Analysis and design of nonlinear control systems with the symbolic computation system MAPLE

Analysis and design of nonlinear control systems with the symbolic computation system MAPLE

During the last decade, important progress has been made in the development of analysis and design methodologies for nonlinear control systems. These methods are based on analytical analysis of nonlinear systems. The non-numeric computations that are involved are very tedious. In this study the use of symbolic computation programs to assist with these computations is discussed. The treatment is restricted to an extensive and consistent discussion of the zero dynamics and the exact linearisation theories. Several procedures are developed which offer computational tools and automate the analysis. Insight is also accorded in the use of symbolic computation. The results established so far seem to be promising and symbolic computation is found to be a useful research tool.

Harm VAN ESSEN
Department of Mechanical Engineering,
Eindhoven University of Technology,
The Netherlands

Introduction

Analysis for nonlinear control systems means mostly simulation, i.e. numerically solving the nonlinear differential equations describing the system. Design of nonlinear control systems is often based on a linear model of the system. Linearisation takes place in a variety of ways and once a linear model is obtained, several standard methods for analysis and design of control systems are available. So, although most systems are inherently nonlinear, the majority of controller design techniques is based on linear models. These control strategies usually provide adequate performance if the system is sufficiently linear around the operating point. However, if the system is highly nonlinear or its state deviates significantly from the state at which the linear model was obtained, this strategy may not be adequate. To ensure stability for a wider range of operating conditions, the controllers must be detuned and performance is degraded. In case a high performance - e.g. fast and accurate robot control, critical process control etc.- is required, controllers obtained from a linear model will not be adequate.

For highly nonlinear systems, controller design strategies using nonlinear models can be expected to provide improved performance. During the last decade progress has been made in the development of systematic design methodologies for nonlinear control systems. These are based on analytical analysis of nonlinear systems. A comprehensive view is presented by Isidori in his book *Nonlinear Control Systems, an Introduction* [1] in which several problems are posed and exact (analytical) solutions are given. The methods in this book are used in our study. Another source may be the book *Nonlinear Dynamical Control Systems* [2] by Nijmeijer and Van der Schaft. The non-numeric computations that are involved in the algorithms have to be done by hand. Because these computations are very tedious, these methods can only be applied when the computations are assisted by Symbolic Computation systems.

This study makes a contribution to the development of systematic design methodologies for nonlinear control systems. The contribution comprises the implementation of some analytical analysis and design strategies in a symbolic computation environment. In doing this, some specific choices have been made: the symbolic computation system *MAPLE* is used throughout this research and this study is restricted to a treatment of the *zero dynamics* and the *exact linearisation theories.* This means in particular that stabilisation and connected control objectives like tracking and attenuation will not be discussed. (Note also that several other areas in nonlinear control - such as nonlinear optimal, adaptive, and H_∞ control - are not treated.)

The goal of this study is twofold. First, to investigate the application of a modern symbolic computation program like MAPLE to the development and design of (nonlinear) control systems. Second, to offer computational tools which provide facilities for analysis and design of nonlinear control systems and which will be useful in further research to employ and develop these nonlinear control strategies.

Several researchers investigated the use of symbolic computation programs for control purposes and to develop packages for analysis and design of nonlinear systems. An actual overview can be found in [3]. Characteristic references are Zeitz [4] and Blankenship [5].

In section 2 symbolic computation and the program MAPLE are briefly discussed. In section 3 the control problems, and their solutions, that are treated in this study are formulated. Section 4 discusses the implementation in MAPLE and presents the results of this study. In section 5 an illustrative example is presented and section 6 contains the conclusions.

Symbolic Computation and MAPLE

A program for symbolic mathematical computation can deal with unassigned variables in general mathematical problems like solving (sets of) equations, differentiation, integration, and manipulation of expressions. This means for example that equations in x can be solved for x or that the influence of a parameter in a model can be made explicit by keeping it symbolic. A simple example is shown in (1) were an inverse matrix is computed and x is treated symbolically.

$$A = \begin{bmatrix} 1 & x \\ 2 & 3 \end{bmatrix} \quad A^{-1} = \frac{1}{-3 + 2x} \begin{bmatrix} -1 & x \\ 2 & -1 \end{bmatrix} \tag{1}$$

Already in the sixties the first programs for symbolic computation were developed. These programs, *Reduce* and *Macsyma,* have rather limited possibilities. Developments in mathematics and computer hardware have given ground to a new generation of programs. State of the art are *MAPLE* and *Mathematica.* These two programs offer roughly the same possibilities (and constraints!).

Symbolic computation offers enormous possibilities: as shown in this article, even complete algorithms can be implemented and analytical results -e.g. control laws- are obtained while tedious computations are performed by the computer. At the moment, the possibilities and power of symbolic computation have arrived at a level where several scientists show serious interest. Unfortunately also some disadvantages accompany the application of symbolic computation: inherent in the use of symbols is the creation of very long intricate expressions. The validation of such results is seriously restricted. Symbolic computation also requires large amounts of memory, 16 Mb is easily needed for serious applications.

MAPLE has been developed since 1981 by the University of Waterloo, Canada. In essence, MAPLE is an interactive program which is designed for advanced calculus and algebra. It also provides an internal programming language. This language allows to write functions and procedures, in which all MAPLE commands and functions can be used. Besides symbolic computation, MAPLE can also be used for numerical (floating point) calculations with infinite accuracy and for (2D,3D) plotting and animations of functions. More information on the use and properties of MAPLE can be found in [6] and [7]. At the moment version V.2 (1993) is available for several processor systems.

Control problems and solutions

In this section four important notions from nonlinear control theory will be briefly discussed on the basis of a simple SISO nonlinear system in general notation. The emphasis is on gaining insight, therefore the complicated mathematical background is skipped and for extensive formulations one can refer to [1], [2], [3], [8]. Instead, some explanatory block diagrams are used. It should be held in mind that all notions and algorithms are valid for SISO as well as for MIMO systems, although the last are a lot more complicated.

The normal form and the relative degree

Consider the nonlinear system in general state space notation (2), with input u, state vector x and output-function y.

$$\begin{aligned} \dot{x} &= f(x) + g(x)\,u \\ y &= h(x) \end{aligned} \tag{2}$$

The system equations can be transformed to the so called normal form (3) by means of a nonlinear change of coordinates in the state space. The new coordinates z depend on the output y and its derivatives. The transformation Φ is described in [1] and generally involves (among other operations) the solution of a set of partial differential equations.

$$\begin{aligned} z &= \Phi\,(x) \\ \dot{z} &= fn(z) + gn(z)u \\ y &= hn(z) \end{aligned} \tag{3}$$

The normal form is of interest because it reveals the structure inherent to the system and can be used to advantage in analysis and design of control systems. This structure is shown in *figure 1* and the corresponding dynamics are (4). From the block diagram it is clear that the normal form entails a series of r connected integrators, a state feedback, and a part of "remaining" dynamics.

$$\begin{aligned} y &= z_1 \\ \dot{z}_1 &= z_2 \\ &\cdots \\ \dot{z}_{r-1} &= z_r \\ \dot{z}_r &= b(z) + a(z)\,u \\ \dot{z}_i &= q_i \qquad (r+1 \leqslant i \leqslant n) \end{aligned} \tag{4}$$

The scalar r is called the relative degree of the system. It is a structural property of a dynamical system. (A MIMO system has a vector relative degree.) The relative degree is the number of times the output has to be differentiated in order to have the input appearing algebraically. From this it is clear that the relative degree is dependent on the choice of the output. When r is equal to the dimension n of the state space, the system is said to have a full order relative degree and it is obvious that in that case no remaining dynamics exist.

The zero dynamics

The notion of zero dynamics is an important property of nonlinear systems with regard to control objectives like feedback stabilisation and disturbance attenuation. In fact, asymptotical stability of zero dynamics is a sufficient and necessary condition for the existence of stabilising state feedback laws

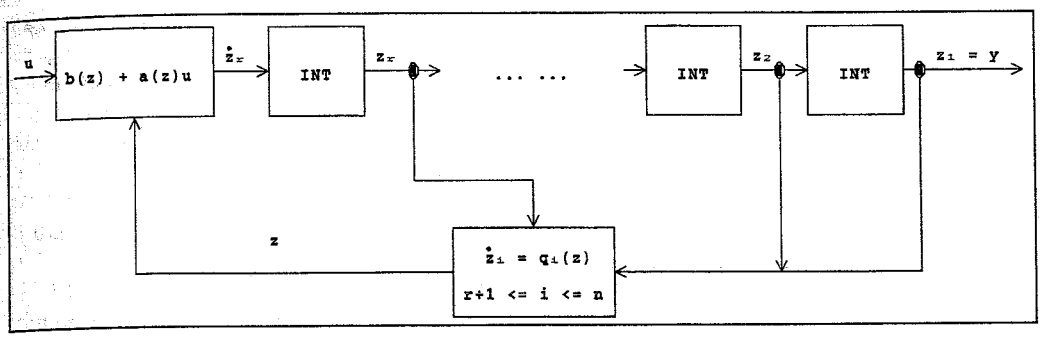

Figure 1.

[1]. The zero dynamics are regarded as the nonlinear analogue of the zeros of a transfer function of a linear system.

The concept of zero dynamics can be viewed in two different ways. First, the zero dynamics can be regarded as the internal dynamics imposed on the system when the constraint that the output is identically zero is enforced by an appropriate choice of input and initial state conditions. In the case of an invertible linear system, the eigenvalues corresponding to these dynamics are exactly the zeros of the system transfer function matrix. This *Problem of Zeroing the Output* is well-defined and analytically solved in [1]. The relevance of the equations in normal form with respect to the zero dynamics is obvious, this can be seen in the block diagram of figure 1 : when proper pairs of initial state conditions (z^0) and inputs u^0 (t) (consistent with the constraint that the output y(t) is identically zero for all times) are imposed on the system, the corresponding remaining dynamics are the zero dynamics. From this schematic view it is intuitively clear that when the zero dynamics are not stable, no stabilising feedback laws exist. In fact, the unobservable and unstable zero dynamics may present a time-bomb in the system !

The other way to approach the zero dynamics is to regard a nonlinear system rendered maximally unobservable via feedback. Here, the zero dynamics can be considered as the internal dynamics of the unobservable part of the closed-loop system. In fact these dynamics are the nonlinear analogue of the "numerator dynamics" of a linear system. The *Zero Dynamics Algorithm* [1] is an iterative procedure which constructs a feedback which renders the closed-loop system maximally unobservable. The internal dynamics of this unobservable part are the zero dynamics. It is important to state that this approach is not restricted to systems having a relative degree and is therefore applicable to a broader class of systems (systems which have a relative degree are fully incorporated). Because there is no coordinate transformation involved, the feedback and the zero dynamics computed by the Zero Dynamics Algorithm are expressed in the original coordinates.

Exact linearisation

Unlike Jacobian linearisation, which linearises the system only at the nominal operating point, methods based on exact linearisation provide linearisation of the model in the neighbourhood of the operating point. The nonlinear dynamical system is modified to one that behaves linearly by means of nonlinear transformation of coordinates and/or a nonlinear feedback. The nonlinear model can be linearised in either an *input-state* or in an *input-output* point of view. Once an exact linearised model is obtained, linear controller design techniques can be employed in an outer loop, to satisfy the control objectives. This principle is schematically shown in *figure 2*.

The problem of exact linearisation of the input-state equations of a nonlinear system is best understood from normal form analysis. Suppose a system (2) has a well defined and full-order relative degree. We already know that this means that after transformation to the normal form the system does not have a part of remaining dynamics in figure 1. After this transformation a static state feedback $u = (-b(z) + v) / (a(z))$ is applied, where v is the new reference input and $a(z)$, $b(z)$ are defined by the dynamics of the normal form (4). Then, the dynamics of the system evaluate to a straight chain of integrators from input v to output y, shown in *figure 3*, which is obviously linear in the state equations. Therefore a full order relative degree is a sufficient condition for a nonlinear system to be rendered linear. In [1] Isidori demonstrated that this is also a necessary condition.

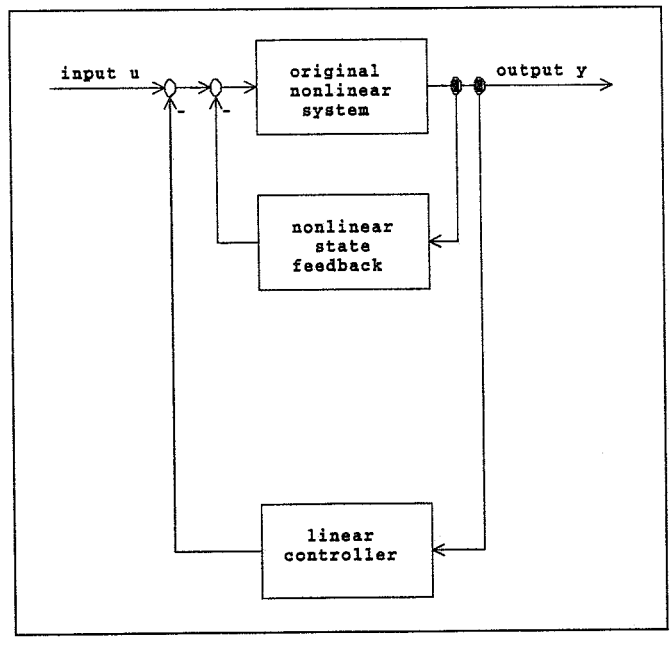

Figure 2.

We already know that the relative degree depends on the chosen output of the system. So, if we want to linearise the system we should find the output function for which the system has a full order relative degree. In [1], Isidori derived conditions for the existence of such output functions. In the proof of this result, it is indicated how the functions can be constructed.

Under some less restrictive conditions the nonlinear system can be linearised in an input-output point of view. This

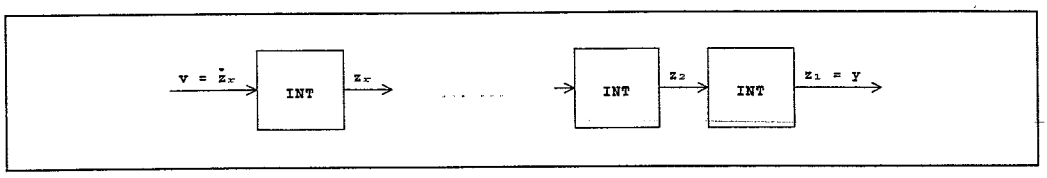

Figure 3.

problem, how to construct a static state feedback to give the system a linear input-output mapping, is solved by the *Structure Algorithm* [1]. This algorithm is a test for the fulfilment of the solvability as well as a procedure to construct the linearising feedback. There is no coordinate transformation involved, therefore the exact linearised system is expressed in the original coordinates x. The state feedback is computed as $u = \alpha(x) + \beta(x)\, v$. In this case no relative degree is necessary and the Structure Algorithm is applicable to a broader class of systems. However, when a full order relative degree is defined, the results from both approaches of exact linearisation are (apart from the different coordinate system) exactly equal.

MAPLE Implementation

In this section, the implementation - in the symbolic computation system MAPLE- of the algorithms mentioned in the previous section will be presented and discussed. With this implementation an attempt is made to automate the analyses for nonlinear systems in state space notation like (2). The algorithms are implemented in procedures, which are written in the MAPLE programming language and therefore only useable within MAPLE. In most cases, the procedure is a straightforward implementation of the algorithm. For details on the implementation one should refer to [8], in which all procedures are treated extensively. It is made possible to compute (and analyse) symbolically for SISO as well as MIMO systems:

- The (locally defined) relative degree.
- A change of coordinates leading to the normal form.
- The normal form, the zero dynamics and the corresponding unique zeroing input.
- A (locally defined) static state feedback which, if it exists, renders the system linear and controllable.
- A set of output functions which, if they exist, give the system a full order relative degree. If such an output function does not exist, an output function which maximises the relative degree can be computed, but only for SISO systems.
- A (locally defined) static state feedback which, if it exists, renders the system linear in an input-output sense.

Classes in which the procedures can be ordered

A total of 19 procedures have been written. Two classes can be distinguished with respect to the object of each procedure. In the first class, a number of procedures concerning basic mathematical computations such as special differential operators (e.g. Lie bracket), a test on involutivity and procedures to solve (sets) of partial differential equations are written. This is of particular interest because MAPLE does not yet provide facilities to solve partial differential equations and has difficulties with solving sets of ordinary differential equations. The new procedures are able to construct solutions of a special system of partial differential equations of the first order. The algorithm which solves the equations is based on a constructive proof of the *Frobenius theorem* [1]. This class also represents a few standard MAPLE procedures, mainly concerning Linear Algebra, which are adjusted ad-hoc and extended to fit in the algorithms.

In the second class, a number of procedures concerning the analyses themselves are written. The procedures in this class are the main results of this study, they offer symbolic computational tools to perform analysis and design strategies for nonlinear (control) systems. The procedures are ordered in the scheme of *figure 4*. Based on this scheme, two interesting remarks can be made: in the first place the two fields of analysis can be clearly distinguished (columns left and right), but also their common foundation in the relative degree and the normal form can be recognised. In the second place a clear distinction can be made (above and under the dashed line) between the analyses that are applicable to systems for which the relative degree can be defined *(normform, statelin)* and analyses applicable to a broader class of systems *(extnormform, inoutlin)*. As stated, the results of both approaches are equivalent in case of a well defined relative degree.

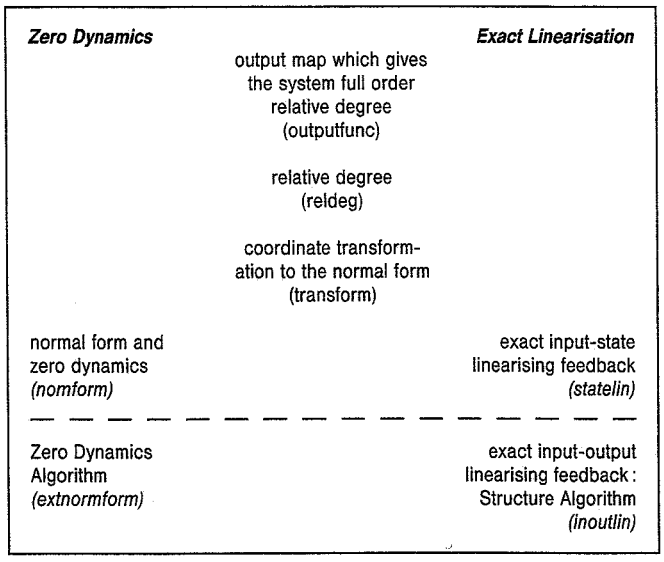

Figure 4.

Discussion on the use of MAPLE

Without too many difficulties, it has been possible to implement the analytical algorithms in the MAPLE programming language. The results established so far with the procedures seem to be very promising, an example of the possibilities is presented in the next section. Additional research is required to find the limits of the procedures and MAPLE. At this moment it can be stated that the symbolic computation software is not yet mature enough to solve large-scale problems [3].

Concerning the limits of the procedures, these may be in the implementation and can possibly be adjusted by an appropriate extension of the procedure. Of course, and most likely, the limits may also be in the proper-

ties of the system that is being analysed. In most cases a suitable error message is returned, provided by either MAPLE or the procedure itself.

Concerning the limits of MAPLE some remarks can be made. The built-in possibilities are strictly limited, e.g. not all systems of equations can be solved (even if they seem to be well-solvable), there exist limits the maximum object size, and the computer memory is a limiting factor. Another problem with MAPLE is that due to the internal representation of symbolic results, it is sometimes hard to perform (internal) validation. It is, however, expected that these and other shortcomings will be improved in future versions. In general, symbolic computations in MAPLE are surprisingly fast. Runtimes for rather complicated problems are often below 60 seconds.

Example

Consider a mechanical arm consisting of two rigid links, *figure 5,* interconnected by a spring joint. This model can be considered as the most simplified model of a flexible one-link robot arm. In state space description:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = f(x) + g(x)\,u$$

$$f(x) = \begin{bmatrix} x_3 \\ x_4 \\ \dfrac{\theta n (x_3+x_4)^2 \sin x_2 + (\theta + n\cos x_2)\, K x_2 + n^2 x_3 \sin x_2 \cos x_2}{\Delta} \\ \dfrac{-(\theta+n\cos x_2)(x_4+2x_3)\, n x_4 \sin x_2 + (\gamma+2n\cos x_2)(n x_3^2 \sin x_2 + K x_2)}{\Delta} \end{bmatrix}$$

$$g(x) = \begin{bmatrix} 0 \\ 0 \\ \dfrac{\theta}{\Delta} \\ \dfrac{-(\theta+n\cos x_2)}{\Delta} \end{bmatrix}$$

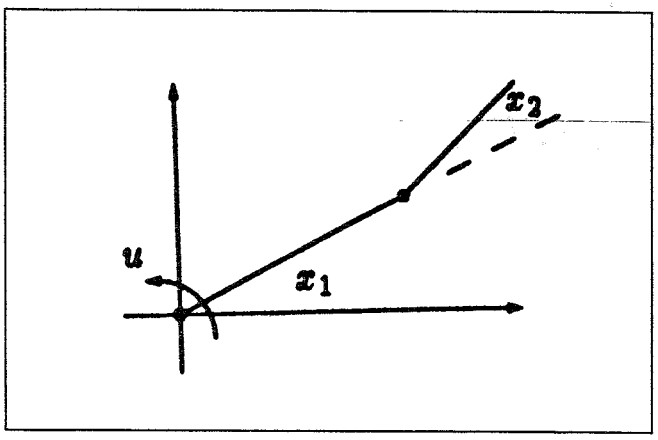

Figure 5.

x_1 = position of the first link

x_2 = relative position of the second link

x_3, x_4 = corresponding velocities

u = torque on the first link

K = constant of spring joint

$\Delta = \theta(\gamma-\theta) - (n\cos x_2)^2$

γ, θ, n = system parameters: functions of mass, length and inertia

The relative degree (reldeg)

SISO system, relative degree is a scalar: [2], with outputfunction y = x[1]. Note that this is not a full order relative degree.

Transformation to the normal form (transform)

transformation: z = Φ (x) =

[x[1], x[3], x[2], n/θ x[3]cos(x[2]) + x[3] + x[4]/θ]

inverse transformation:

{ x[1] = z[1], x[2] = z[3], x[3] = z[2],

x[4] = - z[4] + z[2] n/θ cos(z[3]) + z[2] }

The normal form (normform)

$\dot{z} = fn(z) + gn(z)\,u$

With fn(z), gn(z) respectively

[z[2],

- (n^2 z[2]^2 sin(z[3]) cos(z[3]) θ + n sin(z[3]) z[4]^2 θ^2

- 2 n^2 sin(z[3]) z[4] θ z[2] cos(z[3])

+ n^3 sin(z[3]) z[2]^2 cos(z[3])^2 + z[3] K θ^2

+ z[3] K n cos(z[3]) θ) / (θ (- θ γ + θ^2 + n^2 cos(z[3])^2)),

z[4] - z[2] n/θ cos(z[3]) - z[2],

n^2 z[2]^2 sin(z[3]) cos(z[3]) - sin(z[3]) z[4] θ z[2] n - z[3] K θ) / θ^2]

$$\begin{bmatrix} 0 \\ \theta \,/\, (-\theta\gamma + \theta^2 + n^2\cos(z[3])^2) \\ 0 \\ 0 \end{bmatrix}$$

Zero dynamics (normform)

the zero dynamics: (outputfunction y = x[1] is forced to be zero)
{ x[1]dot = 0, x[2]dot = x[4], x[3]dot = 0, x[4]dot = -x[2] K / θ }
(corresponding to a vibration in the second link, intuitively clear)

and the unique zeroing input:

u = { - n sin(x[2]) x[4]^2 + x[2] K + x[2] K/θ n cos(x[2])]

Exact linearisation of the state-input equations (statelin)

Not possible to compute a state feedback because this system does not have a full order relative degree.

Finding output functions which give the system a full order relative degree (outputfunc)
MAPLE failed in finding solutions for a partial differential equation that should be solved to find the output function
- limitations of MAPLE (e.g. in solving a coupled set of ordinary differential equations)
- problems with exponential and goniometric entries
- problems in verifying results generated by MAPLE

Exact linearisation of the input output mapping (inoutlin)
Though an exact input-state linearisation does not exist, an input-output linearising feedback can be computed

$$\dot{x} = f_l(x) + g_l(x)\, v$$

linearising feedback $u = u(v,x) =$

$$[- (- v[1]\, \gamma + v[1]\, \theta + v[1]/\theta\, n^2 \cos(x[2])^2 + n \sin(x[2])\, x[3]^2$$
$$+ 2\, n \sin(x[2])\, x[3]\, x[4] + n \sin(x[2])\, x[4]^2 + x[2]\, K$$
$$+ x[2]\, K\, n/\theta \cos(x[2]) + n^2/\theta\, x[3]^2 \sin(x[2]) \cos(x[2]))]$$

and linearised system dynamics *fl, gl* respectively:

$$[\, x[3],\ x[4],\ 0,\ - n/\theta\, x[3]^2 \sin(x[2]) - x[2]\, K/\theta\,]$$

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ - (1 + n/\theta \cos(x[2])) \end{bmatrix}$$

For more examples refer to [3] or [8]. In [3] a benchmark problem of a vehicle simulation problem is formulated.

Conclusions

The collection of procedures developed in this study forms a contribution towards the development of systematic design methodologies for nonlinear control systems. The procedures offer computational tools for symbolic analysis and design of nonlinear control systems. Together, the procedures provide a complete and consistent treatment of two important fields of interest in the (analytic) analysis and design of nonlinear control systems: exact linearisation and the zero dynamics. It is made possible to compute and analyse symbolically - for a class of (MIMO as well as SISO) systems for which the relative degree can be defined - the relative degree, the normal form, the zero dynamics, an exact linearisation of the input-state equations, and an exact linearisation of the input-output behaviour. Also for a broader class of systems the zero dynamics and the second form of exact linearisation can be computed symbolically. In this broader class, the systems which do have a relative degree are fully incorporated.

The results established so far with the procedures are promising, especially for textbook and small scale problems, but the symbolic computation software is not yet mature enough to solve interesting large scale problems. More general shortcomings of MAPLE are found to be the inability of solving partial differential equations and problems that were encountered in the automatic validation of (intermediate) symbolic results. For application in the procedures described in this article, these deficiencies are redressed by appropriate extension of MAPLE. Being far from perfect, it is recommended that these extensions are further analysed and added to MAPLE in a generalised form.

As follows from the successful implementation in the MAPLE programming language, MAPLE seems to be a suitable environment for the use of analytic control strategies. It is stressed that the use of symbolic computation is not restricted to the present implementations: Symbolic Computation is found to be a valuable research tool.

Acknowledgement
Special thanks are due to Dr.Ir. A.G. (Bram) de Jager, initiator and coach of my Master's study, for his valuable contributions.

References
1. Isidori, Alberto; Nonlinear Control Systems, An Introduction, Springer-Verlag, Berlin, 2nd edition, 1989.
2. Nijmeijer, H. and Schaft, A.J. van der; Nonlinear Dynamical Control Systems, Springer-Verlag, New York, corrected 2nd printing, 1991.
3. Jager, Bram de; 'The use of symbolic computation in nonlinear control: Is it viable?', Proceedings of the 32nd Conference on Decision and Control, San Antonio, USA, vol. 1, pp. 276-281, December 1993.
4. Rothfuß, R., Schaffner, J., and Zeitz, M.; 'Rechnergestützte Analyse und Synthese nichtlinearer Systeme', Nichtlineare Regelung: Methoden, Werkzeugen, Anwendungen, vol. 1026 of VDI Berichte, pp. 267-291, VDI-Verlag, Dsseldorf, 1993.
5. Akhriff, O. and Blankenship, G.L.; 'Computer Algebra algorithms for nonlinear control', 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, Springer-Verlag, Berlin, 1988.
6. Char, Bruce W. et al; 'MAPLE V Language Reference Manual', Springer-Verlag, New York, 1991.
7. Char, Bruce W. et al; 'MAPLE V Library Reference Manual', Springer-Verlag, New York, 1991.
8. Essen, Harm van; Symbols speak louder than Numbers; Analysis and Design of Nonlinear Control Systems with the Symbolic Computation System MAPLE, Master's thesis, report no. WFW92.061, TUE-WFW, Eindhoven, 1992.

Harm VAN ESSEN

■ Department of Mechanical Engineering, WOC-WET WH3.132, Eindhoven University of Technology, P.O.Box 513, NL-5600 MB, Eindhoven, The Netherlands.

■■ Harm van Essen was born in 1968. In 1986 he started studying Mechanical Engineering at Eindhoven University of Technology where he graduated Cum Laude in 1992. For his Master's thesis he was awarded the 'Control Engineering Award 1992', presented by the Measurement and Control Engineering Division of the Royal Dutch Institute of Engineers. This article is based on his Master's thesis. After his studies he became involved in the field of Energy Technology in a two years' AIO-programme at the same university. His current interests are in decentralised cogeneration units which offer flexible trade-off between process steam demand and electrical power yield.