The use of symbolic computation in nonlinear control: is it viable?

Proceedings of the 32nd Conference
on Decision and Control
San Antonio, Texas • December 1993

The use of symbolic computation in nonlinear control: Is it viable?

Bram de Jager
Department of Mechanical Engineering, WH 2.137
Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
Email: jag@wfw.wtb.tue.nl Fax: +31 40 447355

Abstract

To aid in the analysis and design of nonlinear control systems the NONSICON package, an acronym for Nonlinear Control, has been developed. This paper addresses the usefulness of the NONSICON package as a self contained tool in the symbolic analysis and design of nonlinear control systems, as a replacement and/or complement for numerical tools, and as a substitute for analytical (paper and pencil) work. The symbolic computation program MAPLE is used as a computing substrate for NONSICON.

To assess the viability of this tool, several algorithms available for analysis and design are implemented in NONSICON, and applied to examples of textbook problems with published solutions and also to a larger scale problem. The examples in the paper address the following symbolic computational areas for nonlinear dynamic systems: (1) the normal form, (2) the zero dynamics, (3) the input-output exact linearizing state-feedback, (4) the state space exact linearizing state-feedback, including the search for output functions that give the system a full or maximal order relative degree. The systems are not required to have a well defined relative degree.

The textbook problems show that the NONSICON package can be successfully used. The larger scale problem is, however, too complex to be solved with the current versions of NONSICON and MAPLE. Some recommendations to improve the implementation and to make MAPLE more suitable for use with NONSICON are given. The conclusion is that symbolic computation is a viable approach for straightforward textbook problems, but not grown up enough to tackle larger problems. However, we envision substantial future progress.

1. Introduction

The use of symbolic computation programs for control purposes is investigated by several researchers. Some control problems are handled by REDUCE, see [1]. The use of MACSYMA is discussed in [2] for a specific class of control problems. Zeitz *et al.*, [3], use the program MACNON, based on MACSYMA, to analyze observability and reachability, and to design observers and controllers for nonlinear systems. Their package is mainly used for teaching, and there are no published results for larger scale problems. Blankenship [4] also used MACSYMA to solve some control problems with his implementation CONDENS, but is now switching to MATHEMATICA, and he 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. The use of MAPLE for several control problems is reported in [5]. Some problems reported in this paper, *e.g.*, with solving partial differential equations, are partly resolved in [6]. They describe a MAPLE package, here called NONSICON (a successor of the ZERODYN package presented and used in [5, 7]), 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 applying it to some textbook and a larger scale problem. Contrary to [7], that focusses on the computation of the zero dynamics, and to [8], that treats the state space exact linearization problem, both with the assumption that the relative degree is well defined, here a more thorough treatment and a larger scale example is presented, that more closely resembles problems arising from practice.

The main goals and contributions of this paper are

- a proof of the viability of symbolic computation for problems in the analysis and design of nonlinear control systems
- to show the characteristics of a prototype implementation for the computation of normal forms, the zero dynamics, input-output and state space exact linearization
- to give some examples (also a larger scale or semi-industrial one) of the use of this implementation and to document some applications
- to discuss directions for future research in symbolic computation and in the analysis and synthesis for nonlinear control systems
- to familiarize a larger audience in the control community with the use of symbolic computation.

The paper is structured as follows. First, Section 2 presents and makes some remarks on the control problems that are treated in this study. Then, Section 3 discusses the mathematical details of the problems. Section 4 follows with a solution of the problems, the algorithms used, and an investigation of the functionality needed. The general structure of the implementation in MAPLE of the algorithms needed to solve these problems is treated in Section 5, where the NONSICON package is described. Section 6 presents some textbook examples, using the NONSICON package, for each of the problem areas. The next section contains a discussion of the larger scale problem. Section 8 presents some conclusions and gives directions for future research.

2. The Control Problems

From several areas in nonlinear control, where symbolic computation is likely to be of some profit, we discuss the computation of the normal form, the zero dynamics, and the input-output and state space exact linearization. To be able to solve some of these problems, modifications of the system to be controlled are necessary. 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 the manipulation of the information flowing out of the system. The generation of control signals will be done by a control law, where information of the system is used to generate the control input.

2.1. Normal form

To reveal the structure inherent in the system, but disguised by the general form of the mathematical model, it is necessary to uncover the structure. This structure can then be used to advantage in the analysis and design of control systems. It can be revealed by transformation to standard, canonical or, as we call it here, normal forms. For this purpose a change of coordinates is employed. The new coordinates are related to the system output and its derivatives. Depending on the *relative degree* of the system, *i.e.*, the number of times the outputs have to be differentiated before the input explicitly appears, the system can be decomposed in a set of series connections of integrators, a feedback over these sets, and, if the relative degree is less then the system order, a remaining part that can be made "unobservable" at the output of the system by feedback, the *zero dynamics*. This will be made more precise in the next section.

2.2. Zero dynamics

The zero dynamics of a nonlinear system can be characterized as the remaining dynamics of the system if the output is required to be 0 for all times. It is the dynamics of the system on the largest unobservable submanifold that can be obtained by judiciously manipulating the input to the system by a control law. The characterization of a system by properties of its zero dynamics is of importance for some design goals, *e.g.*, if the zero dynamics is unstable, certain types of control laws are unable to stabilize the system, so these control laws should be avoided.

2.3. Input-output exact linearization

The input-output exact linearization problem is of long standing interest in control theory. In essence, it is the problem of modifying a nonlinear dynamic system such that, after the modification, it behaves like a linear one, in the input-output sense, *i.e.*, the goal is to get a linear (dynamical) relation between the new input and the output of the plant.

In a more complete control system design, the input-output exact linearization is often only a subordinate goal, to make it possible to use other design methods for attaining additional goals.

2.4. State space exact linearization

It is also possible to consider a more ambitious goal, where the behavior between the new input and the (transformed) state of the plant is required to be linear. This exact (or state space exact) linearization problem is also of long standing interest in control theory. In essence, it is the problem of modifying a nonlinear dynamic system such that, after the modification, it behaves like a linear one, so powerful design methods for linear systems can again be employed.

3. The Mathematical Formulation of the Problems

In the presentation of the mathematics, we closely follow the work of Isidori [9]. We start with a nonlinear model of a plant and assume that it can be described adequately by a set of nonlinear differential equations, affine in the input u, and without direct feed-through from input to output

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

with *state* vector $x \in R^n$, containing all necessary information of the plant, *input* vector $u \in R^m$, and *output* vector $y \in R^m$. 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. Parts of the theory can also be derived if the number of inputs is larger than the number of outputs. The vector field f is smooth, g has m columns g_i of smooth vector fields, and h is a column of m scalar-valued smooth functions h_i.

The nonlinear system (1) is said to have a *vector relative degree* $\{r_1, \ldots, r_m\}$ at $x = x^o$ if

1. $L_{g_j} L_f^k h_i(x) = 0$ for $k = 1, \ldots, r_i - 2$ $(i, j = 1, \ldots, m)$ and all x in a neighborhood of x^o,
2. the following $m \times m$ matrix is nonsingular at x^o

$$A(x) = \begin{bmatrix} L_{g_1} L_f^{r_1 - 1} h_1(x) & \cdots & L_{g_m} L_f^{r_1 - 1} h_1(x) \\ \vdots & \ddots & \vdots \\ L_{g_1} L_f^{r_m - 1} h_m(x) & \cdots & L_{g_m} L_f^{r_m - 1} h_m(x) \end{bmatrix}.$$

Here $L_f^k h_i(x)$ means the k^{th} successive Lie derivative of the scalar function $h_i(x)$ in the direction of the vector field f, e.g., $L_f h_i(x) = (\partial h_i(x)/\partial x) f(x)$. The matrix A is sometimes called the *decoupling matrix*.

The modification we allow is *state-feedback*, *i.e.*, a feedback based on the explicit knowledge of the value of the state vector $x(t)$. The type of control law used here is restricted to *static* state-feedback. Hence, the value of the input vector $u(t)$ depends on the state $x(t)$ and a new *reference* input vector $v(t)$. This dependence is of the form

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

where the components α_i and β_{ij} are smooth functions.

For linear systems a linear change of coordinates $z = Tx$ with a nonsingular matrix T is usually adequate. For nonlinear systems it is more appropriate to allow for a nonlinear change of coordinates

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

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, because then Φ can be used as a two-way mapping.

When the system has a well defined relative degree we can use such a coordinate transformation to transform (1), under some involutivity conditions for MIMO systems, to the *normal form*

$$\begin{aligned} y_i = h_i(x) &= \xi_i^1 \\ \dot{\xi}_i^1 &= \xi_i^2 \\ &\cdots \\ \dot{\xi}_i^{r_i} &= b_i(\xi, \eta) + \sum_{j=1}^{m} a_{ij}(\xi, \eta) u_j \qquad \text{for } i = 1, \ldots, m \\ \dot{\eta}_i &= q_i(\xi, \eta) \qquad \text{for } i = 1 + \sum_{j=1}^{m} r_j, \ldots, n \end{aligned} \tag{4}$$

where

$$\begin{aligned} a_{ij}(\xi, \eta) &= L_{g_j} L_f^{r_i - 1} h_i(\Phi^{-1}(\xi, \eta)) \\ b_i(\xi, \eta) &= L_f^{r_i} h_i(\Phi^{-1}(\xi, \eta)) \qquad \text{for } i, j = 1, \ldots, m. \end{aligned}$$

The terms a_{ij} are the entries of matrix A and therefore we can compactly write (with only equations containing the input u in (4))

$$\begin{aligned} \xi^{(r)} &= b(\xi, \eta) + A(\xi, \eta) u \\ \dot{\eta} &= q(\xi, \eta), \end{aligned}$$

where $\xi^{(r)}$ are the m elements of ξ on the places $r_1, r_1 + r_2, \ldots, r$, with $r = \sum_{j=1}^{m} r_j$. Because A is nonsingular if the relative degree is well defined, the control

$$u = A^{-1}(v - b) \tag{5}$$

with the new input v is properly defined and linearizes the part of system (4) that is visible at the output of the system

$$\xi^{(r)} = v.$$

The nonlinear dynamics obtained when the output $y = h(x)$ is restricted to 0 by suitable initial conditions for ξ, *i.e.*, $\xi = 0$, and a suitable control v in (5), *i.e.*, $v = 0$, is

$$\dot{\eta} = q(0, \eta), \quad \eta(0) = \eta^o.$$

It is invisible at the output, and is called the *zero dynamics* of the system, because the dynamics is related to the zeros for linear systems, and also because it *is related to the zero output*. When the involutivity conditions for MIMO systems are not fulfilled, the zero dynamics will also depend on u.

We can now state our problems more formally.

The problem of transforming the system (1) to the *normal form* amounts to setting up the transformation, computing the inverse transformation and *splitting the system* in the standard part and in the part associated with the zero dynamics. Of course, this can be done if the relative degree is well defined only. If there is no relative degree we have to resort to another method to derive the equations for the zero dynamics and to solve the input-output exact linearization problem (see the following sections).

The *zero dynamics* problem is to obtain the dynamics of the system when the output y is required to be 0 for all t, by a proper choice of initial state $x(0)$ and control input $u(t)$. Here we have to employ an appropriate static state-feedback and use proper initial conditions. More specific: we are looking for the *locally maximal output zeroing* submanifold, and its associated dynamics.

For systems with a well defined relative degree, the zero dynamics follows quite easily from the normal form. For systems without a relative degree the situation is more complicated. This problem has been treated in [9, Section 6.1], and the solution does not require the system to have a well defined relative degree. Our aim is to implement the solution algorithm and compute the zero dynamics for real systems.

The *input-output exact linearization* problem: under which conditions is it possible to transform the system (1) to a linear one by state-feedback (2)? The linearity property should be established between the new input v and the output y. Formally, we are looking for a neighborhood U of x^o and a static state-feedback such that for all $k \geq 0$ and all $1 \leq i, j \leq m$ the expression $L_{(g\beta)_i} L_{f+g\alpha}^k h_j(x)$ is independent of x on U.

For systems with a well defined relative degree, the input-output linearizing feedback follows quite easily from the normal form. For systems without a relative degree the situation is more complicated. This has also been solved, see, *e.g.*, [9, Section 5.4]. Our goal is to test the conditions under which the problem can be solved and to derive explicit expressions α and β for the feedback (2).

The *state space exact linearization problem* can be stated as: 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.*, [9], 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 solution is only valid for systems with a well defined relative degree, and requires the existence of (synthetic) outputs for which the system has a *full order* relative degree, $r = n$.

When a full order relative degree cannot be obtained, it is sometime convenient to strive for a maximal relative degree. Then the corresponding input-output linearizing state-feedback realizes a minimal dimension of the zero dynamics.

4. The Solution of the Problems

4.1. Normal forms

An algorithm to compute the normal form is inherent to the definition of the normal form in the previous section, and needs no further discussion.

A symbolic computation program should provide facilities for computing (Lie) derivatives, testing involutiveness, computing solutions of sets of nonlinear algebraic equations for the inverse mapping, and of sets of partial differential equations to transform to a simple normal form. Besides, more mundane facilities like symbolic substitution are needed.

4.2. Zero dynamics

When the system has a well defined relative degree, the zero dynamics follows from the normal form, by substitution of the output nulling input u^* and using the property that the states ξ, that do not belong to the zero dynamics, can be set to 0.

For systems without a relative degree the zero dynamics can be computed by using the *Zero Dynamics Algorithm*. The way this algorithm works is by considering a sequence of nested submanifolds M_i, with $M_i \supset M_{i+1}$ and $M_0 = h^{-1}(0)$, *i.e.*, the first submanifold is the inverse image of the point $y = 0$. When some conditions are fulfilled this sequence converges to the locally maximal output zeroing submanifold Z^* in some neighborhood of x^o and there exists a mapping u^* such that $f^*(x) = f(x) + g(x)u^*(x)$ is tangent to Z^*. The pair (Z^*, f^*) is called the zero dynamics of the system. When the mapping $H(x)$ is defined in a neighborhood U of x^o by $Z^* \cap U = \{x \in U : H(x) = 0\}$ the input u^* can be computed as the solution of $L_f H(x) + L_g H(x) u^* = 0$. For further details of this algorithm, and the conditions to be imposed to guarantee convergence of the sequence to Z^*, we refer to [9].

An analysis of the Zero Dynamics Algorithm shows that a symbolic computation program should be able to compute the Lie derivative, the Jacobian, the rank, the Gauss Jordan decomposition, and the determinant of a matrix. The main problems are the computation of solutions for sets of nonlinear equations and the computation of the kernel of a matrix. The implementation of this algorithm is still experimental, *e.g.*, transformation to an *extended normal form* is not yet possible.

4.3. Input-output exact linearization

For systems with a well defined relative degree the input-output exact linearization follows easily from the normal form, and is in fact given by (5).

For systems without a relative degree the static state-feedback which solves the input-output exact linearization problem can be computed with the *Structure Algorithm*. This algorithm uses a sequence of Toeplitz matrices

$$M_k(x) = \begin{bmatrix} T_0(x) & \cdots & T_k(x) \\ \vdots & \ddots & \vdots \\ 0 & \cdots & T_0(x) \end{bmatrix}$$

where

$$T_k(x) = \begin{bmatrix} L_{g_1} L_f^k h_1(x) & \cdots & L_{g_m} L_f^k h_1(x) \\ \vdots & \ddots & \vdots \\ L_{g_1} L_f^k h_m(x) & \cdots & L_{g_m} L_f^k h_m(x) \end{bmatrix} \quad \text{for } 0 \le k \le 2n-1.$$

The conditions for existence can be expressed in rank conditions on $M_k(x)$. In the course of the Structure Algorithm the functions $\gamma_1, \ldots, \gamma_q$ are computed. With

$$\Gamma(x) = \begin{bmatrix} \gamma_1(x) \\ \cdots \\ \gamma_q(x) \end{bmatrix}$$

we can solve

$$(L_g \Gamma(x))\alpha(x) = -L_f \Gamma(x)$$
$$(L_g \Gamma(x))\beta(x) = -\begin{bmatrix} I & 0 \end{bmatrix}$$

for α and β in a neighborhood of x^o. For further details of this algorithm we refer to [9].

An analysis of the Structure Algorithm shows that a symbolic computation program should be able to compute the Lie derivative, perform matrix-vector and matrix-matrix multiplication, check if some expression is equal to 0, perform rank tests, compute the kernel of a mapping, etc. These computations are relatively straightforward. Here also, the main problem is to find solutions for sets of nonlinear equations.

4.4. State space exact linearization

The conditions for solving the exact linearization problem are given. To state the solution more compactly, we define the so-called distributions

$$G_i = \text{span}\{\text{ad}_f^k g_j : 0 \le k \le i, 1 \le j \le m\} \quad \text{for } 0 \le i \le n-1.$$

To define the adjoint ad, we need the Lie product

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

and ad is defined recursively as $\text{ad}_f^k g_i = [f, \text{ad}_f^{k-1} g_i]$ with $\text{ad}_f^0 g_i = g_i$.

We now state the conditions for a solution of the state space exact linearization problem [9, Theorem 5.2.4].

Theorem 4.1. Suppose a system

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

with rank $g(x^o) = m$ 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^o for each $0 \le i \le n-1$
2. G_{n-1} has dimension n
3. G_i is involutive for each $0 \le i \le n-2$.

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.

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 given conditions are fulfilled, there exist solutions $\lambda_i(x), i = 1, \ldots, m$, for the following partial differential equations

$$L_{g_j} L_f^k \lambda_i(x) = 0, \quad \text{for } 0 \le k \le r_i - 2 \text{ and } 0 \le j \le m. \tag{6}$$

Also $\sum_{i=1}^m r_i = n$, where the set of integers $\{r_1, \ldots, r_m\}$ is the relative degree vector. The m functions λ_i can be computed, based on a constructive proof of Theorem 4.1.

Using the functions λ_i, the change of coordinates and state-feedback $u = \alpha(x) + \beta(x)v$ that solve the state space exact linearization problem can be computed.

An analysis of this algorithm 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. We will discuss this in Section 5.

5. Solution with Symbolic Computation

Solutions for the problems described in the previous section, *i.e.*, to compute

1. the normal form
2. the zero dynamics
3. the input-output exact linearization
4. the state space exact linearization

are included in NONSICON. For problems (1) and (4) the system should have a well defined relative degree, for (2) and (3) this is not necessary. Implementations of algorithms for some other problems, *e.g.*, to solve partial differential equations, are included in this package also. The structure of the implementation is sketched in Fig. 1.

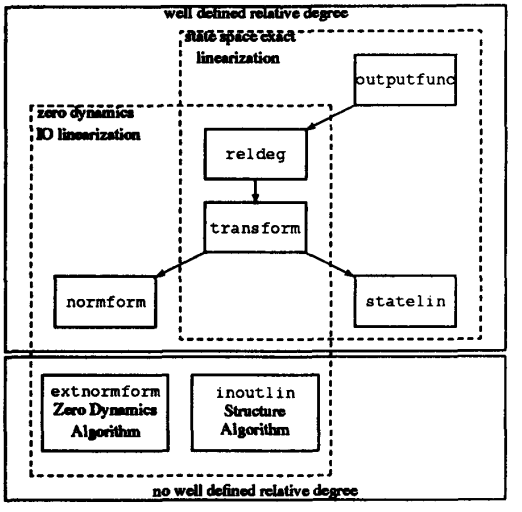

Figure 1: Structure of NONSICON

Because the NONSICON package is ample documented in [10], there is no need for repetition here. For a more accessible account, see [6]. Here, we only focus on some specific problems that appeared during the implementation of the algorithms presented in Section 4. These problems present the major bottlenecks for the symbolic solution of our design problems.

In the previous section we noted that a main problem was to compute solutions for sets of nonlinear equations. In MAPLE a facility is available to solve sets of nonlinear equations, namely the `solve` function, but this facility is not always sufficient. We did not try to improve this.

In the previous section we also remarked that another 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. To improve this the following route was chosen.

The partial differential equations we have to solve are from the "completely integrable" type, in other words, 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.

The solution of the partial differential equation can be constructed from the solutions of related sets of ordinary differential equations. Because MAPLE provides some facilities for this type of problems, 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, to handle a larger class of problems, by implementing a recursive procedure to solve sets of differential equations, 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. Despite this extension, the computation of the functions λ_i is often unsuccessful, especially for more complicated systems, so NONSCON cannot finish the computations. This part of NONSCON should therefore be considered as experimental. It seems unlikely that another symbolic computation program will improve this situation.

For some other minor problems, that were encountered during the implementation of the algorithms, work arounds are applied. We discuss some of these work arounds. The problem was that some of the standard MAPLE functions in the `linalg` package are only suitable for rational polynomials. This was too limited for our purposes. Therefore, the `rank` and implicitly the `gausselim` and `gaussjord` procedures were extended, so a larger class of problems could be handled. The modification consisted simply in removing the check for the type of the entries of the matrix for which the rank, Gauss elimination or Gauss Jordan form should be computed, and adding an additional call to the `simplify` function to assure that the detection of zero expressions was guaranteed for a larger class of problems. The new functions `extrank`, `extgausselim`, and `extgaussjord` are therefore extended in a rather ad hoc way, and this part of NONSCON should be considered as experimental also.

6. Textbook problems

To illustrate the use of NONSCON we consider several examples. The first three examples do have a well defined relative degree. The first example computes the zero dynamics of a SISO system with two cases for input and output. The second example is for the zero dynamics computation of a MIMO system with two inputs and two outputs. The third example is for the input-output exact linearization problem, again for a SISO system. For the fourth example the zero dynamics is computed with the Zero Dynamics Algorithm. For the fifth the input-output linearizing state-feedback is computed with the Structure Algorithm. The last two examples are based on models that do not have a well defined relative degree. Both of these examples are MIMO with two inputs and two outputs. All examples are contrived ones. The first example is taken from [11, Example 12.43], and the next four from [9, Examples 7.4.1, 4.8.1, 6.1.2, 5.4.1 resp.].

Example 6.1. The model of this system, a robot with two revolute joints, see Fig. 2, can be derived with the method of Lagrange from the following expressions for the kinetic and potential energy T and V [11]

$$2T = m_1 l_1^2 \dot\theta_1^2 + m_2 \left(l_1^2 \dot\theta_1^2 + l_2^2 (\dot\theta_1 + \dot\theta_2)^2 + 2 l_1 l_2 (\dot\theta_1 + \dot\theta_2) \cos\theta_2 \right)$$
$$-V = g \{ (m_1 + m_2) l_1 \cos\theta_1 + m_2 l_2 \cos(\theta_1 + \theta_2) \} .$$

The inputs u_1 and u_2 are the joint torques and the outputs y_1 and y_2 the joint positions. The state x of the model corresponds with the degrees-of-freedom and their derivatives in the following way $x^T = \begin{bmatrix} \theta_1 & \theta_2 & \dot\theta_1 & \dot\theta_2 \end{bmatrix}$.

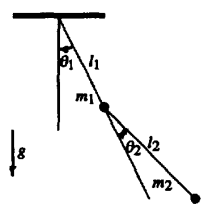

Figure 2: Robot with two revolute joints

The aim is to derive the dynamics of the model when it is constrained. There are two cases, the first one with $y = y_1 = \theta_1$ constrained to 0 and $u_2 = 0$, the second one with $y = y_2 = \theta_2$ constrained to 0 and $u_1 = 0$. According to [11] the zero dynamics are given by (with all model parameters set to 1)

$$\dot x_2 = x_4 \quad \dot x_4 = g \sin x_2 \qquad \text{resp.} \qquad \dot x_1 = x_3 \quad \dot x_3 = (3/5)\, g \sin x_1 .$$

The following log of a NONSCON session shows that the results of [11] can be reproduced. Two functions are used: `normform` to compute the zero dynamics (expressed in the new coordinates) and `transform` for the transformation to the new coordinates and its inverse. The log for the first case, where $y = \theta_1$.

```
* normform *
zero dynamics is
[eta[1]dot = eta[2], eta[2]dot = grav sin(eta[1])]
output nulling input is
                    2
[ sin(eta[1]) eta[2]  + cos(eta[1]) grav sin(eta[1]) ]
* transform *
transformation
[ x[1], x[3], x[2], x[4] ]
inverse transformation
{x[2]=eta[1], x[4]=eta[2], x[1]=zeta[1], x[3]=zeta[2]}
```

Then the second case for $y = \theta_2$.

```
* normform *
zero dynamics is
[eta[1]dot = eta[2], eta[2]dot = 3/5 grav sin(eta[1])]
output nulling input is   [ 1/5 grav sin(eta[1]) ]
* transform *
transformation
[ x[2], x[4], x[1], x[3] ]
inverse transformation
{x[4]=zeta[2], x[3]=eta[2], x[2]=zeta[1], x[1]=eta[1]}
```

The results coincide with the ones given above.

Example 6.2. The model of the system is

$$\dot x = \begin{bmatrix} x_1 + x_1 x_4 \\ x_2 e^{x_3} \\ x_2 + x_3^2 \\ x_1 + x_2 - x_4 + x_1 x_4 \end{bmatrix} + \begin{bmatrix} x_3 & 1 \\ 1 & 0 \\ 0 & 0 \\ 1 + x_3 & 1 \end{bmatrix} u, \qquad y = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} .$$

The output functions do not give the model a full order relative degree as shown by the following log of a MAPLE session where the `reldeg` function is used to compute the relative degree.

```
* reldeg *
vector relative degree = [ 1, 1 ]
total relative degree  = 2
matrix Adeg            = [ x[3]  1 ]
                         [   1   0 ]
       bdeg = [ x[1] + x[1] x[4], x[2] exp(x[3]) ]
```

To permit a linearizing state-feedback we have to find functions λ for which the model has a full order relative degree. The next log of a NONSCON session shows that this is possible. The function `outputfunc` is used to compute the λ's and the function `statelin` to compute the linearizing state-feedback.

```
* outputfunc *
output functions lambda for full (vector) relative degree
[ x[3], x[4] - x[2] - x[1] ]
* statelin *
the exact linearizing feedback: (u)
                                                          3
[ v[1] - x[2] exp(x[3]) - 2 x[3] x[2] - 2 x[3] ,
- x[3]v[1] - v[1]exp(x[3]) - v[2] + 3x[3]x[2]exp(x[3])
             2          4
+ 2 x[2] x[3]  + 2 x[3]
                3                          2
+ 2 exp(x[3]) x[3]  + x[2] exp(x[3]) - x[2]  exp(x[3])
                2
- x[2] exp(x[3]) x[3]  - x[1] - x[2] + x[4] - x[1] x[4] ]
```

This cannot be verified against published results, because this model is used in [9] for other purposes. Nevertheless, due to internal verification in the NONSCON package, the result is correct.

Example 6.3. The model of the system is

$$\dot x = \begin{bmatrix} x_2 - x_3^2 \\ x_3 + 2 x_1^2 x_3 \\ x_1^2 \\ x_1 + x_3^2 \end{bmatrix} + \begin{bmatrix} 0 \\ 2x_3 \\ 1 \\ 0 \end{bmatrix} u .$$

For this model we cannot find a function λ that gives the model a full order relative degree, because the conditions of Theorem 4.1 are violated. According to [9] the function $\lambda(x) = x_1$ gives the model a relative degree of 3, the largest that is possible. The function `outputfunc` that is normally used to compute functions that give the model a full order relative degree is also able, but only for SISO systems, to compute functions that will give a maximal relative degree. The following log of a MAPLE session shows this.

```
* outputfunc *
the output function gives relative degree of [ 3 ]
output lambda for maximal relative degree [ x[1] ]
```

This result is in agreement with the result mentioned above. The normal form of this model with $\lambda = x_1$ as output y is given by [9, p. 212] as

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= z_3 \\ \dot{z}_3 &= v \\ \dot{\eta} &= z_1 + z_3^2. \end{aligned}$$

This is computed with the NONLINCON function `normform`.

```
• normform •
the normal form: (fn, gn)
                                  2          2
[ zeta[2], zeta[3], zeta[1] , zeta[3]  + zeta[1] ]
                    [ 0 ]
                    [ 0 ]
                    [ 1 ]
                    [ 0 ]
the zero dynamics: (zerodyn)   [eta[1]dot = 0]
the zeroing input: (uzero)     [ 0 ]
```

This result (of the form $\dot{z} = f_n(z) + g_n(z)v$) does not agree with the result given above. The difference is in the equation for $\dot{z}_3$ that should be $\dot{z}_3 = z_1^2 + v$. The term z_1^2 has been inadvertently dropped in [9].

Example 6.4. The model of the system is

$$\dot{x} = \begin{bmatrix} x_2 \\ x_4 \\ x_1 x_4 \\ x_5 \\ x_3 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ x_3 & x_2 \\ 0 & 1 \\ x_5 & x_2 \\ 1 & 1 \end{bmatrix} u, \qquad y = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

This model has no well defined relative degree at $x^o = 0$, because

$$A = \begin{bmatrix} 1 & 0 \\ x_3 & x_2 \end{bmatrix}$$

is singular for $x_2 = 0$. The zero dynamics is given in [9] by $\dot{x}_3 = -x_3$ (in the original coordinates) and the zeroing input by

$$u^*(x) = \begin{bmatrix} 0 \\ -x_3 \end{bmatrix}.$$

The next edited log of a MAPLE session shows that the zero dynamics can be computed by NONLINCON. From the functions supplied by NONLINCON we only need `extnormform` that implements the Zero Dynamics Algorithm.

```
• extnormform •
1st step, output zeroing submanifold:
 [x[2] = 0, x[1]= 0]
1st step, constraints on the neighborhood: none
2nd step, output zeroing submanifold:
 [x[2] = 0, x[1] = 0, x[4] = 0]
2nd step, constraints on the neighborhood: none
3rd step, output zeroing submanifold:
 [x[2] = 0, x[1] = 0, x[4] = 0, x[5] = 0]
3rd step, constraints on the neighborhood: none
3rd step, matrix L_gH has full rank m
the zero dynamics: (zerodyn)
{x[1]dot=0,x[2]dot=0,x[3]dot=-x[3],x[4]dot=0,x[5]dot=0}
the zeroing input: (uzero)   [ 0, - x[3] ]
```

Only the non trivial equations of the zero dynamics are of importance. The results coincide with the ones given above.

Example 6.5. The model of the system is

$$\dot{x} = \begin{bmatrix} x_1^2 + x_2 \\ x_1 x_3 \\ -x_1 + x_3 \\ 0 \\ x_5 + x_3^2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ x_2 & 0 \end{bmatrix} u, \qquad y = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix}.$$

This model has no well defined relative degree, because

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

is singular for all x. According to [9] the necessary feedback for input-output linearization is $u = \alpha(x) + \beta(x)v$ with

$$\alpha = -\begin{bmatrix} -x_1 + x_3 \\ 2x_1^3 + 2x_1 x_2 + x_1 x_3 \end{bmatrix}, \qquad \beta = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The following log of a MAPLE session, with some edits, shows that NONLINCON can compute the input-output linearizing state-feedback. From the functions supplied by NONLINCON we only need `inoutlin` that implements the Structure Algorithm.

```
• inoutlin •
the exact linearizing feedback and linearized
system dynamics: (u, fl, gl)
                                                    3
[ v[1]+x[1]-x[3], v[2]-2x[1]x[2]-2x[1] -x[1]x[3] ]
          2                  3
[ x[2]+x[1] , -2x[1]x[2]-2x[1] , 0, x[1]-x[3],
          2
x[5]+x[3] +x[1]x[2]-x[2]x[3] ]
                    [   0   0 ]
                    [   0   1 ]
                    [   1   0 ]
                    [   1   0 ]
                    [ x[2]  0 ]
the explicit feedback: (alpha, beta)
                                                    3
[ x[1] - x[3], - 2 x[1] x[2] - 2 x[1]  - x[1] x[3] ]
                    [ 1  0 ]
                    [ 0  1 ]
```

The computed state-feedback agrees with the results given above.

7. Vehicle simulation problem

This larger scale problem is derived from an inverse simulation problem in multi-body dynamics. In this problem a vehicle is required to perform a standardized manoeuvre. To simulate the manoeuvre the required inputs to the system must be known. Normally, the manoeuvre is such that a suitable selected output of the system can be set to 0. In most cases these outputs do not fully determine the behavior of the system and the remaining freedom represents exactly the zero-dynamics. If this dynamics is stable, then the simulation is also stable; if it is unstable, then additional measures are needed to perform a stable simulation.

As an example we use a simple two dimensional model of a vehicle, a frame moving in the horizontal plane, with traction and tire forces acting on the frame end-points. It looks like the model of a bike, because it only considers the center line of a motor vehicle, see Fig. 3.

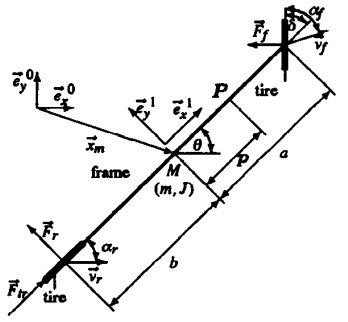

Figure 3: Model of a frame moving in the horizontal plane with forces acting on the frame end-points

The required manoeuvre is steady state turning: the longitudinal speed of the center of mass M of the vehicle is constant, and a point P on the axis of the frame should describe a circle of specified radius. The specific problem is for which distances p, from P to M, the zero-dynamics is stable or unstable.

The equations of motion of the model are

$$\begin{aligned} m\ddot{x}_m &= -F_f \sin(\delta + \theta) - F_r \sin\theta + F_{tr} \cos\theta \\ m\ddot{y}_m &= +F_f \cos(\delta + \theta) + F_r \cos\theta + F_{tr} \sin\theta \\ J\ddot{\theta} &= aF_f \cos\delta - bF_r \end{aligned}$$

with inputs F_{tr} (traction force) and δ (steering angle). The three degrees-of-freedom x_m, y_m, and θ, are respectively the coordinates of M and the orientation of the frame with respect to a fixed reference frame $(\vec{e}_x^{\,0}, \vec{e}_y^{\,0})$, indicated by the superscript ^{0}. The steering angle δ and the drift angles α_f and α_r are given with respect to a body fixed reference frame. The frame has mass m and moment of inertia J with respect to M. The forces acting on the frame are the traction force F_{tr} and the lateral tire forces F_f, F_r. The last two forces can be expressed in the drift angle and the normal tire force F_n by the so called *Magic formula*

$$\begin{aligned} F(F_n, \alpha) &= D(F_n)\sin(C \arctan(Bv)) \qquad (7) \\ v &= (1 - E)\xi + (E/B)\arctan(B\xi), \quad \xi = \alpha + Sh \end{aligned}$$

The dependency holds for both front and rear. The parameters in these formulae have to be fitted to experimental data. The drift angles can be expressed in the degrees-of-freedom and steering angle by

$$\alpha_f = \delta - \arctan(v_{fy}^1/v_{fx}^1), \qquad \alpha_r = -\arctan(v_{ry}^1/v_{rx}^1)$$

with

$$\vec{v}_f^{\,1} = R^T(\dot{\vec{x}}_m^{\,0} + \dot{R}\vec{s}_f^{\,1}), \qquad \vec{v}_r^{\,1} = R^T(\dot{\vec{x}}_m^{\,0} + \dot{R}\vec{s}_r^{\,1})$$

and

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \quad \dot{\vec{x}}_m^{\,0} = \begin{bmatrix} \dot{x}_m \\ \dot{y}_m \end{bmatrix}, \quad \vec{s}_f^{\,1} = \begin{bmatrix} a \\ 0 \end{bmatrix}, \quad \vec{s}_r^{\,1} = \begin{bmatrix} -b \\ 0 \end{bmatrix}$$

where the superscript ^{1} indicates the coordinates with respect to the body fixed reference frame $(\vec{e}_x^{\,1}, \vec{e}_y^{\,1})$. Especially the functional dependency of the lateral tire forces on the degrees-of-freedom and the steering angle make this system quite nonlinear and difficult to analyze.

By choosing the outputs as

$$y_1 = \dot{x}_m \cos\theta + \dot{y}_m \sin\theta - V_d$$
$$y_2 = (x_m + p\cos\theta)^2 + (y_m + p\sin\theta)^2 - R_d^2,$$

with R_d the desired radius of the circle and V_d the desired longitudinal speed, the required manoeuvre corresponds with a zero output. The output y_2 depends on the distance p from $\mathcal{P}$ to M. To solve the problem, first compute the zero dynamics and then determine its stability as a function of p.

The first problem encountered in computing the zero dynamics is the non affine character of the equations with respect to the steering angle δ. To overcome this problem an additional integrator can be added to the system and $\dot{\delta}$ can be regarded as a new input. However, this causes the system to have a singular decoupling matrix A, which makes the analysis more difficult. The `extnormform` function, based on the Zero Dynamics Algorithm, can be used but this leads to insurmountable problems in the computation due to huge memory needs. Another solution is to add an integrator to the other input F_{tr} also, and using $\dot{F}_{tr}$ as a new input, making the decoupling matrix regular by also "delaying" the input F_{tr}. All this adding of integrators must be performed by hand through modifications of f and g, because the *Dynamic Extension Algorithm* is not implemented.

The addition of two integrators permits the use of the standard transformation to the normal form with the function `normform`, although for a larger system. Due to the larger number of states again problems arised with the memory needed to perform the computation. To simplify the problem, another "saturating" function was used in the magic formula (7) by using $\sqrt{\cdot}$ instead of $\sin(\arctan(\cdot))$. Now, the transformation, and even a solution for the partial differential equation $L_f\phi(x) = 0$, can be computed, but the inverse transformation cannot, because of an artificial limit in the size of objects permitted by the software, putting an untimely end to the computation.

Another attempt to solve the problem of non affine input is not to enhance the model, but to simplify it so the input δ appears linear in the equations. To this end, the assumption that δ is small has to be adopted. Then, by using a Taylor series expansion, the system of equations can be written linear in δ if the dependency of δ of the front lateral tire force is dropped, so δ should also be small relative to α_f, an unrealistic assumption. This time the transformation to the normal form can be computed, but the nonlinear system, needed to compute the inverse transformation, could not be solved, this time not due to memory constraints but due to limitations of the `solve` function of MAPLE.

Concluding: several attempts to solve this problem were in vain due to

- limits on the maximal object size
- limitations of the `solve` function
- limits of the available physical (or virtual) memory.

The last problem is relatively easy to solve, but because the memory requirements are likely to be exponential or double exponential with respect to the problem size, this is not a cheap way out. The first two problems should be resolved by the MAPLE developers.

All in all, the symbolic computation software yet is not mature enough to solve larger scale problems, and in this case we have to resort to a purely numeric approach to determine the stability of the zero dynamics.

We propose to use this problem as a benchmark, or yardstick, for the viability of symbolic computation programs.

8. Conclusion and Discussion

The computation of the normal form, of the zero dynamics, of the solution of the input-output and the state space exact linearization problems can be automated by using symbolic computation, *e.g.*, by using the NONCON package. This means that it is much easier now to use controllers based on the linearization approach, that can fully take into account the nonlinearities 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, due to the restricted capability to solve sets of nonlinear equations and because the possibilities to solve sets of nonlinear differential equations are limited. Therefore, the designers of control systems cannot yet routinely compute solutions for these problems, using tools that are based on symbolic computation programs.

To remedy this, we recommend to extend the capabilities of symbolic computation programs for solving large and intricate sets of nonlinear equations and for solving sets of differential equations. Another possibility is to use another algorithm, *e.g.*, for the state space exact linearization problem, that completely avoids the integration step.

Future research will have to aim at

- devising new 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

To come back to the title of the paper, at the moment we cannot deny for sure the viability of this approach without being unjust to the developers of symbolic computation programs and without being shut of of a promising approach that can complement a combined analytical (paper and pencil) and numerics approach. Both attitudes to symbolic computation, *i.e.*, to refrain and to embrace are (not) viable, depending on the expectations of the near future, and the viability question is still open. It is hoped for, and more or less expected, that within this decade the efficiency and capacity of general symbolic computation programs are enhanced, and that the capabilities of relatively cheap computers will enable the solution of the benchmark and larger problems at reasonable costs.

Acknowledgement

Harm van Essen implemented the analysis and design algorithms in MAPLE and is the main author of NONCON. Ted van de Broek supplied the vehicle simulation problem and corresponding model equations. The author is indebted for both contributions.

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*, vol. 3, (Seoul, Korea), pp. 1102–1106, IEEE, Aug. 1987.

[2] 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), pp. 131–136, IFAC, Oxford: Pergamon Press, 1989.

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

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

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

[6] 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.* (J. Nieuwenhuis, C. Praagman, and H. Trentelman, eds.), vol. 4, (Groningen, The Netherlands), pp. 2081–2085, June 1993.

[7] 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* (S. M. Watt, ed.), (Bonn, Germany), pp. 321–322, ACM Press, New York, July 1991.

[8] B. de Jager, "Symbolics for control: Maple used in solving the exact linearization problem," in *Studies in Computer Algebra For Industry, Proceedings of the 1992 SCAFI Seminar*, vol. II of *The SCAFI Papers*, (Amsterdam, The Netherlands), pp. 125–134, CAN, 1992.

[9] A. Isidori, *Nonlinear Control Systems: An Introduction*. Berlin: Springer-Verlag, 2nd ed., 1989.

[10] 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.

[11] H. Nijmeijer and A. J. van der Schaft, *Nonlinear Dynamical Control Systems*. New York: Springer-Verlag, 1990.