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

Citation for published version (APA):

Jager, de, A. G. (1995). The use of symbolic computation in nonlinear control : is it viable? *IEEE Transactions on Automatic Control*, 40(1), 84-89. <https://doi.org/10.1109/9.362897>

DOI:

[10.1109/9.362897](https://doi.org/10.1109/9.362897)

Document status and date:

Published: 01/01/1995

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

Abstract—To help along the analysis and design of nonlinear control systems the NONLINCON package, an acronym for *Nonlinear Control*, was developed. This note addresses the usefulness of symbolic computation, and of the NONLINCON package in particular, for the symbolic analysis and design of nonlinear control systems. The symbolic computation program MAPLE is used as computing substratum. Textbook problems show that the NONLINCON package can be used successfully. A larger scale problem is too complex, however, to be solved with the current versions of NONLINCON and MAPLE. The conclusion is that symbolic computation is a viable approach for textbook problems, but not yet for more complex ones.

I. INTRODUCTION

The use of symbolic computation for control purposes is investigated by several researchers. Zeitz *et al.*, [1] applies the program MACNON, based on MACSYMA, to analyze observability and reachability, and to design observers and controllers for nonlinear systems. Blankenship [2] also used MACSYMA to solve some control problems with his implementation CONDENS. The use of MAPLE for several control problems is reported in [3]. Some problems reported in this note, e.g., with solving partial differential equations, are partly resolved in [4]. They describe a MAPLE package, here called NONCON, that can compute, e.g., the zero dynamics and provide solutions for exact linearization problems. In this note we illustrate the use of this package by applying it to some textbook problems and a more complex one.

The main goals and contributions of this note are

- a proof of the viability of symbolic computation for some problems in the analysis and design of nonlinear control systems
- to show the characteristics of a prototype implementation
- to give some examples and to document some applications
- to discuss directions for future research
- to familiarize a larger audience in the control community with the use of symbolic computation.

The note is structured as follows. First, Section II formulates the control problems and presents the algorithms used. The implementation in NONLINCON of these algorithms is treated in Section III. Section IV presents textbook examples for some problem areas. A more complex example is treated in Section V. Section VI closes with conclusions and gives directions for future research.

II. THE PROBLEMS

Of 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. In the presentation of these problems, we follow Isidori [5].

A. Preliminaries

We start with a nonlinear model of a plant and assume that it can be described adequately by a set of nonlinear differential equations,

Manuscript received March 26, 1993; revised February 12, 1994.

The author is with the Faculty of Mechanical Engineering, Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands. IEEE Log Number 9405674.

affine in the input u , and without direct feed-through from input to output

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

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 $y \in \mathbb{R}^m$. The number of inputs is equal to the number of outputs, i.e., the plant is square. This assumption is for convenience only. Part of the theory can also be derived if the number of inputs is not equal to 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 type of control law chosen is static state-feedback. Therefore, the value of the input vector $u(t)$ depends on the state $x(t)$ and a new reference input vector $v(t)$ as

$$u = \alpha(x) + \beta(x)v \quad (2)$$

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

For nonlinear models it is appropriate to allow for a nonlinear change of coordinates

$$z = \Phi(x). \quad (3)$$

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

B. Relative Degree

The nonlinear model (1) is said to have a vector relative degree $\{r_1, \dots, r_m\}$ at $x = x^0$ if

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

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

Here $L_f^k h_i(x)$ means the k th successive Lie derivative of the scalar function $h_i(x)$ in the direction of the vector field f , e.g., $L_f h_i(x) = (\partial h_i(x)/\partial x) f(x)$. The matrix A is sometimes called the decoupling matrix. The relative degree can also be interpreted as the number of times the outputs have to be differentiated before the input explicitly appears.

The models we consider do not necessarily have a relative degree, either because the first condition cannot be satisfied or because the matrix A is singular at x^0 .

C. Normal Form

When the model has a well-defined relative degree we can use a change of coordinates (3), with $z = (\xi, \eta)$, to transform (1) to the normal form

$$\begin{aligned} y_i &= h_i(x) = \xi_i^1 \\ \dot{\xi}_i^1 &= \xi_i^2 \\ &\vdots \\ \dot{\xi}_i^{r_i} &= b_i(\xi, \eta) + \sum_{j=1}^m a_{ij}(\xi, \eta) u_j \quad \text{for } i = 1, \dots, m \\ \dot{\eta}_i &= q_i(\xi, \eta) + p_i(\xi, \eta) u \quad \text{for } i = r+1, \dots, n \end{aligned} \quad (4)$$

with $r = \sum_{j=1}^m r_j$ and

$$a_{ij}(\xi, \eta) = L_{g_j} L_f^{r_i-1} h_i(\Phi^{-1}(\xi, \eta))$$

$$b_i(\xi, \eta) = L_f^{r_i} h_i(\Phi^{-1}(\xi, \eta)) \quad \text{for } i, j = 1, \dots, m.$$

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

$$\xi^{(r)} = b(\xi, \eta) + A(\xi, \eta)u \quad (5)$$

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

where $\xi^{(r)}$ contains $\xi_i^{r_i}$, $i = 1, \dots, m$. The equation for $\dot{\eta}$ is called the internal dynamics. Because A is nonsingular if the relative degree is well defined, the control

$$u = A^{-1}(v - b) \quad (6)$$

with the new input v , is properly defined and linearizes the part of model (5) that is visible at the output.

D. Zero Dynamics

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

When the model has a well-defined relative degree, the zero dynamics follows from the normal form, by substitution of the output nulling input u^* and using the property that the states ξ can be set to zero in the internal dynamics.

For models without a relative degree the zero dynamics can be computed by using the Zero Dynamics Algorithm. The way this algorithm works is by considering a sequence of nested submanifolds M_i , with $M_i \supset M_{i+1}$ and $M_0 = h^{-1}(0)$, i.e., the first submanifold is the inverse image of the point $y = 0$. When some conditions are fulfilled this sequence converges to the locally maximal output zeroing submanifold Z^* in some neighborhood of x^o and there exists a mapping u^* such that $f^*(x) = f(x) + g(x)u^*(x)$ is tangent to Z^* . The pair (Z^*, f^*) is called the zero dynamics of the model. When the mapping $H(x)$ is defined in a neighborhood U of x^o by $Z^* \cap U = \{x \in U: H(x) = 0\}$ the input u^* can be computed as the solution of $L_f H(x) + L_g H(x)u^* = 0$. For further details of this algorithm and the conditions to be imposed to guarantee convergence of the sequence to Z^* , we refer to [5, Section 6.1].

E. Input-Output Exact Linearization

The input-output exact linearization problem is: find out if it is possible to transform model (1) to a linear one by state feedback (2) and compute this feedback. The linearity property should be established between the new input v and the output y . Formally, we are looking for a neighborhood U of x^o and a static state feedback such that for all $k \geq 0$ and all $1 \leq i, j \leq m$ the expression $L_{(g_j)} L_f^k h_i(x)$ is independent of x on U .

For models with a well-defined relative degree, the input-output exact linearizing feedback follows from the normal form and is given by (6).

For models without a relative degree, the static state feedback that solves the problem can be computed with the Structure Algorithm. This algorithm uses a sequence of Toeplitz matrices

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

where

$$T_k(x) = \begin{bmatrix} L_{g_1} L_f^k h_1(x) & \cdots & L_{g_m} L_f^k h_1(x) \\ \vdots & \ddots & \vdots \\ L_{g_1} L_f^k h_m(x) & \cdots & L_{g_m} L_f^k h_m(x) \end{bmatrix}$$

for $0 \leq k \leq 2n - 1$. The conditions for existence can be expressed in rank conditions on $M_k(x)$. Within the Structure Algorithm the feedback functions $\alpha(x)$ and $\beta(x)$ are constructed. For further details of this algorithm we refer to [5, Section 5.4].

F. State-Space Exact Linearization

The state-space exact linearization problem is: under which conditions is it possible to transform the model (1) to a linear and controllable one by state feedback (2) and a change of coordinates (3)? The linearity property should be established between the new input v and the transformed state z .

This problem has been solved. The solution is only valid for models with a well-defined relative degree and requires the existence of (synthetic) outputs for which the model has a full order relative degree, $r = n$. When this cannot be obtained, it is sometimes convenient to strive after a maximal relative degree. Then the corresponding input-output linearizing state feedback realizes a minimal dimension of the internal dynamics.

Using the Lie product

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

define the adjoint $\text{ad}_f^k g_i = [f, \text{ad}_f^{k-1} g_i]$ with $\text{ad}_f^0 g_i = g_i$. Then define the distributions

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

We now state the conditions for a solution [5, Theorem 5.2.4].

Theorem 1: Given the model

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

with rank $g(x^o) = m$. There exists a solution for the state-space exact linearization problem if and only if

- 1) G_i has constant dimension near x^o for each $0 \leq i \leq n - 1$
- 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 given conditions are fulfilled, there exist solutions $\lambda_i(x)$, $i = 1, \dots, m$, for the following partial differential equations

$$L_{g_j} L_f^k \lambda_i(x) = 0, \quad \text{for } 0 \leq k \leq r_i - 2 \quad \text{and} \quad 1 \leq j \leq m.$$

Also $\sum_{i=1}^m r_i = n$, where the set of integers $\{r_1, \dots, r_m\}$ is the relative degree vector. The m functions λ_i can be computed, based on a constructive proof of Theorem 1. Using the functions λ_i , the change of coordinates Φ and state feedback $u = \alpha(x) + \beta(x)v$ follow.

III. SOLUTION WITH SYMBOLIC COMPUTATION

An analysis of the algorithms shows that symbolic computation programs should be able to compute the Lie derivative and Lie product, the Jacobian, the rank, the Gauss Jordan decomposition, the inverse of a matrix, and the determinant of a matrix; to do matrix-vector and matrix-matrix multiplication; and to test involutiveness. More mundane facilities like symbolic substitution are needed also. The main problems are the computation of solutions for sets of nonlinear equations, e.g., to compute the kernel of a mapping, and

Fig. 1. Structure of NONLINCON.

for sets of partial differential equations. These problems present the major bottlenecks for the symbolic solution of our problems.

Solutions for the problems described in the previous section

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

are included in NONLINCON. For problems 1) and 4) the model should have a well-defined relative degree; for 2) and 3) this is not necessary. Implementations of other algorithms, e.g., the Dynamic Extension Algorithm and a method to solve partial differential equations, are included in this package also. The structure of the implementation is sketched in Fig. 1. For a thorough discussion of the implementation, see [6].

Above 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.

We also noted that another main problem was to solve sets of partial differential equations. In MAPLE almost no facilities are available for their solution. To improve this, the following route was chosen.

The partial differential equations we have to solve are from the "completely integrable" type; in other words, based on Frobenius' Theorem, we know that a solution for the partial differential equations exists. To compute the solutions Frobenius' Theorem itself is of no help. This problem was solved by computing the solutions with an algorithm based on a constructive proof of Frobenius' Theorem. The procedure is as follows. The solution of the partial differential equation can be constructed from the solutions of related sets of ordinary differential equations. Because MAPLE provides some facilities for this type of problems, the `dsolve` command, the problem seems solved. The `dsolve` command is not very powerful, however, 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 setting up 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, which would be a more rigorous option. Despite this extension the solution of the partial differential equations is often unsuccessful, so NONLINCON cannot complete the computations.

A minor problem was that some standard MAPLE `linalg` functions are only suitable for rational polynomials. This was too limited for our purposes. Therefore, the rank and implicitly the `gausselim` and `gaussjrd` procedures were extended, so a larger class of problems could be handled. This resulted in the new functions `extrank`, `extgausselim`, and `extgaussjrd`.

Fig. 2. Robot with two revolute joints.

IV. TEXTBOOK PROBLEMS

To illustrate the use of NONLINCON we consider several examples. The first example computes the zero dynamics of a single-input single-output model with a well-defined relative degree. For the second example the zero dynamics is computed with the Zero Dynamics Algorithm. For the last example the input-output linearizing state feedback is computed with the Structure Algorithm. The last two examples are based on models that do not have a well-defined relative degree. All examples are contrived ones. The first example is taken from [7, Example 12.43] and the other two from [5, Examples 6.1.2 and 5.4.1].

Example 1: The model for a robot with two revolute joints (see Fig. 2) can be derived from the kinetic and potential energy T and V

$$2T = m_1 l_1^2 \dot{\theta}_1^2 + m_2 (l_1^2 \dot{\theta}_1^2 + l_2^2 (\dot{\theta}_1 + \dot{\theta}_2)^2 + 2l_1 l_2 (\dot{\theta}_1 + \dot{\theta}_2) \cos \theta_2)$$

$$V = -g\{(m_1 + m_2)l_1 \cos \theta_1 + m_2 l_2 \cos(\theta_1 + \theta_2)\}.$$

The inputs u_1 and u_2 are the joint torques and the outputs y_1 and y_2 the joint positions. The state x of the model corresponds with the degrees-of-freedom and their derivatives in the following way $x^T = [\theta_1 \ \theta_2 \ \dot{\theta}_1 \ \dot{\theta}_2]$. The aim is to derive the dynamics of the model when it is constrained. There are two cases, the first one with $y = y_1 = \theta_1$ constrained to 0 and $u_2 = 0$, the second one with $y = y_2 = \theta_2$ constrained to 0 and $u_1 = 0$. According to [7] the zero dynamics are given by (with all model parameters set to one)

$$\dot{x}_2 = x_4 \quad \dot{x}_4 = g \sin x_2$$

respectively

$$\dot{x}_1 = x_3 \quad \dot{x}_3 = (3/5)g \sin x_1.$$

The following log of a NONLINCON session shows that these results can be reproduced. Two functions are used: `normform` to compute the zero dynamics and `transform` for the inverse transformation. The log for the first case, where $y = \theta_1$, is found at the bottom of the next page (as marked by (x)). The second case for $y = \theta_2$ is found at the bottom of the next page, as marked by (y). The results coincide with the ones given above.

Example 2: The model of the system is

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

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

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

is singular for $x_2 = 0$. The zero dynamics is given in [5] by $\dot{x}_3 = -x_3$ and the zeroing input by

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

The next edited log of a NonLinCon session shows that the zero dynamics can be computed. From the functions supplied we only need `extnormform` that implements the Zero Dynamics Algorithm is shown at the bottom of the page (as indicated by (z)). The term computed last (for $\dot{x}_3$) represents the zero dynamics. The results coincide with the ones given above.

Example 3: The model of the system is

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

This model has no well-defined relative degree because

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

is singular for all x . According to [5] the necessary feedback for input-output linearization is $u = \alpha(x) + \beta(x)v$ with

$$\alpha = -\begin{bmatrix} -x_1 + x_3 \\ 2x_1^3 + 2x_1x_2 + x_1x_3 \end{bmatrix}, \quad \beta = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The following shows that NonLinCon can compute the input-output linearizing state feedback. From the functions supplied we only need `inoutlin` that implements the Structure Algorithm, which is found at the bottom of the next page. The computed state feedback agrees with the previous result.

Fig. 3. One-track model of a vehicle.

V. VEHICLE SIMULATION PROBLEM

This more complex problem is derived from an inverse simulation problem in multibody dynamics. A vehicle is required to perform a standardized maneuver. To simulate the maneuver the inputs to the model must be known. Normally, the maneuver is such that a suitable selected output of the model can be set to zero. Often this output does not fully determine the behavior of the model 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 for a stable simulation.

As an example, we use a simple two-dimensional one-track model of a vehicle, with traction and cornering forces acting on the tires. It looks like the model of a bike, because it only considers the center line of a motor vehicle (see Fig. 3).

The required maneuver is steady-state turning: the longitudinal speed of the center of mass M is constant and a point P on the

```
* normform *
zero dynamics is
[eta[1]dot = eta[2], eta[2]dot = grav sin(eta[1])]
output nulling input is
2
(x)
```

```
[ sin(eta[1]) eta[2] + cos(eta[1]) grav sin(eta[1]) ]
* transform *
inverse transformation
{x[2]=eta[1], x[4]=eta[2], x[1]=zeta[1], x[3]=zeta[2]}

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

```
* extnormform *
1st step, output zeroing submanifold:
{x[2] = 0, x[1] = 0}
2nd step, output zeroing submanifold:
{x[2] = 0, x[1] = 0, x[4] = 0}
3rd step, output zeroing submanifold:
{x[2] = 0, x[1] = 0, x[4] = 0, x[5] = 0}
3rd step, matrix L_gH has full rank m
the zero dynamics: (zerodyn)
{x[1]dot=0, x[2]dot=0, x[3]dot=-x[3], x[4]dot=0, x[5]dot=0}
the zeroing input: (uzero) [ 0, - x[3] ]
(z)
```

center line should describe a circle of specified radius. The specific problem is for which distances p , from P to M , the zero-dynamics is stable or unstable.

The equations of motion of the model are

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

with inputs F_{lr} (traction force) and δ (steering angle). The three degrees-of-freedom x_m , y_m , and θ , are, respectively, the coordinates of M and the orientation of the vehicle with respect to a fixed reference frame $(\bar{e}_x^0, \bar{e}_y^0)$, indicated by the superscript 0. The steering angle δ and the drift angles α_f and α_r are given with respect to a body fixed reference frame. The vehicle has mass m and moment of inertia J with respect to M . The forces acting on the vehicle are the traction force F_{lr} , and the lateral tire forces F_f , F_r . The last two forces can be expressed in the drift angle and the normal tire force F_n by the so-called magic formula for pure slip [8]

$$F(F_n, \alpha) = D(F_n) \sin(C \operatorname{atan}(B\xi - E(B\xi - \operatorname{atan}(B\xi)))) \quad (7)$$

with $\xi = \alpha + S_h$. The dependency holds for both front and rear. The parameters in this formula have to be fitted to experimental data. The drift angles can be expressed in the degrees-of-freedom and steering angle by

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

with

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

and

$$R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}, \quad \dot{x}_m^0 = \begin{bmatrix} \dot{x}_m \\ \dot{y}_m \end{bmatrix},$$

$$\dot{s}_f^1 = \begin{bmatrix} a \\ 0 \end{bmatrix}, \quad \dot{s}_r^1 = \begin{bmatrix} -b \\ 0 \end{bmatrix}$$

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

By choosing the outputs as

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

with R_d the desired radius of the circle and V_d the desired longitudinal speed, the required maneuver corresponds with a zero output. The output y_2 depends on the distance p from P to M . To solve the problem, first compute the zero dynamics and then find out its stability as a function of p .

The first problem encountered in computing the zero dynamics is the nonaffine character of the equations of motion with respect to the steering angle δ . To overcome this problem an integrator can be added to the model and $\dot{\delta}$ can be regarded as a new input. Then the model has a singular decoupling matrix A , however, which makes the analysis more difficult. The `extnormform` function can be used, but this leads to insurmountable problems in the computation due to huge memory needs. Another solution is to add another integrator and use $\dot{F}_{lr}$ as a new input, making the decoupling matrix regular by "delaying" the input F_{lr} also. The extension with two integrators allows the use of the standard transformation to the normal form with the function `normform`, although for a larger model. Due to the larger number of states, again problems arise with the memory needed.

To simplify the problem another "saturating" function was used in (7), $\sqrt{\cdot}$ instead of $\sin(\operatorname{atan}(\cdot))$. Now, the transformation can be computed, but the inverse transformation cannot, because of an artificial limit in the size of objects allowed by the software, putting an untimely end to the computation. Results are therefore only available in terms of the old coordinates x , but not in z .

Another attempt to solve the problem of nonaffine input is not to extend the model, but to simplify it so that the input δ appears linear in the equations. To this end, the assumption that δ is small has to be adopted. Then, by using a Taylor series expansion, the system of equations can become linear in δ if the dependency on δ of the front lateral tire force is dropped, so δ should also be small compared with α_f , an unrealistic assumption. This time the transformation to the normal form can be computed, but the set of nonlinear equations, needed to compute the inverse transformation, could not be solved, not due to memory constraints but due to limitations in the `solve` function of MAPLE. Further model simplifications are, of course, possible and may remedy this.

```
* inoutlin *
the i/o linearized system dynamics: (fl, gl)
      2      3
[ x[2]+ x[1], -2x[1]x[2] - 2x[1], 0, x[1]- x[3],
      2
x[5] + x[3] + x[1]x[2] - x[2]x[3] ]
      [ 0 0 ]
      [ 0 1 ]
      [ 1 0 ]
      [ 1 0 ]
      [ x[2] 0 ]
the explicit feedback: (alpha, beta)
      3
[ x[1] - x[3], - 2 x[1] x[2] - 2 x[1]- x[1] x[3] ]
      [ 1 0 ]
      [ 0 1 ]
```

Still another possibility is to use an algorithm that does not require a model with affine inputs; see [7]. This has not been implemented yet.

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

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

The first two problems should be resolved by the MAPLE developers. The last problem is not so easy to solve because the memory requirements are likely to be exponential or double exponential with respect to the problem size [9]. More efficient algorithms are necessary, but some problems will still be computationally intractable.

Numerics can be helpful, e.g., to solve sets of nonlinear (partial differential) equations. For the vehicle problem we resorted to a purely numeric approach to find out the stability of the zero dynamics. We propose to use this problem as a benchmark, or yardstick, for the viability of symbolic computation programs in nonlinear control.

VI. 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 NONLINCON package. At the moment, the computations cannot be done for complex models, due to the limited capability to solve sets of nonlinear (differential) equations. Therefore, the designers of control systems cannot yet routinely compute solutions for these problems, using tools based on symbolic computation programs. To remedy this, we recommend extending the capabilities of symbolic computation programs for solving large and intricate sets of nonlinear (differential) equations. It is also necessary to use more efficient algorithms. Future research will therefore aim at

- devising new or modifying existing algorithms to be more efficient in space and time
- implementing the algorithms in a more efficient way, especially with regard to computer memory requirements
- investigating a merge of symbolic with numeric computation for a hybrid solution of the problems
- solving more small and larger scale problems, to further guide in the selection of pressing lines of research.

To come back to the title of the note, at the moment we cannot deny nor confirm the unqualified viability of this approach. It is expected that within this decade the capability and efficiency of symbolic computation programs are enhanced and that the increasing computing power/cost ratio of computers will enable the solution of the benchmark and larger problems at reasonable costs.

ACKNOWLEDGMENT

H. van Essen implemented most of the algorithms in MAPLE. T. van de Broek supplied the vehicle simulation problem.

REFERENCES

- [1] R. Rothfuß, J. Schaffner, and M. Zeitz, "Rechnergestützte Analyse und Synthese nichtlinearer Systeme," in *Nichtlineare Regelung: Methoden, Werkzeugen, Anwendungen*, vol. 1026 of VDI Berichte. Düsseldorf: VDI-Verlag, 1993, pp. 267–291.
- [2] O. Akhrif and G. L. Blankenship, "Computer algebra algorithms for nonlinear control," in *Advanced Computing Concepts and Techniques in Control Engineering*, M. J. Denham and A. J. Laub, Eds. Berlin: Springer-Verlag, 1988, pp. 53–80.
- [3] B. de Jager, "Nonlinear control system analysis and design with Maple," in *Artificial Intelligence, Expert Systems and Symbolic Computing*, E. N. Houstis and J. R. Rice, Eds. Amsterdam: North-Holland, 1992, pp. 155–164.
- [4] H. van Essen and B. de Jager, "Analysis and design of nonlinear control systems with the symbolic computation system Maple," in *Proc. 2nd European Contr. Conf.*, Groningen, The Netherlands, vol. 4, June 1993, pp. 2081–2085.
- [5] A. Isidori, *Nonlinear Control Systems: An Introduction*, 2nd ed. Berlin: Springer-Verlag, 1989.
- [6] H. van Essen, "Symbols speak louder than numbers: Analysis and design of nonlinear control systems with the symbolic computation system MAPLE," Rep. WFW 92.061, Eindhoven Univ. Tech., Fac. of Mechanical Engineering, June 1992.
- [7] H. Nijmeijer and A. J. van der Schaft, *Nonlinear Dynamical Control Systems*. New York: Springer-Verlag, 1990.
- [8] H. B. Pacejka and E. Bakker, "The magic formula tyre model," in *Tyre Models for Vehicle Dynamics Analysis: Proc. 1st Int. Coll. on Tyre Models for Vehicle Dynamics Analysis*, Delft, The Netherlands, 1993, pp. 1–18.
- [9] B. de Jager, "Computer aided hybrid analysis and design of nonlinear control systems," in *Proc. 5th Int. Symp. on Application of Multivariable System Techniques*, Bradford, UK, Mar. 1994, pp. 247–254.

Note on Decentralized Adaptive Controller Design

Joon Lyoo

Abstract—This note extends the decentralized adaptive controller design of Gavel and Siljak [1] to the case when the relative order of each isolated subsystem does not exceed two.

I. INTRODUCTION

Recently, Gavel and Siljak [1] presented a stable adaptive decentralized design in the face of unknown interconnection strengths and under some structural conditions placed on the interconnections. As a main result, they suggested a decentralized high-gain feedback scheme for model reference adaptive control (MRAC) of an uncertain interconnected system which is composed of single-input single-output subsystems, and in each subsystem of which either incoming interactions enter through the control channel (Class I) or outgoing interactions pass through the measurement channel (Class 0). However, the design technique for the Class I has the disadvantage of disconnecting the interconnection signal from the input of the plant. And moreover, the main result is restricted to the case when the relative order of each subsystem (n_i^*) is equal to one (Case 1).

This note is concerned with developing a decentralized MRAC design for an uncertain interconnected system which belongs to the Class 0 and allows for the relaxed assumption that n_i^* is less than and equal to two. Since in case that $n_i^* = 2$ (Case 2), it is not possible to choose the reference model strictly positive real any longer, the Case 2 needs to be treated in a different way from the Case 1.

Manuscript received July 24, 1992; revised March 23, 1994. This work was supported in part by the Postdoctoral Program of the Korea Science and Engineering Foundation, Republic of Korea.

The author is with the Department of Electronic Engineering, Chungnam National University, Taejeon 305-764, Korea.

IEEE Log Number 9407014.