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

Citation for published version (APA):

DOI:
10.1109/CDC.1993.325146

Document status and date:
Published: 01/01/1993

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
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.tue.nl  Fax: +31 40 447355

Abstract

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

To assess the viability of this tool, several algorithms available for analysis and design are implemented in NONCON, and applied to examples of textbook problems as well as 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 NONCON package can be successfully used. The larger-scale problem is, however, too complex to be solved with the current versions of NONCON and MAPLE. Some recommendations to improve the implementation and to make MAPLE more useful for use with NONCON 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. Blankenhip [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., COSYPAK, 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 these capabilities are not fully exploited yet. 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 NONCON (a successor of the ZEROCON 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 focuses 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 NONCON package is described. Section 6 presents some textbook examples, using the NONCON package, for each of the problem areas. The next section contains a discussion of the larger scale problem. Section 7 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 inferred 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-observable 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 of these problems, modifications of the system to be controlled are necessary. The zero dynamics are 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 (transformed) state 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 Iwai [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.

$$x = f(x) + g(x)u, \quad y = h(x)$$

(1)
with state vector $x \in \mathbb{R}^n$, containing all necessary information of the plant, input vector $u \in \mathbb{R}^m$, and output vector $y \in \mathbb{R}^p$. 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 $x = x^0$ if

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

$$A(x^0) = \begin{bmatrix}
L_{x^0}^{r_i - 1} h_1(x) & \cdots & L_{x^0}^{r_i - 1} h_m(x) \\
L_{x^0}^{r_i - 2} h_1(x) & \cdots & L_{x^0}^{r_i - 2} h_m(x) \\
\vdots & \ddots & \vdots \\
L_{x^0}^{r_i - 1} h_1(x) & \cdots & L_{x^0}^{r_i - 1} h_m(x)
\end{bmatrix}.$$  

Here $L^k h_i(x)$ means the $k^{th}$ successive Lie derivative of the scalar function $h_i(x)$ in the direction of the vector field $g_i$, e.g., $L^k h_i(x) = \partial^k h_i(x)/\partial x^j$. The matrix $A(x^0)$ 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

$$u = a(x) + b(x)v$$  

where the components $a_i$ and $b_i$ are smooth functions.

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

$$z = (\xi, \eta) = \Phi(x).$$

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

$$y_1 = h_1(x) = \xi_1$$
$$y_2 = h_2(x)$$
$$\ldots$$
$$y_i = h_i(x)$$
$$\xi_i = b_i(\xi, \eta) + \sum_{j=1}^{i-1} a_{ij}(\xi, \eta) y_j$$
$$\eta_i = q_i(\xi, \eta)$$

for $i = 1, \ldots, m$.

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))

$$\xi^{(0)}(t) = b_1(\xi, \eta) + A(\xi, \eta) u$$
$$\eta(t) = q(\xi, \eta),$$

where $\xi^{(0)}$ are the $m$ elements of $\xi$ on the places $r_1, r_2, \ldots, r_r$ with $r = \sum_{i=1}^m r_i$. Because $A$ is nonsingular if the relative degree is well defined, the control

$$u = A^{-1}(\nu - \eta)$$

with the new input $\nu$ is properly defined and linearizes the part of system (4) that is visible at the output of the system $\xi^{(0)} = \nu$.

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

$$\eta = q(0, \eta), \quad \eta(0) = \eta^0.$$

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 linear systems are not satisfied, 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 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 zero for all $i$, by a proper choice of initial state $x(0)$ and control input $u(t)$. Here we have to employ an appropriate state feedback controllers. 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 $x^0$ and a static-state feedback such that for all $k \geq 0$ and all $l = 1, \ldots, m$ the expression

$$L^k \phi(x) = \phi_{x^k}(x)$$

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 $a_i$ and $b_i$ 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 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 under which the feedback (2) 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, $r = m$.

When a full order relative degree cannot be obtained, it is sometimes 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 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 $\xi$, that do not belong to the zero dynamics, can be set to zero.

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 $M_1 = M_t$ and $M_0 = M^*(0)$, i.e., the first submanifold is the inverse image of the point $y^0 = 0$. When some conditions are fulfilled this sequence converges to the locally maximal output zeroing submanifold $Z^*$ in some neighborhood of $x^0$ and there exists a mapping $u^*$ such that $f(x) = x + g(x) + c(x)^T u^*$ is tangent to $Z^*$. The pair $(Z^*, f_0)$ is called the zero dynamics of the system. When the mapping $H(x)$ is defined in a neighborhood $U$ of $x^0$ by $Z^* \cap U = \{ x \in U : H(x) = 0 \}$ the input $u^*$ can be computed as the solution of $L_{x^0}^k H(x) + L_{x^0}^k c(x) = 0$. For further details of this algorithm, and the conditions to be imposed to guarantee the convergence of the sequence $Z^*$, we refer to [19].
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 (3). 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 Toplitz matrices

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

where

$$T_k(x) = \begin{bmatrix}
L_{x_k} f_{i}(x) & \cdots & L_{x_k} f_{i}(x) \\
\vdots & \ddots & \vdots \\
L_{x_k} f_{m}(x) & \cdots & L_{x_k} f_{m}(x)
\end{bmatrix}$$

for \(0 \leq k \leq 2n - 1\).

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

$$\Gamma(x) = \begin{bmatrix}
\gamma_0(x) \\
\vdots \\
\gamma_k(x)
\end{bmatrix}$$

we can solve

$$L_\alpha \Gamma(x)(u(x)) = -L_\beta \Gamma(x)$$

for \(\alpha\) and \(\beta\) in a neighborhood of \(x^0\). 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

$$G_i = \text{span} \{a f_{x_k} g_i ; 0 \leq k \leq i, 1 \leq j \leq m\}$$

for \(0 \leq i \leq n - 1\).

To define the adjoint \(a f_{x_k} g_i\) we need the Lie product

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

and \(a f_{x_k} g_i\) is defined recursively as \(a f_{x_k} g_i = [a f_{x_k} g_i, a f_{x_k} g_i] \cdots [a f_{x_k} g_i, a f_{x_k} g_i] g_i\).

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

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

with rank \(g(x)\) is finite. There exists a solution for the state space exact linearization problem if and only if

1. \(G_i\) has constant dimension near \(x^0\) 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.

When the conditions for the exact linearization problem are fulfilled, the static 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

$$L_{x_k} \lambda_i(x) = 0, \quad 0 \leq k \leq n - 2 \text{ and } 0 \leq j \leq m.$$  \hspace{1cm} (6)

Also \(\sum_{i=1}^{m} \lambda_i(x) = e\), where the set of integers \(\{1, \ldots, m\}\) is the relative degree vector. The \(m\) functions \(\lambda_i\) can be computed, based on a constructive proof of Theorem 4.1.

Using the functions \(\lambda_i\), the change of coordinates and state-feedback \(u = \xi(x) + \beta(x)\) 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 \(\lambda_i\), 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 NOBECON. 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 NOBECON](image)

Because the NOBECON package is simple 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 \(\lambda_i\), 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.

\(278\)
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 spent 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 compensation of the function \( f \) is often unsuccessful, especially for more complicated systems, as NonCON cannot finish the computations. This part of NonCON 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, workarounds are applied. We discuss some of these workarounds. The problem was that some of the standard Maple\textsuperscript{TM} 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, Gaussian 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 extract, output, gausselim, and gaussjord are extended in a rather ad hoc way, and this part of NonCON should be considered as experimental also.

6. Textbook problems

To illustrate the use of NonCON 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 one output. 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 example the zero dynamics is computed with the State Feedback 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 derived ones.

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

\[
2T = m_1 (\dot{\theta}_1 + \dot{\theta}_2)^2 + m_2 (\dot{\theta}_1 + \dot{\theta}_2 + \theta_1 + \theta_2)^2 + 2m_1 \dot{\theta}_1 (\dot{\theta}_1 + \dot{\theta}_2) \cos \theta_1 \]

The inputs \( u_1 \) and \( u_2 \) are the joint torques and the outputs \( \dot{\theta}_1 \) and \( \dot{\theta}_2 \) the joint positions. The state \( x \) of the model corresponds with the degrees-of-freedom and their derivatives in the following way \( x = [\theta_1 \ \theta_2 \ \dot{\theta}_1 \ \dot{\theta}_2]^T \).

![Figure 2: Robot with two revolute joints](image)

The aim is to derive the dynamics of the model when it is constrained. There are two cases, the first one with \( y = \dot{\theta}_1 \) and \( u = 0 \), the second case with \( y = \dot{\theta}_2 \) and \( u = 0 \). According to [11] the zero dynamics are given by (with all model parameters set to 1)

\[
\dot{x}_2 = x_2 \quad x_4 = g \sin x_3 \quad \text{resp.} \quad x_1 = \dot{x}_3 = 0 \quad \text{with} \quad g = 9.81 \text{ m/s}^2.
\]

The following log of a NonCON 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 \( y = \dot{\theta}_1 \),

\[
\text{normform}
\]

zero dynamics is

\[
[\text{eta}[1] = \text{eta}[2], \text{eta}[2] = 0]
\]

output nulling input is

\[
[\text{eta}[1] = \text{eta}[2] = 0]
\]

Example 6.2. The model of the system is

\[
x = [\begin{array}{c} x_1 + x_2 x_4 \\ x_2 + x_3 \end{array}] + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u, \quad y = [x_1, x_2].
\]

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.

\[\text{reldeg} = [1, 1] \]

\[\text{total relative degree} = 2 \]

\[\text{matrix Ad} = [x_3, 1] \]

\[\text{bdeg} = [x_1 + x_4, x_2 x_3] \]

To permit a linearizing state-feedback we have to find functions \( \lambda \) for which the model has a full order relative degree. The next log of a NonCON session shows that this is possible. The function outputfunct is used to compute the \( \lambda \)'s and the function statelin to compute the linearizing state-feedback.

\[\text{outputfunct} \]

\[\text{output functions lambda for full (vector) relative degree } [x_1, x_2, -x_2 - x_1] \]

\[\text{statelin} \]

\[\text{the exact linearizing feedback: } u \]

\[\text{normform} \]

\[\text{zero dynamics is } [\text{eta}[1] = \text{eta}[2] = 0, \text{eta}[3] = 3/5 \text{ grav sin(eta[1])}] \]

\[\text{output nulling input is } [1/5 \text{ grav sin(eta[1])}] \]

Example 6.3. The model of the system is

\[
x = [\begin{array}{c} x_2 \\ x_3 + 2 x_4 \end{array}] + \begin{bmatrix} 0 & 2 \\ 2 & 1 \end{bmatrix} u.
\]

For this model we cannot find a function \( \lambda \) that gives the model a full order relative degree, because the conditions of Theorem 4.1 are violated. According to [9] the function \( A(0) = x_2 \) gives the model a relative degree of 3, the largest that is possible. The function outputfunct 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.

\[\text{outputfunct} \]

\[\text{the output function gives relative degree of } [3] \]

\[\text{output lambda for maximal relative degree } [x_1] \]
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

$$
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= x_3 \\
\dot{x}_3 &= v \\
\dot{\eta} &= x_1^2 + x_2^2.
\end{align*}
$$

This is computed with the Maple function `normform`.

The zero dynamics: (zerodyn) [eta[]=0]

$$
\begin{align*}
\dot{x}[\text{zero}] &= \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}, \\
0 &\equiv 1 & 0 & u, \\
\dot{y} &= \begin{bmatrix} x_2 \\ x_3 \\ x_4 \end{bmatrix}.
\end{align*}
$$

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

$$
A = \begin{bmatrix} 1 & 1 \\ -3 & 2 \end{bmatrix}
$$

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

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

The next edit of a Maple session shows that the zero dynamics can be computed by NQSOCON. From the functions supplied by NQSOCON we only need `normform` that implements the Zero Dynamics Algorithm.

Example 6.4. The model of the system is

$$
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= x_3 + x_4 \\
\dot{x}_3 &= 0 \\
\dot{x}_4 &= 1 \\
u &= y = x_2.
\end{align*}
$$

This model has no well defined relative degree, because

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

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

$$
\begin{align*}
\alpha &= \begin{bmatrix} -x_1 \\ x_2 + x_3 \end{bmatrix}, \\
\beta &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}.
\end{align*}
$$

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

**Example 6.5.** The model of the system is

$$
\begin{align*}
\dot{x}_1 &= x_2 + x_3 \\
\dot{x}_2 &= -x_1 + x_3 \\
\dot{x}_3 &= x_2 \\
\dot{x}_4 &= 0 \\
u &= y = x_2.
\end{align*}
$$

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](image)

The required manoeuvre must be 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

$$
\begin{align*}
\dot{x}_2 &= -F_t \sin(\delta) - F_t \sin(\delta) + F_t \cos(\delta) \\
\dot{x}_3 &= -F_t \cos(\delta) + F_t \cos(\delta) \cos(\delta) + F_t \sin(\delta) \\
\dot{\theta} &= F_t \cos(\delta) - bF_t,
\end{align*}
$$

with inputs $F_t$ (tire force) and $\delta$ (steering angle). The three degrees-of-freedom $x_2, x_3$, and $\theta$, respectively the coordinates of $M$ and the orientation of the frame with respect to a fixed reference frame $\mathcal{G}, P$, indicated by the superscript 0. The steering angle $\delta$ and the drift angles $\alpha$ and $\alpha_r$ 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 $F_t$ and the lateral tire force $F_l$. The last two forces can be expressed in the drift frame $\mathcal{G}$ and the normal tire force $F_n$ by the so-called Magic formula

$$
F_n(F_\alpha, \alpha) = D(F_\alpha) \sin(\theta) \tan(\theta) \tan(\beta) v = (1 - E) \xi + (E(t) \tan(\beta), \xi = \alpha + \delta
$$

$$\text{(7)}$$

280
The dependency holds for both front and rear. The parameters in these formulas have to be fitted to experimental data. The drift angles can be expressed in the degrees-of-freedom and steering angle by
\[ \alpha_f = \arctan\left(\tan\theta_1/\rho_{d1}\right), \quad \alpha_r = -\arctan\left(\tan\theta_2/\rho_{d2}\right) \]
with
\[ y_f' = R^2 (\delta_u + \eta_{d2}'), \quad y_r' = R^2 (\delta_u + \eta_{d2}'), \]
and
\[ R = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \end{bmatrix}, \quad \eta = \begin{bmatrix} x_n \\ y_n \\ y_r \end{bmatrix}, \quad \eta_f = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \]
where the superscript \(^1\) indicates the coordinates with respect to the body fixed reference frame \((\xi'_1, \xi'_2)\). 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
\[ y_1 = x_n \cos \theta + y_n \sin \theta - V_d \]
\[ y_2 = (x_n + \rho \cos \theta') + (y_n + \rho \sin \theta') - R^2 \]
with \( R \) the desired radius of the circle and \( V_d \) the desired longitudinal speed, the required maneuver corresponds to a zero output. The output \( y_2 \) depends on the distance \( p \) from \( P \) to \( M \). To solve this 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 \( \delta \). To overcome this problem an additional integrator can be added to the system and \( \delta \) 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 dynamic extension form, based on the Zero Dynamics Algorithm, can be used but this leads to unsolvable problems in the computation due to huge memory needs. Another solution is to add an integrator to the other input \( P_x \) also, and using \( P_x \) as a new input, making the decoupling matrix regular by also “delaying” the input \( P_x \). 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 \( \text{normform} \), although for a larger system. Due to the larger number of states again problems arise with the memory needed to perform the computation. To simplify the problem, another “saturating” function was used in the magic formula (7) by using \( \sqrt{\sin(\arctan(\cdot))} \). Now, the transformation, and even a solution for the partial differential equation \( \partial_x \psi(x) = 0 \), can be computed, but the inverse transformation cannot, because of an artificial limit in the size of objects permitted by the software, putting an unrealistic 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 that the input \( \delta \) appears linear in the equations. To this end, the assumption that \( \delta \) is small has to be adopted. Then, by using a Taylor series expansion, the system of equations can be written linear in \( \delta \) if the dependency of \( \delta \) of the front lateral tire force is dropped, so \( \delta \) should also be small relative to \( \alpha \), 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 \text{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 \text{solve} function
- limits of the available physical (or virtual) memory.

The last problem is relatively easy to solve, but because the memory required limits 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 large 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 \text{NooCON} 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
- deviating 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 promising 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 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 next 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 \text{NooCON}. Ted van de Broek supplied the vehicle simulation problem and corresponding model equations. The author is indebted for both contributions.

References


