# Symbolic computation in nonlinear control system modeling and analysis

TuA1-3 10:40

Proceedings of the 1999 IEEE
International Symposium on Computer Aided Control System Design
Kohala Coast-Island of Hawai'i, Hawai'i, USA • August 22-27, 1999

# Symbolic Computation in Nonlinear Control System Modeling and Analysis

Bram de Jager
Faculty of Mechanical Engineering, WH 2.137
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
Email: A.G.de.Jager@wfw.wtb.tue.nl Fax: +31 40 2461418

**Abstract**

The paper discusses the use of symbolic computation for model formulation, model integration, model checking, and model analysis. The zero dynamics plays an important role in the areas of modeling, analysis, and control of linear and nonlinear systems. The zero dynamics gives additional insight in the structure of the model employed and is an aid in modifying a model to satisfy some needs of the modeler. For nonlinear systems the analytical calculations to get the zero dynamics by paper and pencil may be quite involved. Symbolic computation has been used to overcome this difficulty. For a reasonable class of systems the computation can be performed without human aid or intervention, making the zero dynamics procedure a feasible and valuable addition to the toolbox of the modeler, analyst, or control system designer. For system models that are more than moderately complex symbolic computation cannot be fully enjoyed due to the complexity of parts of the algorithms that is (double) exponential in some measure of the problem size, or due to expression swell that cannot be easily eliminated. This implies that symbolic computation will not replace other tools, like those based on numerics, but will complement them.

**Keywords**: Symbolic computation, differential algebraic equations, zero dynamics, nonlinear systems, computer aided control system design.

## 1. Introduction

Symbolic computation, also known as computer algebra, is a powerful approach in solving tough and intricate problems in applied mathematics. With the advent and improvement of commercially available symbolic computation packages this approach is becoming more generally available and used. It promises drastic changes in the way problems, in fields as diverse as one can imagine, are tackled.

In nonlinear control system analysis and design advanced tools are necessary to be able to meet the ever increasing requirements on the speed and quality of the control system design cycle. Those tools serve at least two goals. First, by incorporating new and powerful analysis and design algorithms they permit more severe specifications to be met by the closed loop system. Second, by providing fast, extensive, integrated, and easy to use facilities, they allow a streamlined design cycle and a shorter time-to-market.

Many analysis and design methods are provided in the literature. To facilitate the analysis of nonlinear systems and the design of nonlinear controllers, computer based design tools are used. The use of numeric computation in the computer aided design of nonlinear control systems is well known for some time. Therefore, it will not be introduced further. Methods not solely based on numerics require analytical calculations, normally done by hand. The complexity of the methods combined with the increasingly complex models discourage the use of hand computations. To assist in this respect, symbolic computation programs may provide an alternative.

Tools that implement methods using computer algebra are rarely commercially available, unlike the situation for numerical computations. It is not even well known if computer algebra programs are well suited for this task in the control system design cycle. Their capabilities are limited in at least two respects. First, one is not sure if all interesting methods are amenable for implementation. Perhaps they require computations that are too hard to be implemented. Second, the complexity of the problems to be solved may be so overwhelming that the computation is brought down by too heavily taxing the computer, with respect to both time and memory requirements. Most algorithms are expected to be not only $\mathcal{NP}$-hard, but to show at least a growth rate double exponential with the problem size or with other measures of complexity.

A set of symbolic computation functions, together called the NONCON package, see [1,2] for more details, was developed and employed to solve problems in

0-7803-5500-8/99 $10.00 © 1999 IEEE

the modeling, analysis, and design of nonlinear control system. The NONLINCON package, or rather its use, could answer the question if computer algebra programs deserve a place in the toolbox for the control system designer.

The functions available in the package range from the computation of the relative degree, the normal form, or the zero dynamics of the model, that does not need to have a well-defined relative degree, to computing exact linearizing or input-output linearizing feedbacks. It implements among others the dynamic extension algorithm, the zero dynamics algorithm, the structure algorithm, and the controlled invariant distribution algorithm [3]. The structure of NONLINCON is illustrated in Fig. 1. Some of the blocks in this scheme will be discussed in more detail later.

This paper dedicates its attention to the use of symbolic computation in systematic modeling and model analysis in general, and the use of NONLINCON in these areas in particular.

## 2. Modeling

In systematic modeling of large-scale systems a modular approach has distinct advantages: the modeling task can be divided between teams, code can be reused, *etc.* In control systems a natural division is between the plant and the controller, where both constituent parts can be decomposed in several modules that again can be split up in submodules.

A disadvantage of this type of modeling is the connection or interaction between (sub)modules that will often lead to algebraic relations, or even to algebraic loops. Control system design often requires dynamic models, *e.g.*, given by (partial) differential equations that naturally follow from basic conservation laws. In general, due to the conservation laws and algebraic relations, one will end up with a set of differential and algebraic equations (DAE). It is not always easy, not always desirable, or sometimes not even possible, to eliminate the algebraic relations to obtain a set of differential equations only. A measure of difficulty for numerical simulation for DAE models is the so-called index. For index 0 or 1 numerical simulation is relatively straightforward, for higher index DAE models numerical computation becomes unreliable or infeasible.

Symbolic computation can be used to obtain models that can be simulated easily. An example is in the index reduction problem, where one manipulates the DAE model to reduce its index and make it easier for numerical simulation. A way to achieve this is by symbolically computing the zero dynamics, either by a normal form computation or by the zero dynamics algorithm, and to use this dynamics in generating an equivalent set of DAE of lower index. The reasoning is as follows.

The zero dynamics represent the dynamics of a system that is compatible with the requirement that outputs of the system are kept zero, by a suitable control input and initial conditions. For this application area, the output is formed by equating it to one or more of the algebraic equations, written in such a form that they are satisfied when a "slack" variable is zero, the output is then defined as this variable. Computing the zero dynamics then yields the dynamics of the differential equations that is compatible with the algebraic constraints. This approach can be taken for DAE in semi-explicit form, where the algebraic relations are separated out.

The NONLINCON packages provides other facilities for symbolic analysis of a model, besides the zero dynamics computation. The model, in general a set of nonlinear differential and algebraic equations in the state, input, and output of the system, need not necessarily be affine in the input. The functions range from computing coordinate changes, requiring a symbolic solution to a set of partial differential equations, to computing invariant manifolds. All functions are based on constructive algorithms and are implemented in MAPLE.

The NONLINCON package successfully handles low order models that do not contain large intricate expressions. For large order models the package is less successful, due to expression swell caused by, *e.g.*, the (double) exponential nature of some underlying algorithms, and to inherent limitations of symbolic computation, *e.g.*, due to the lack of suitable algorithms.

## 3. Algorithms

The two standard models used are

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

for affine systems and

$$\dot{x} = f(x,u), \quad y = h(x,u) \tag{2}$$

for nonaffine systems, with $x \in \mathbb{R}^n$ the state, $u \in \mathbb{R}^m$ the input, $y \in \mathbb{R}^p$ the output, $f$ a smooth vector field, $g$ representing a set of smooth vector fields $g_j$ (column-wise), and $h$ is a column with stacked scalar-valued smooth functions $h_i$.

### 3.1. Relative degree

The relative degree of a nonaffine model of a nonlinear system in $(x^\circ, u^\circ)$ is, based on the definition in [4, p. 417], the set of integers $r_i$, $i = 1, \ldots, p$ such that

$$\frac{\partial}{\partial u_j} L_f^k h_i(x,u) = 0 \quad \forall k < r_i, 1 \le i \le p, 1 \le j \le m$$

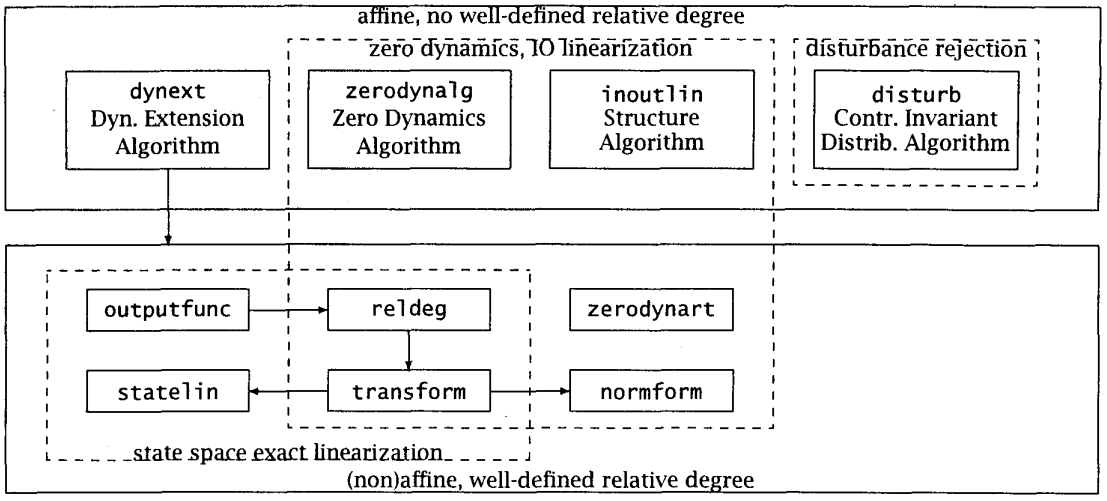


Figure 1: Structure of NONLINCON

and the matrix

$$A_{\text{deg}}(x,u) = \begin{bmatrix} \frac{\partial}{\partial u_1} L_f^{r_1} h_1(x,u) & \dots & \frac{\partial}{\partial u_m} L_f^{r_1} h_1(x,u) \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial u_1} L_f^{r_p} h_p(x,u) & \dots & \frac{\partial}{\partial u_m} L_f^{r_p} h_p(x,u) \end{bmatrix}$$

is nonsingular in $(x^\circ, u^\circ)$, with $L_f h_i(x,u)$ the Lie derivative, *i.e.*, the derivative with respect to $x$ of $h_i(x,u)$ along $f$

$$L_f h_i(x,u) = \frac{\partial h_i}{\partial x} f.$$

Remark that the requirement that $A_{\text{deg}}$ is to be nonsingular is not posed in [4]. The main differences with the definition for affine models is that there the term $\partial L_f / \partial u_j$ is replaced by the derivative along $g_j$, $L_{g_j}$, and the conditions do not depend on $u^\circ$. For affine models both definitions are, however, equivalent. The two definitions are implemented in the function `reldeg`. If no set of integers can be found that fulfills the two conditions, the relative degree is said to be not well-defined.

### 3.2. Normal form

Normal forms have been defined for several classes of models, among them affine models with or without a well-defined relative degree and nonaffine models with a well-defined relative degree, but not (yet) for nonaffine models without a well-defined relative degree.

For affine systems with a well-defined relative degree a normal form is easy to derive by a (local) coordinate transformation

$$z = \begin{bmatrix} \zeta \\ \eta \end{bmatrix} = \Phi(x)$$

based on the model outputs, their derivatives, and the relative degree. The normal form is

$$\begin{aligned} y_i &= \zeta_1^i \\ \dot{\zeta}_1^i &= \zeta_2^i \\ &\vdots \\ \dot{\zeta}_{r_i-1}^i &= \zeta_{r_i}^i \\ \dot{\zeta}_{r_i}^i &= b_i(\zeta,\eta) + \sum_{j=1}^{m} a_{ij}(\zeta,\eta) u_j \end{aligned}$$

for $1 \le i \le m$ and with

$$\zeta = \{\zeta_1^1, \dots, \zeta_{r_1}^1, \zeta_1^2, \dots, \zeta_{r_2}^2, \dots, \zeta_1^m, \dots, \zeta_{r_m}^m\}$$

while for the remaining states, the *internal dynamics*, holds

$$\dot{\eta} = q(\zeta,\eta) + p(\zeta,\eta)u.$$

If the distribution spanned by $g_1(x), \dots, g_m(x)$ is involutive near $x^\circ$, see later, a change of coordinates is possible such that $p(\zeta,\eta) = 0$ and, hence, the internal dynamics has an especially simple form

$$\dot{\eta} = q(\zeta,\eta).$$

Involutive means that the distribution is closed under the Lie or bracket product, $[f,g_i] = L_f g_i - L_{g_i} f$, *i.e.*, the dimension of the distribution $G = \text{span}\{g_1, \dots, g_m\}$ does not change when a vector field, generated by the Lie product of each combination of two of the vector fields in $G$, is included in the set of vector fields generating the distribution.

If this involutivity condition is fulfilled, Frobenius' Theorem states that the differential equations

$$L_{g_j} \phi_i = \frac{\partial \phi_i}{\partial x} g_j = 0$$

do have a full set of solutions. Those solutions can be used as a partial set of new coordinates to obtain $p = 0$.

This computation of the normal form, only suitable for affine models with a well-defined relative degree, was easily adapted to incorporate nonaffine models. It was also extended to affine models without a well-defined relative degree by building the "generalized" normal form.

Computation of the generalized normal form is based on some characteristics of the model, taken from the application of the Zero Dynamics Algorithm (ZDA), see [3, Section 6.1]. The ZDA is an iterative procedure, that, under certain regularity conditions, converges in a finite number of steps.

### 3.3. Zero dynamics

The zero dynamics is 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 $u(t)$.

Given the normal form, the zero dynamics can easily be derived for affine models. If the model is nonaffine a complication may arise because the zeroing input $u_{zero}$ need not be unique.

For models that do not have a well-defined relative degree, that can therefore be put in the generalized normal form only, the zero dynamics is computed directly by the ZDA, which is possible under some conditions. This algorithm can also be used for models with a well-defined relative degree, circumventing the derivation via the normal form, but not for nonaffine models. There the normal form has to be computed for obtaining the zero dynamics.

The ZDA proceeds as follows [3, p. 294]: Define

$$M_k = \{x \in M_{k-1}^c : f(x) \in \mathrm{span}\{g_1(x), \ldots, g_m(x)\} + T_x M_{k-1}^c\},$$

where $M_{k-1}^c$ is the connected component of $M_{k-1} \cap U_{k-1}$ containing the point $x^\circ$ with $h(x^\circ) = 0$, $U_{k-1}$ is a neighborhood of $x^\circ$, $T_x M$ is the tangent space of $M$ at $x$, and the iteration starts with $M_0 = h^{-1}(0)$.

For a certain $k^*$ it holds that $Z^* = M_{k^*}^c$ with $Z^*$ the locally maximal output zeroing submanifold. Characteristic for an output zeroing submanifold $M$ is that - for a suitable choice of feedback $u(x)$ - the trajectories of the closed loop system that start in $M$ stay in $M$ for a certain time, and the corresponding output is zero.

Remark that for affine models with a well-defined relative degree there are three options available to compute the zero dynamics, namely by

1. first computing the normal form, and deriving the zero dynamics from this form,
2. applying the zero dynamics algorithm,
3. using the proposal in [5].

Differences occur, at least for the implementation in NONSCON, in the coordinates used to represent the results.

## 4. Examples

The examples presented use interesting models of mechanical systems, a friction based clutch and a double pendulum. The first example uses the model of a drive line, where an algebraic restriction is introduced when a clutch, that formerly was open, is closed. In general this will reduce the number of degrees-of-freedom of the system by one, so the structure of the dynamics is changed. Given the original model of the system, the zero dynamics for a suitable defined output then represent the model for the system with closed clutch.

The double pendulum example can be formulated as a position constrained multibody model of index 3. Writing the equations of motion for both links of the pendulum separately (as will be done in a systematic and modular modeling approach) the fact that endpoints of the links are connected is equivalent with algebraic relations that equate position, and therefore also speed, of the endpoints of the links. The resulting DAE is of higher index, so difficult to simulate. The zero dynamics of this system, using the derivative of the position constraints to define an output, is computed. It has a reduced index, so is easier to simulate than the original DAE.

### 4.1. Drive line example

A simple drive line model is given by two rotating masses with a clutch in between. When the clutch is open the system has two degrees-of-freedom. If it is closed one degree-of-freedom (DOF) is removed, because both inertias then have the same rotational speed. This effect can be accounted for by computing the zero dynamics of the two DOF system and requiring that an output, in this case the speed difference between the two inertias, is to be kept zero. A simple example of such a system is represented in Fig. 2.

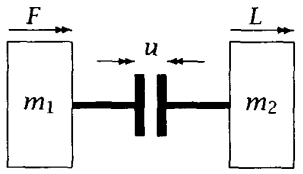


Figure 2: Two rotating masses with clutch

The model employed is

$$\dot{x} = \begin{bmatrix} x_2 \\ F/m_1 \\ x_4 \\ L/m_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m_1 \\ 0 \\ -1/m_2 \end{bmatrix} u = f(x) + g(x)u$$

$$y = x_2 - x_4 = h(x)$$

while $x^\circ = \begin{bmatrix} p_1 & v & p_2 & v \end{bmatrix}^T$ with $x_1$ and $x_2$ the position and speed of mass $m_1$, while $x_3$ and $x_4$ are those of mass $m_2$. The torques $F$ and $L$ represent the driving torque and load, respectively. The input $u$ is the torque generated by the clutch. The positions $p_1$ and $p_2$ may be different while the velocities $x_2$ and $x_4$ are equal to $v$ in $x^\circ$.

For this model the relative degree is $r = 1$. The new coordinates are computed as

$$z = \begin{bmatrix} \phi_1(x) \\ \phi_2(x) \\ \phi_3(x) \\ \phi_4(x) \end{bmatrix} = \begin{bmatrix} x_2 - x_4 \\ x_3 - p_2 \\ x_1 - p_1 \\ \dfrac{m_1 x_2 + m_2 x_4 - v(m_1 + m_2)}{m_2} \end{bmatrix}$$

where the partial differential equations

$$L_g \phi_i = 0, \quad i = 3, 4$$

were successfully solved. The normal form is

$$\begin{bmatrix} \dot{\zeta} \\ \dot{\eta}_1 \\ \dot{\eta}_2 \\ \dot{\eta}_3 \end{bmatrix} = \begin{bmatrix} \dfrac{m_2 F - m_1 L}{m_1 m_2} \\ \dfrac{-m_1 \zeta + m_2 \eta_3 + v(m_1 + m_2)}{m_1 + m_2} \\ \dfrac{m_2(\eta_3 + \zeta) + v(m_1 + m_2)}{m_1 + m_2} \\ \dfrac{F + L}{m_2} \end{bmatrix} + \begin{bmatrix} \dfrac{m_1 + m_2}{m_1 m_2} \\ 0 \\ 0 \\ 0 \end{bmatrix} u.$$

The zero dynamics can be obtained from the normal form by selecting the last three, *i.e.*, $n - r$, equations in the normal form and setting $\zeta = 0$

$$\dot{\eta} = \begin{bmatrix} v + \eta_3 \dfrac{m_2}{m_1 + m_2} \\ v + \eta_3 \dfrac{m_2}{m_1 + m_2} \\ \dfrac{F + L}{m_2} \end{bmatrix}.$$

The zeroing input is

$$u = \frac{m_1 L - m_2 F}{m_1 + m_2}.$$

The results depend on $\eta$, on system parameters, on parameters in $x^\circ$, and on the torques. The zero dynamics in this case represent only the one DOF of the combined inertias. The input $u$ required to keep the output, *i.e.*, the difference in velocity, equal to zero is precisely the torque transmitted through the clutch. It may be surprising that the order of the zero dynamics is three, where only one DOF is relevant, but this is caused by the fact that the positions of both masses are being tracked, they need not be equal due to different initial conditions, necessitating the use of an additional integrator.

In this example the zero dynamics could easily be computed by hand, because the model is linear. However, if effects like link elasticity and nonlinear damping are introduced in the model, solving the problem by paper and pencil becomes harder. A package for symbolic computation can be an asset in such a situation.

### 4.2. Double pendulum example

The double pendulum example starts by proposing a general model of two links with concentrated unit mass at the far end of the links, moving freely in a two dimensional space, and then requiring that the near end of a link is attached to a fixed mounting, while its far end is attached to the near end of the other link at a fixed distance of the mounting. A picture of such a set-up is in Fig. 3.

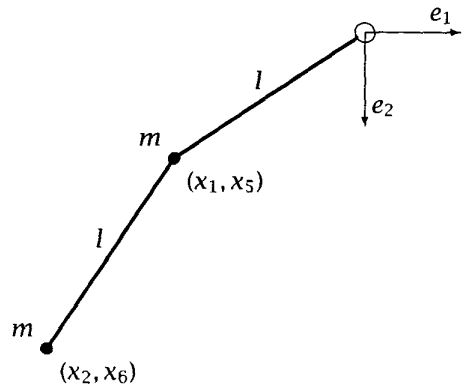


Figure 3: Double pendulum

The two joints allow free rotation but no relative translations, so both the Cartesian coordinates and the velocities of the parts of the links that connect at the joints should be equal, and the links are of fixed length. This gives a set of algebraic conditions. Stacking the Cartesian $e_1$-coordinates of the joint between the two links and of the endpoint of the second link and their derivatives first in $x$, *i.e.*, they are $x_i$, $i = 1, \ldots, 4$, and the $e_2$-coordinates and their derivatives last, *i.e.*, they are $x_i$, $i = 5, \ldots, 8$, taking the origin

in the base joint and neglecting gravity, the differential equations for the model are

$$\dot{x} = \begin{bmatrix} x_3 \\ x_4 \\ 0 \\ 0 \\ x_7 \\ x_8 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 2x_1 \\ 0 \\ 0 \\ 0 \\ 2x_5 \\ 0 \end{bmatrix} \lambda_1 + \begin{bmatrix} 0 \\ 0 \\ 2(x_1 - x_2) \\ 2(x_2 - x_1) \\ 0 \\ 0 \\ 2(x_5 - x_6) \\ 2(x_6 - x_5) \end{bmatrix} \lambda_2$$
$$= f(x) + g_1 u_1 + g_2 u_2$$

while the position constraints are

$$l^2 = x_1^2 + x_5^2$$
$$l^2 = (x_2 - x_1)^2 + (x_6 - x_5)^2.$$

The derivatives of the constraints, *i.e.*, constraints on the velocity, are

$$0 = x_1\dot{x}_1 + x_5\dot{x}_5 = x_1x_3 + x_5x_7$$
$$0 = (x_2 - x_1)(\dot{x}_2 - \dot{x}_1) + (x_6 - x_5)(\dot{x}_6 - \dot{x}_5)$$
$$= (x_2 - x_1)(x_4 - x_3) + (x_6 - x_5)(x_8 - x_7).$$

Here, $\lambda_i$ are Lagrange multipliers, related to the joint forces, and $l$ is the length of the links. Eliminating the velocity constraints is possible by keeping

$$y = \begin{bmatrix} x_1x_3 + x_5x_7 \\ (x_2 - x_1)(x_4 - x_3) + (x_6 - x_5)(x_8 - x_7) \end{bmatrix}$$

equal to zero. Computing the zero dynamics, in this case via the normal form, is possible and has been carried out with NONLCON. To obtain the especially simple normal form with $p(\zeta, \eta) = 0$, a complete set of independent solutions $\phi(x)$ for the partial differential equations

$$\frac{\partial \phi}{\partial x}\begin{bmatrix} g_1 & g_2 \end{bmatrix} = 0$$

is needed. The equations are completely integrable, and so solvable according to Frobenius' Theorem, if the distribution spanned by $g_1$ and $g_2$ is involutive, as is the case here, because the Lie product $[g_1, g_2] = 0$. The new coordinates as computed by NONLCON are

$$z = \Phi(x) = \begin{bmatrix} x_2 \\ x_1 \\ x_6 - 2l \\ x_5 - l \\ \dfrac{x_8(x_1 + x_2) + x_4(x_5 + x_6)}{x_2 - x_1} \\ \dfrac{(x_1x_7 - x_3x_5)(x_2 - x_1) + x_4(x_1x_6 - x_2x_5)}{x_1(x_2 - x_1)} \end{bmatrix}.$$

Using this result the zero dynamics is rather involved. The solutions of the partial differential equations are not unique, and the most simple solution is not always found. Another set of coordinates, due to M. Weiss, in this case leading to simpler expressions for the zero dynamics, is

$$z = \begin{bmatrix} x_1 \\ x_2 \\ x_5 \\ x_6 \\ x_5(x_3 + x_4) - x_1(x_7 + x_8) \\ x_4(x_6 - x_5) - x_8(x_2 - x_1) \end{bmatrix}.$$

The main simplification is that now the terms do not have a denominator. Nevertheless, the expressions for the zero dynamics are still too involved to be included. Remark that the input $u$ needed to keep the output $y$ equal to zero is related to the connecting forces between the links.

## 5. Conclusions

The paper shows that the concepts of zero dynamics has many applications and works out several of them in the area of modeling and model analysis. It also gives examples where the zero dynamics has been computed symbolically. The zero dynamics as used here are not a goal in itself, but a tool to modify a model that is to be set up or to get more insight in a model that is to be analyzed.

The second example shows that for system models that are moderately complex, symbolic computation cannot be fully enjoyed due to expression swell that is not easily eliminated. This implies that symbolic computation will not replace other tools, like those based on numerics, but will complement them.

## References


[1] 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. W. Nieuwenhuis, C. Praagman, and H. L. Trentelman, eds.), vol. 4, (Groningen, The Netherlands), pp. 2081-2085, June 1993.

[2] B. de Jager, "The use of symbolic computation in nonlinear control: Is it viable?," *IEEE Trans. Automat. Control*, vol. 40, pp. 84-89, Jan. 1995.

[3] A. Isidori, *Nonlinear Control Systems.* Berlin: Springer-Verlag, 3rd ed., 1995.

[4] H. Nijmeijer and A. J. van der Schaft, *Nonlinear Dynamical Control Systems.* New York: Springer-Verlag, 1991. Corrected 2nd printing.

[5] H. G. Kwatny and G. L. Blankenship, "Symbolic tools for variable structure control system design: the zero dynamics," in *Proc. of the IEEE Workshop on Robust Control via Variable Structure & Lyapunov Techniques*, (Benevento, Italy), IEEE, Sept. 1994.