input equations, (statelin)*
the exact linearizing feedback: (u)

                                  2
              v[1] - v[2] + x[2]                2
              ------------------, v[2] - x[2]
                cos(x[1] - x[5])

* transformation to the normal form, (transform)*

x[1] - x[5], x[2], x[3] - x[1] x[4] + x[4] x[5], x[4], x[5]
```

The computed state-feedback and change of coordinates complete the results given in [18].

Example 3. In this example the linearizing state-feedback is computed for a less trivial model. The model to be considered is for a three degrees-of-freedom satellite [24, 25, 26], orbiting in a circular orbit in a square law gravitational field with constant orbital angular speed ω_0 of the center-of-mass; see Figure 4.

The following two coordinate frames are used (for $i = 1, 2, 3$):

1. x_i a body-fixed frame through the center-of-mass and aligned with the principal axes of inertia,

2. ζ_i an orbital frame fixed to the center-of-mass and aligned with the orbit, with ζ_1 tangent to the orbit in the direction of movement, ζ_3 pointing away from the center of the circular orbit and ζ_2 completing the orthogonal dextral frame.

The angles θ_i $(i = 1, 2, 3)$ are pitch, yaw, and roll, respectively, and are also called the Tait–Bryan angles. Rotation from the ζ to the x frame is by

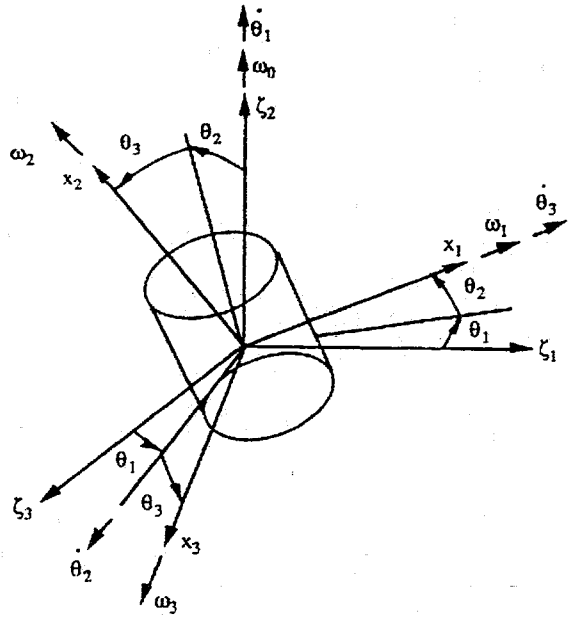

Figure 4: Spacecraft model with attitude angles θ_i, from [26].

the $(2,3,1)$ rotation sequence. The projections of the angular velocity with respect to a fixed inertial frame on the body-fixed frame x are denoted by ω_i while ω_0 is defined as above. The time derivatives of the Tait–Bryan angles are $\dot{\theta}_i$.

Assuming that the attitude has no influence on the orbit, the following equations of motion can be derived [24]

$$I\dot{\omega} = 3\omega_0^2\tilde{\xi}I\xi - \tilde{\omega}I\omega + u \tag{7}$$

with I the diagonal matrix of the moments of inertia I_i about the body-fixed frame and [25]

$$\omega = R(\theta)\dot{\theta} + \omega_c(\theta) \tag{8}$$

$$R(\theta) = \begin{bmatrix} \sin\theta_2 & 0 & 1 \\ \cos\theta_2\cos\theta_3 & \sin\theta_3 & 0 \\ -\cos\theta_2\sin\theta_3 & \cos\theta_3 & 0 \end{bmatrix} \tag{9}$$

$$\omega_c(\theta) = \begin{bmatrix} \omega_0\sin\theta_2 \\ \omega_0\cos\theta_2\cos\theta_3 \\ -\omega_0\cos\theta_2\sin\theta_3 \end{bmatrix} \tag{10}$$

and

$$\xi = \begin{bmatrix} -\sin\theta_1\cos\theta_2 \\ \cos\theta_1\sin\theta_3 + \sin\theta_1\sin\theta_2\cos\theta_3 \\ \cos\theta_1\cos\theta_3 - \sin\theta_1\sin\theta_2\sin\theta_3 \end{bmatrix} \tag{11}$$

while for any $v = [v_1, v_2, v_3]^\top$

$$\tilde{v} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}.$$

The R, ω_c, and ξ are related to the direction cosine matrix for the rotation from the x to the ζ coordinate frame. The moments u are due to jets aligned with the principal axes of inertia.

Using relations (7)–(11) a state space model can be derived. Using as state x of the model the Tait–Bryan angles θ and the angular velocities ω, so that $x = [\theta_i, \omega_i]^\top$ (note that the states x are different from the coordinate frame x_i), the following set of first-order differential equations is found [25]

$$\dot{x} = \begin{bmatrix} R^{-1}(\theta)(\omega - \omega_c(\theta)) \\ I^{-1}(-\tilde{\omega} I \omega + 3\omega_0^2 \tilde{\xi} I \xi) \end{bmatrix} + \begin{bmatrix} 0 \\ I^{-1} \end{bmatrix} u.$$

It can be worked out [26] (in our case with the help of MAPLE although the computation is not very complicated) to yield

$$\dot{x} = \begin{bmatrix} (x_5 \cos x_3 - x_6 \sin x_3)(\cos x_2)^{-1} - \omega_0 \\ x_5 \sin x_3 + x_6 \cos x_3 \\ (x_6 \sin x_3 - x_5 \cos x_3) \sin x_2 (\cos x_2)^{-1} + x_4 \\ (x_5 x_6 - 3\omega_0^2 \xi_2 \xi_3)(I_2 - I_3) I_1^{-1} \\ (x_6 x_4 - 3\omega_0^2 \xi_3 \xi_1)(I_3 - I_1) I_2^{-1} \\ (x_4 x_5 - 3\omega_0^2 \xi_1 \xi_2)(I_1 - I_2) I_3^{-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ I_1^{-1} u_1 \\ I_2^{-1} u_2 \\ I_3^{-1} u_3 \end{bmatrix}$$

This system satisfies the conditions of Theorem 1. We have to construct three functions λ_i because we have three inputs also. The result is

$$\lambda_1 = x_1 \qquad \lambda_2 = x_2 \qquad \lambda_3 = x_3$$

and this is easily verified, because the λ_i's correspond with the three degrees of freedom. With these outputs the vector relative degree is $r = (2, 2, 2)$.

Using these outputs the construction of a linearizing state-feedback and change of coordinates is a straightforward but tedious calculation because some intermediate expressions are lengthy, but the resulting state-feedback is not all that involved. The following log of a MAPLE session, with some edits, shows how the results are obtained by NONLINCON.

```
* finding functions for exact linearization, (outputfunc)*
no states are found for which the matrix A turns out to be singular

the function(s) lambda that fulfil the demands are:
                        [ x[1], x[2], x[3] ]

* exact linearization of the state input equations, (statelin)*
```

the exact linearizing feedback: (u)

```
                                                              2
 s[2] I[1] v[1] + I[1] v[3] + I[1] c[3] x[5]  s[3]

                    2                                    2
      + 2 I[1] c[3]  x[5] x[6] - I[1] s[3] x[6]  c[3]

      - x[5] x[6] I[2] + x[5] x[6] I[3]

                   2
      + 3 omega[0]  xi[2] xi[3] I[2]

                   2
      - 3 omega[0]  xi[2] xi[3] I[3] - I[1] x[6] x[5],

          2
 - (- c[2]  I[2] c[3] v[1] - c[2] I[2] s[3] v[2]

                      2     2                          3
      + s[2] I[2] c[3]  x[5]  s[3] + 2 s[2] I[2] c[3]  x[5] x[6]

                           2     2
      - s[2] I[2] s[3] x[6]  c[3]  - I[2] x[6] x[4] c[2]

      + c[2] I[3] x[6] x[4] - c[2] x[6] x[4] I[1]

                          2
      - 3 c[2] I[3] omega[0]  xi[3] xi[1]

                     2
      + 3 c[2] omega[0]  xi[3] xi[1] I[1]

                    2
      - s[3] I[2] x[6]  s[2])/c[2],

                                                        2
 - (- c[2] I[2] x[4] x[5] + c[3] I[3] x[6]  s[2]

               3     2            2
      + I[3] c[3]  x[5]  s[2] + c[2]  I[3] s[3] v[1]

                     2                                          2
      - 3 c[2] omega[0]  xi[1] xi[2] I[1] - 2 s[2] I[3] c[3] x[5]

                                                   2
      - c[2] I[3] c[3] v[2] - 2 s[3] s[2] I[3] c[3]  x[5] x[6]
```

```
                                                          2          3
      + 2 s[3] s[2] I[3] x[6] x[5] - I[3] x[6]  s[2] c[3]

      + I[3] x[5] x[4] c[2] + c[2] x[4] x[5] I[1]

                                 2
      + 3 c[2] I[2] omega[0]  xi[1] xi[2])/c[2]

* transformation to the normal form, (transform)*

      c[3] x[5] - s[3] x[6] - omega[0] c[2]
x[1], ------------------------------------, x[2], s[3] x[5]
                        c[2]

        + c[3] x[6],

        s[2] c[3] x[5] - s[2] s[3] x[6] - x[4] c[2]
x[3], - -------------------------------------------
                              c[2]
```

The linearizing state-feedback and change of coordinates in normalized, though not in their most compact form are listed above, but are easier to read as

$$
\begin{aligned}
u_1 &= s_2 I_1 v_1 + I_1 v_3 + I_1 c_3 x_5^2 s_3 + 2 I_1 c_3^2 x_5 x_6 - I_1 s_3 x_6^2 c_3 \\
&\quad - x_5 x_6 I_2 + x_5 x_6 I_3 + 3\omega_0^2 \xi_2 \xi_3 I_2 - 3\omega_0^2 \xi_2 \xi_3 I_3 - I_1 x_6 x_5 \\
u_2 &= -\left(-c_2^2 I_2 c_3 v_1 - c_2 I_2 s_3 v_2 + s_2 I_2 c_3^2 x_5^2 s_3 + 2 s_2 I_2 c_3^3 x_5 x_6 \right. \\
&\quad - s_2 I_2 s_3 x_6^2 c_3^2 - I_2 x_6 x_4 c_2 + c_2 I_3 x_6 x_4 - c_2 x_6 x_4 I_1 \\
&\quad \left. - 3 c_2 I_3 \omega_0^2 \xi_3 \xi_1 + 3 c_2 \omega_0^2 \xi_3 \xi_1 I_1 - s_3 I_2 x_6^2 s_2 \right) / c_2 \\
u_3 &= -\left(c_3 I_3 x_6^2 s_2 - c_2 I_2 x_4 x_5 + I_3 c_3^3 x_5^2 s_2 + c_2^2 I_3 s_3 v_1 - 3 c_2 \omega_0^2 \xi_1 \xi_2 I_1 \right. \\
&\quad - 2 s_2 I_3 c_3 x_5^2 - c_2 I_3 c_3 v_2 - 2 s_3 s_2 I_3 c_3^2 x_5 x_6 + 2 s_3 s_2 I_3 x_6 x_5 \\
&\quad \left. - I_3 x_6^2 s_2 c_3^3 + I_3 x_5 x_4 c_2 + c_2 x_4 x_5 I_1 + 3 c_2 I_2 \omega_0^2 \xi_1 \xi_2 \right) / c_2
\end{aligned}
$$

$$
\Phi = \begin{bmatrix} x_1 \\ (c_3 x_5 - s_3 x_6 - \omega_0 c_2)/c_2 \\ x_2 \\ s_3 x_5 + c_3 x_6 \\ x_3 \\ -(s_2 c_3 x_5 - s_2 s_3 x_6 - x_4 c_2)/c_2 \end{bmatrix}
$$

making use of the LaTeX output capabilities of MAPLE. The output is compressed using the aliases

$$c_i = \cos x_i, \qquad s_i = \sin x_i$$

and with the expression for ξ. In this example there are no conditions reported for the singularity of the matrix A so the linearizing state-feedback

seems to be well defined in the whole state space $\mathbb{R}^6$. It is however easy to see that when $\cos x_2 = 0$, both the state-feedback and change of coordinates are not defined.

Example 4. A more involved variant of Example 3 is the following. Consider in addition three rotors aligned with the principal axes of inertia of the satellite and with moments of inertia J_i about the spin axes. The inertia matrix I also accounts for the rotors. Now it is not only possible to control the satellite with moments about the principal axes, supplied by jets, but also with the rotors. Denoting the angular coordinates of the rotors with φ_i, the equations of motion for this six degrees-of-freedom model become [24]

$$\begin{aligned} I\dot{\omega} + J\ddot{\varphi} &= 3\omega_0^2 \tilde{\xi} I \xi - \tilde{\omega}(I\omega + J\dot{\varphi}) + u_{1:3} \\ J\dot{\omega} + J\ddot{\varphi} &= u_{4:6}, \end{aligned}$$

where $u_{1:3}$ is due to jet control moments and $u_{4:6}$ to the rotor control moments. Both input columns consist of three elements.

Using as state of the model $x = [\theta_i, \varphi_i, \omega_i, \dot{\varphi}_i]^\mathsf{T}$, the following set of first order differential equations is found:

$$\dot{x} = \begin{bmatrix} R^{-1}(\theta)(\omega - \omega_c(\theta)) \\ \dot{\varphi} \\ \begin{bmatrix} I & J \\ J & J \end{bmatrix}^{-1}_{(1:3)} \left[3\omega_0^2 \tilde{\xi} I \xi - \tilde{\omega}(I\omega + J\dot{\varphi}) \right] \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \begin{bmatrix} I & J \\ J & J \end{bmatrix}^{-1} \end{bmatrix} u.$$

where $(1:3)$ denotes the first three columns of the matrix. This can again be worked out with MAPLE.

This system satisfies the conditions of Theorem 1 also. The result obtained with the NON-CON function outputfunc is as follows.

```
* finding functions for exact linearization, (outputfunc)*
no states are found for which the matrix A turns out to be singular

the function(s) lambda that fulfill the demands are:
            [ x[2], x[4], x[6], x[5], x[1], x[3] ]
```

Using these functions as output h, the construction of a linearizing state-feedback and change of coordinates is straightforward. The results are not presented because they are voluminous.

7 Conclusion and Discussion

The solution of the exact linearization problem can be automated by using symbolic computation, e.g., by using the NON-CON package. This means that it is easier now to use controllers based on the linearization approach,

that can fully take into account the non-linearities in real systems. An enhancement of the performance of some control systems, for a large set of operation conditions, can therefore be expected.

At the moment, the computations cannot be performed for complicated systems, mainly because the possibilities to solve sets of differential equations are limited. Therefore, the designers of control systems cannot yet routinely compute a solution for the exact linearization problem, using tools that are based on symbolic computation programs.

To remedy this, we recommend extending the capabilities of symbolic computation programs for solving sets of differential equations. Another possibility is to use another algorithm, that completely avoids the integration step and hopefully does not introduce other operations of high complexity.

Future research will have to aim also at:

- improving the capabilities for solving sets of non-linear (differential) equations,
- devising new algorithms or modifying existing algorithms to be more efficient in space and time,
- implementing the algorithms in a more efficient way, especially with regard to computer memory requirements,
- solving more small and larger scale problems, to further guide in the selection of pressing lines of research.

Acknowledgement

Harm van Essen implemented the algorithms in MAPLE and is the author of the core part of NONCON.

References

[1] T. Umeno, K. Abe, S. Yamashita and O. Saito, A software package for control design based on the algebraic theory using symbolic manipulation language REDUCE, In: *Proc.* TENCON 87 (Seoul, Korea, Aug. 1987), vol 3, IEEE, 1102–1106.

[2] O. Saito, M. Kanno and K. Abe, Symbolic manipulation CAD of control engineering by using REDUCE, *Int. J. Control* **48** (1988), 781–790.

[3] T. Umeno, S. Yamashita, O. Saito and K. Abe, Symbolic computation application for the design of linear multivariable control systems, *J. Symb. Comput.* **8** (1989), 581–588.

[4] T. Mimpen, Design of a nonlinear observer with REDUCE, *Trainee project*, Eindhoven University of Technology, Dept. of Mechanical Engineering, Aug. 1990, *Report* WFW 90.028 (In Dutch).

[5] H.A. Barker, Y.W. Ko and P. Townsend, The application of a computer algebra system to the analysis of a class of nonlinear systems, In: *Nonlinear Control Systems Design: Science papers of the IFAC Symposium*, (Capri, Italy), IFAC, Pergamon Press, Oxford, 1989, 131–136.

[6] J. Birk and M. Zeitz, Anwendung eines symbolverarbeitenden Programmsystems zur Analyse und Synthese von Beobachtern für nichtlineare Systeme, *Messen, Steuern, Regeln* **33** (1990), 536–543.

[7] R. Rothfuß, J. Schaffner, and M. Zeitz, Rechnergestützte Analyse und Synthese nichtlinearer Systeme, In: *Nichtlineare Regelung: Methoden, Werkzeugen, Anwendungen*, Vol. 1026 of *VDI Berichte*, VDI-Verlag, Düsseldorf, 1993, 267–291.

[8] H. Hahn, K.-D. Leimbach and X. Zhang, Nonlinear control of a spatial multi-axis servo-hydraulic test facility, In: *Preprints of the 12th IFAC World Congress*, Vol. 7, (Sydney, Australia), IFAC, 1993, 267–270.

[9] F. Allgöwer and E.D. Gilles, Nichtlinearer Reglerentwurf auf der Grundlage exakter Linearisierungstechniken, In: *Nichtlineare Regelung: Methoden, Werkzeugen, Anwendungen*, Vol. 1026 of *VDI Berichte*, VDI-Verlag, Düsseldorf, 1993, 209–234.

[10] D.W.C. Ho, J. Lam, S.K. Tin, and C.Y. Han, Recent applications of symbolic computation in control system design, In: *Preprints of the 12th IFAC World Congress*, Vol. 9, (Sydney, Australia), IFAC, 1993, 509–512.

[11] O. Akhrif and G.L. Blankenship, Computer algebra for analysis and design of nonlinear control systems, In: *Proc. of the 1987 American Control Conf.*, Vol. 1, (Minneapolis, MN), IEEE, 1987, 547–554.

[12] O. Akhrif and G.L. Blankenship, Computer algebra algorithms for nonlinear control, In: *Advanced Computing Concepts and Techniques in Control Engineering* (eds. M.J. Denham and A.J. Laub), vol. 47 of *NATO ASI Series F*, Springer-Verlag, Berlin, 1988, 53–80.

[13] R. Wang, Symbolic computation approach for absolute stability, *Int. J. Control* **58** (1993), 495–502.

[14] B. de Jager, Nonlinear control system analysis and design with Maple, In: *Artificial Intelligence, Expert Systems and Symbolic Computing* (eds. E. N. Houstis and J. R. Rice), Selected and revised papers from the IMACS 13th World Congress, (Dublin, Ireland, July 1991), IMACS, North-Holland, Amsterdam, 1992, 155–164.

[15] H. van Essen and B. de Jager, Analysis and design of nonlinear control systems with the symbolic computation system Maple, In: *Proc. of the Second European Control Conf.* (eds. J. W. Nieuwenhuis, C. Praagman and H. L. Trentelman), Vol. 4, (Groningen, The Netherlands), 1993, 2081–2085.

[16] B. de Jager, Symbolic calculation of zero dynamics for nonlinear control systems, In:*Proc. of the 1991 Internat. Symp. on Symbolic and Algebraic Computation, ISSAC'91* (ed. S. M. Watt), (Bonn, Germany), ACM Press, New York, 1991, 321–322.

[17] B. de Jager, Zero dynamics and input-output exact linearization for systems without a relative degree using Maple, In: *Proc. of the 1993 Internat. Symp. on Nonlinear Theory and its Applications*, Vol. 4, (Hawaii), IEICE, 1993, 1205–1208.

[18] A. Isidori, *Nonlinear Control Systems: An Introduction* (2nd edition), Springer-Verlag, Berlin, 1989.

[19] M. Fliess, Generalized controller canonical forms for linear and nonlinear dynamics, *IEEE Trans. Automat. Control*, **35** (1990), 994–1001.

[20] R. B. Gardner and W. F. Shadwick, The GS algorithm for exact linearization to Brunovsky normal form, IEEE *Trans. Automat. Control* **37** (1992), 224–230.

[21] H. van Essen, Symbols speak louder than numbers: Analysis and design of nonlinear control systems with the symbolic computation system MAPLE, *Master's thesis*, Eindhoven University of Technology, Dept. of Mechanical Engineering, June 1992, *Report* WFW 92.061.

[22] B. de Jager, The use of symbolic computation in nonlinear control: is it viable?, In: *Proc. of the 32nd IEEE Conf. on Decision and Control*, Vol. 1, (San Antonio, TX), IEEE, Piscataway NJ, 1993 276–281.

[23] B. de Jager, Symbolic solution for a class of partial differential equations, In: *Proc. of the Rhine Workshop on Computer Algebra* (ed. J. Calmet), (Karlsruhe, March 1994), 1994.

[24] A. A. Anchev, Equilibrium attitude transitions of a three-rotor gyrostat in a circular orbit, *AIAA J.* **11** (1973), 467–472.

[25] S. N. Singh and A. Iyer, Nonlinear decoupling sliding mode control and attitude control of spacecraft, IEEE *Trans. Aerospace Electron. Systems* **25** (1989), 621–633.

[26] H. Elmali and N. Olgac, Robust output tracking via sliding mode control with perturbation estimation, In: *Preprints of the 12th IFAC World Congress*, Vol. 8, (Sydney, Australia), IFAC, 1993, 489–492.