Symbolic analysis and design for (nonaffine) nonlinear control systems

13th Triennial World Congress, San Francisco, USA

2b-24 2

SYMBOLIC ANALYSIS AND DESIGN FOR (NONAFFINE) NONLINEAR CONTROL SYSTEMS

Bram de Jager *

* *Faculty of Mechanical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands. Email: A.G.de.Jager@wfw.wtb.tue.nl*

Abstract. For the analysis and design of dynamical systems symbolic tools are a great help. In the past, several algorithms for nonlinear control systems have been implemented in MAPLE. This implementation has been extended to also handle nonaffine models and to include more algorithms. Three examples taken from the literature in this area illustrate the power and limitations of the implementation. For system models that are more than a little complex symbolic computation cannot be fully enjoyed due to the complexity of the algorithms that are more than polynomial in some measure of the problem size.

Keywords. Nonlinear control systems, algorithms, linearization, robot control, flexible arms, induction motors, automotive control.

1. INTRODUCTION

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

In the past a closely tied set of symbolic computation functions, together called the NONLINCON package, has been presented that solves some problems in the analysis and design of nonlinear control system, see (Van Essen and De Jager, 1993; De Jager, 1993; De Jager, 1994*a*; De Jager, 1995*a*; De Jager, 1995*b*). The functions available in the package range from the computation of 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. All functions are based on constructive algorithms and implemented in MAPLE.

A model of the system must be available for the computation. The model, in general a set of nonlinear differential and algebraic equations in the state, input, and output of the system, was assumed to be affine in the input. For several problems appearing in practice this was too restrictive, see (De Jager, 1995*b*) for an example in vehicle dynamics, so it was decided to extend the NONLINCON package to alleviate this problem and some others.

Two approaches were used to make the package suitable for nonaffine models. First, several computations, *e.g.*, the relative degree (using a definition in (Nijmeijer and Van der Schaft, 1991)), the normal form, the zero dynamics, the exact linearization, and the input-output linearization, were enhanced to be able to gracefully handle nonaffine models. Secondly, an option was added to make a nonaffine model affine by extension of the state space and redefinition of its input.

Besides these enhancements to handle nonaffine models, the Dynamic Extension Algorithm was added as an option to give an affine model a (well-defined) relative degree, by state extension and feedback, in those cases it did not have one yet. Furthermore, the package was extended with a new and potentially more efficient algorithm to compute the local zero dynamics for affine models with a well-defined relative degree that was proposed in (Kwatny and Blankenship, 1994). Finally, a function was added to compute a feedback achieving disturbance decoupling for affine models.

Other packages are available. For an example, see (Blankenship *et al.*, 1995). Some recent additions to this field are in nonlinear $\mathcal{H}_\infty$ control (Møller-Pedersen and Petersen, 1995) and in discrete time nonlinear systems (Post, 1995).

The goal of this paper is to describe some of the enhancements to the NONLINCON package, to comment on several aspects of the implementation, and to provide some realistic examples of the use of the new facilities.

The following Section presents some theoretical background in the form of (comments on) algorithms. Section 3 provides several implementation details while Section 4 covers the examples. The paper closes with a discussion of the results and an indication of some directions for future research.

2. ALGORITHMS

The two standard models used are

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

for affine systems and

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

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 vector field, g representing a set of vector fields g_j (column-wise), and h is a column with stacked scalar functions h_i.

Modifications to two algorithms will be discussed (to compute the relative degree and normal form) that are useful in the case of nonaffine models. Also the Dynamic Extension Algorithm will be touched up on.

2.1 Relative degree

The relative degree of a nonaffine model of a nonlinear system in (x°, u°) is, based on the definition in (Nijmeijer and Van der Schaft, 1991, p. 417), the set of integers r_i, $i = 1, \ldots, p$ such that,

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

and the matrix

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

is nonsingular in (x°, u°), 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 $A_{\deg}$ is to be nonsingular is not posed in (Nijmeijer and Van der Schaft, 1991). The main differences with the definition for affine models is that there the term $\frac{\partial}{\partial u_j} L_f$ is replaced by the derivative along g_j: L_{g_j} and the conditions do not depend on u°.

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.

2.2 Dynamic extension

If a model does not have a well-defined relative degree, several tools or algorithms are not applicable anymore, restricting the analysis and design objectives that can be achieved for this model. To remedy this, a preliminary extension of the state space and state feedback are employed. The extension of the state space is necessary, because the relative degree is invariant under static state feedback, necessitating the use of dynamic state feedback.

The recursive algorithm presented in (Isidori, 1989, p 384), to which one is referred for more details, essentially identifies (combinations of) input channels that have to be "delayed" by an integrating action. The dynamic feedback from a single recursion step is

$$u = \alpha(x) + \beta_1(x)\zeta + \beta_2(x)v_2$$
$$\dot{\zeta} = v_1,$$

where v_1 is the set of "delayed" inputs and v_2 the set of inputs that are not delayed. The functions α, β_1, and β_2 are computed by the algorithm. The algorithm terminates if the relative degree becomes well-defined, or, when this is not possible, after a bounded number of steps, where the bound is at most n.

A restriction of the algorithm is that only square MIMO models are considered. Furthermore, additional states are introduced in several input-output channels, potentially introducing additional phase lag that may make the model more difficult to control. Due to the increase of the dimension of the state space the model becomes also less suited for symbolic computation: memory and compute time requirements may increase.

2.3 Normal form and zero dynamics

Normal forms for nonlinear models are useful for theoretical developments, because only a restricted class of models is to be considered, other classes can be transformed to the normal form class, but are also valuable in their own right, because they may highlight interesting characteristics of the model. One of those characteristics is the zero dynamics. Because the computations involved in obtaining a normal form are often rather tedious, it is of advantage to automate this.

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.

The original computation of the normal form, that was only suitable for affine models with a well-defined relative

degree, could easily be adapted to incorporate nonaffine models. It was also extended to affine models without a well-defined relative degree by building the "generalized" normal form proposed in (Isidori, 1989, p. 308), making use of the Zero Dynamics Algorithm (Isidori, 1989, p. 290).

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. Besides the computation of the zero dynamics by means of the normal form, another potentially more efficient algorithm to compute the local zero dynamics proposed in (Kwatny and Blankenship, 1994) was implemented. One should consult the original paper for the details of the algorithm, but we remark that it can only be used for affine models with a well-defined relative degree.

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 Zero Dynamics Algorithm, 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 first.

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 (Kwatny and Blankenship, 1994).

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

3. IMPLEMENTATION

The (modifications of) algorithms discussed in the previous section are rather straightforwardly implemented in MAPLE using the linear algebra facilities and the procedures to solve sets of nonlinear algebraic equations and sets of nonlinear partial differential equations (De Jager, 1994*b*).

One of the problems in the implementation stems from the determination or selection of a suitable solution, that is not unique, for an under-determined set of algebraic equations in the Dynamic Extension Algorithm, which is not gracefully handled by the standard MAPLE facility `solve`. A work around has been implemented.

4. EXAMPLES

The three examples to be presented use interesting models of electro-mechanical systems, *i.e.*, a two link robot with an elastic joint (an affine model without well-defined relative degree taken from (Isidori, 1989)), a voltage fed induction motor (a nonaffine model taken from (Nijmeijer and Van der Schaft, 1991)), and a four-wheel vehicle with steering (nonaffine and/or no well-defined relative degree) described in (De Jager, 1995*b*).

Example 1 The first example is a two link mechanism with an elastic joint between the two links. Its model is (see (Isidori, 1989, pp. 398–401))

$$f = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ f_4 \\ f_5 \\ f_6 \end{bmatrix}$$

with

$$\begin{aligned} f_4 l &= -6s_3(2x_4x_6 + x_6^2) + 2(1/2x_3 - 1/20x_2) \\ &\quad + (2 + 3c_3)(1/2x_2 - 5x_3 - 3s_3x_4^2) \\ f_5 l &= 6s_3(2x_4x_6 + x_6^2) + \frac{1}{4}(2 + 9c_3^2)(1/2x_3 - 1/20x_2) \\ &\quad - (2 + 3c_3)(1/2x_2 - 5x_3 - 3s_3x_4^2) \\ f_6 l &= 3(2 + 3c_3)s_3(2x_4x_6 + x_6^2) - (2 + 3c_3)(1/2x_3 - 1/20x_2) \\ &\quad - 3(-1 + 2c_3)(1/2x_2 - 5x_3 - 3s_3x_4^2) \end{aligned}$$

and

$$g = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -2/l & 2/l \\ 2/l & \frac{1}{4}(2 + 9c_3^2)/l \\ (2 + 3c_3)/l & -(2 + 3c_3)/l \end{bmatrix}$$

while the output is defined by

$$h = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix}$$

and the following abbreviations are used

$$\begin{aligned} l &= 10 + 9c_3^2 \\ c_3 &= \cos x_3 \\ s_3 &= \sin x_3. \end{aligned}$$

Here $x_1, \ldots, x_3$ represent the rotational coordinates of link 1, joint 2, and link 2 respectively, and $x_4, \ldots, x_6$ represent the velocities. For the parameters appearing in the original equations, integers have been filled in to simplify the computations by reducing the intermediate expression swell.

The following computation of the relative degree in the point $x^\circ = 0$ shows that it is not well-defined.

```
reldeg: the matrix Adeg turns out to be singular!

reldeg: the initial state conditions for which the
        matrix Adeg turns out to be singular are:
                        {all}
```

with

$$A_{deg} = \begin{bmatrix} -2/l & 2/l \\ (2 + 3c_3)/l & -(2 + 3c_3)/l \end{bmatrix}.$$

A well-defined relative degree is achieved with the Dynamic Extension Algorithm in two steps. This is illustrated with the following (condensed) output of the algorithm.

```
Step 1:

 alpha:[- 6 sin(x[3]) x[6] x[4] - 3 sin(x[3]) x[6]^2

        - 9/2 x[3] + 9/20 x[2] - 3 sin(x[3]) x[4]^2

        - 15/2 cos(x[3]) x[3] + 3/4 cos(x[3]) x[2]

        - 9/2 cos(x[3]) sin(x[3]) x[4]^2 ]
        [ ]
        [0]

        [                    2    ]
        [ - 5 - 9/2 cos(x[3])   1 ]
 beta : [                         ]
        [            0          1 ]

 reldeg: the matrix Adeg turns out to be singular!

        [          1           0 ]
 Adeg:  [                       ]
        [ - 1 - 3/2 cos(x[3])  0 ]

Step 2:
        [ 0 ]
 alpha: [   ]
        [ 0 ]

        [ 1  0 ]
 beta : [      ]
        [ 0  1 ]

 calculating Adeg

        [          1             0   ]
 Adeg:  [                           ]
        [ - 1 - 3/2 cos(x[3])  1/16 ]

 rdeg:  [ 4, 4 ]

 vector relative degree WELL-defined!
```

It is easily verified that the two sets of α and β that are computed, when combined in the correct way, agree with the result in (Isidori, 1989, p. 401) that the final dynamic state feedback (or dynamic extension) which gives the model a well-defined relative degree is

$$
\begin{aligned}
u_1 &= \frac{1}{g_{41}(x)}(-f_4(x) + \zeta) + v_2 \\
u_2 &= v_2 \\
\dot{\zeta} &= \xi \\
\dot{\xi} &= v_1.
\end{aligned}
$$

Example 2 This example considers a voltage fed induction motor. The model is taken from (Nijmeijer and Van der Schaft, 1991, pp. 267, 419), with a correction of a printing error. It is nonaffine and given by

$$
f = \begin{bmatrix}
-(\alpha+\beta)x_1 - \omega x_2 + \dfrac{\beta x_3}{L_s} + \dfrac{\omega x_4}{\sigma L_s} + \dfrac{u_1 \cos u_2}{\sigma L_s} \\
\omega x_1 - (\alpha+\beta)x_2 - \dfrac{\omega x_3}{\sigma L_s} + \dfrac{\beta x_4}{L_s} + \dfrac{u_1 \sin u_2}{\sigma L_s} \\
-\alpha\sigma L_s x_1 + u_1 \cos u_2 \\
-\alpha\sigma L_s x_2 + u_1 \sin u_2
\end{bmatrix} \tag{3}
$$

$$
h = \begin{bmatrix} x_3^2 + x_4^2 \\ x_2 x_3 - x_1 x_4 \end{bmatrix}.
$$

Here x_1 and x_2 represent components of the stator current, x_3 and x_4 are the corresponding flux components. The speed of the motor ω is considered constant, just as all the other model parameters. The total stator flux and stator torque are the outputs. The model does have a well-defined relative degree in $(x^\circ, u^\circ) = (0, 0, x_3^\circ, x_4^\circ, u_1^\circ, 0)$, but it is not full, so $\sum r_i < n$, as shown by

```
reldeg: the initial state conditions for which the
        matrix Adeg turns out to be singular are:

 {x[4] = 0, x[3] = x[1] sigma Ls}, {u[1] = 0},

          2       2
        x[4]  + x[3]  - x[3] sigma Ls x[1]
 {x[2] = ----------------------------------},
                x[4] sigma Ls

 {x[3] = 0, x[4] = 0}

vector relative degree = [1, 1]

total relative degree  = 2

 Adeg =  [ 2 (x[3] cos(u[2]) + x[4] sin(u[2])),

  + 2 u[1] (- x[3] sin(u[2]) + x[4] cos(u[2]))]

  [- x[4] cos(u[2])/(sigma Ls) + x[2] cos(u[2])

   + x[3] sin(u[2])/(sigma Ls) - x[1] sin(u[2]),

-u[1](- x[4] sin(u[2])/(sigma Ls) + x[2] sin(u[2])

      - x[3] cos(u[2])/(sigma Ls) + x[1] cos(u[2])]
```

This result is in agreement with the one given in (Nijmeijer and Van der Schaft, 1991, p. 419). Because the relative degree for this nonaffine model is well-defined, the zero dynamics can be computed, and because the model is nonaffine this is done by computing the normal form with the function `normform`. Selected results from the computation are the following.

```
zero dynamics:

 eta[1]dot = -(eta[2] alpha sigma Ls eta[1]

   + eta[2] eta[1] sigma Ls beta - omega eta[2]^2

   - RootOf(_Z^2 + eta[2]^2 ) sigma omega eta[1] Ls

   - RootOf(_Z^2 + eta[2]^2 ) sigma beta eta[2])

  / (eta[2] sigma Ls),

 eta[2]dot =

   alpha sigma Ls eta[1] RootOf(_Z^2 + eta[2]^2 )
  ------------------------------------------------
                      eta[2]

zeroing input uzero: [ 0, locuzero[2] ]
```

The interpretation of the last line of the result is simple. To keep the output $y = 0$ for all t, the input u_1 should be 0, and the value of input u_2 does not matter. This is easy to check from the differential equations (3). The input u_2 does only occur in combination with u_1. When $u_1 = 0$ the value of u_2 is irrelevant. The dynamics of the model under the constraint that $y = 0$ is given by the two differential equations for η.

The model can also be input-output linearized with the function `inoutlin`. The raw results of the IO linearization are voluminous. From the linearizing feedback only the second component of u is presented, as follows.

```
io linearizing feedback (ulin[2]):

arctan((- 2 v[2] sigma Ls x[3]

                    2   2
      + 2 x[2] sigma  Ls  x[3] alpha x[1]

             2      2   2
      + 2 x[2]  sigma  Ls  x[4] alpha

             2                                      3
      - 2 x[3]  x[2] sigma Ls beta - 2 omega x[3]

      + 2 x[4] x[1] sigma Ls beta x[3]

      + 2 x[4] omega x[2] sigma Ls x[3]

                   2
      - 2 omega x[4]  x[3]

             2
      - 2 x[4]  alpha sigma Ls x[2]

             2
      + 2 x[3]  omega x[1] sigma Ls

             2
      - 2 x[3]  alpha sigma Ls x[2]

      + x[2] sigma Ls v[1] - x[4] v[1])  /
                                         /
                                         /

          2      2   2
     (2 x[1]  sigma  Ls  x[3] alpha

                   2   2
      + 2 x[1] sigma  Ls  x[4] alpha x[2]

      + x[1] sigma Ls v[1] + 2 v[2] sigma Ls x[4]

             2
      - 2 x[3]  alpha sigma Ls x[1]

      + 2 x[3] x[2] sigma Ls beta x[4] - x[3] v[1]

      - 2 x[3] omega x[1] sigma Ls x[4]

             2
      - 2 x[4]  alpha sigma Ls x[1]

             2
      - 2 x[4]  x[1] sigma Ls beta

                   2
      - 2 x[4]  omega x[2] sigma Ls

                    2                      3
      + 2 omega x[3]  x[4] + 2 omega x[4] ))
```

Due to the complexity of the output, it is difficult to verify the result by simple inspection.

Example 3 This last example considers a vehicle steering stability problem. It is posed more extensively in (De Jager, 1995*b*). Refer to this for the complete model equations, that will not be repeated here. The original model is nonaffine. In the present research the model is simplified by assuming the lateral tire forces F_f and F_r, at front and rear, to be constant and not functions of the state and steering angle. The reason for this simplification is that the computation for the original model could not be completed. Several results will be discussed concerning the simplified, but still nonaffine, model given by the following equations

$$f = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ \dfrac{-F_f \sin(u_1 + x_3) - F_r \sin x_3 + u_2 \cos x_3}{M} \\ \dfrac{F_f \cos(u_1 + x_3) + F_r \cos x_3 + u_2 \sin x_3}{M} \\ \dfrac{aF_f \cos u_1 - bF_r}{J_t} \end{bmatrix}$$

$$h = \begin{bmatrix} x_4 \cos x_3 + x_5 \sin x_3 - U_d \\ (x_1 + p\cos x_3)^2 + (x_2 + p \sin x_3)^2 - R_d^2 \end{bmatrix},$$

where x_1, x_2 represent the position of the COM (center-of-mass) and x_3 the rotation around COM, the states $x_4, \ldots, x_6$ represent the corresponding velocities. The output y is defined in such a way that it is zero for a curving maneuver with constant radius R_d and constant forward speed component U_d. The input u_1 is the steering angle and u_2 is the traction force.

First, the relative degree of the model is computed. It is well-defined. See the output of the function `reldeg`.

```
vector relative degree = [ 1, 2 ]

total relative degree  = 3
```

The zero dynamics can be computed successfully, but it is complicated, making it tedious to derive its characteristics, *e.g.*, to decide if the zero dynamics is stable.

Second, state extension (an integrator in the first input channel) and input redefinition have been used to get a model that is affine. It appears that the affine model does not have a well-defined relative degree, so the Dynamic Extension Algorithm is applied, giving an affine model with well-defined relative degree, that is of higher order than the original one. The computation of the normal form for this extended model could not be completed due to intermediate expression swell. The zero dynamic could therefore not be computed.

The vehicle model presented in (Kwatny and Blankenship, 1994) has also been investigated. The algorithms to compute the zero dynamics were able to finish their computation for this model. So it appeared that it is much easier to handle.

The results of the first two examples are satisfactory. For the last example, which has a quite complicated model description, the results are disappointing due to

- the complexity of the algorithms,
- limitations of the computer algebra system.

Based on the MAPLE results for several examples, not all presented here, the following conclusions can be drawn.

- To make a model affine and to give it a well-defined relative degree are relatively simple problems, that can be solved rather easily by symbolic computation tools,
- making a nonaffine model affine is not always advantageous, the affine model of larger dimension may be more involved for computational purposes,
- the newly proposed algorithm to compute the zero dynamics is more efficient than the other two methods presented,
- the full vehicle zero dynamics problem can be solved by none of the algorithms presently available, model simplifications are mandatory.

5. DISCUSSION

The extensions to the NONLINCON package enable one to treat a larger class of models, including those that occur in practice for frequently encountered systems.

The extended NONLINCON package is successful for low order models that do not contain large intricate expressions. For large order models the package is less successful, among others due to an intrinsic limitation in MAPLE for the handling of large numbers of objects. To avoid this limitation, and so to fully exploit the power of symbolic computation, one can hardly avoid to become familiar with the peculiarities of the program used for implementation. Ideally one would want to avoid this. This should be improved.

Some other problems stem from the need to solve intricate sets of nonlinear ((partial) differential) equations. For these kind of problems - very difficult ones - no powerful, built-in, ready to use, functions are available. Therefore, some ad-hoc facilities were implemented to improve the capability of MAPLE in this respect, see, *e.g.*, (De Jager, 1994*b*). This should be improved also.

The main trouble lies, however, in the fact that the algorithms (and their implementations) are rather complex. While in numerical computations one is ready to accept a computational complexity growth rate that is a cube of the problem size, in symbolic computations one should be ready to accept a double exponential relation, see (De Jager, 1994*a*). This means that slightly larger problems than those used in this paper will readily become intractable, gathering dark clouds at the horizon for practical symbolic computation.

Acknowledgements

The extensions to NONLINCON were coded by Gosé Fisher (1994), starting from the code by Harm van Essen (1992).

6. REFERENCES

Blankenship, G. L., R. Ghanadan, H. G. Kwatny, C. LaVigna and V. Polyakov (1995). Tools for integrated modeling, design, and nonlinear control. *IEEE Control Systems Mag.* **15**(2), 65–79.

de Jager, Bram (1993). Zero dynamics and input-output exact linearization for systems without a relative degree using Maple. In: *Proc. of the 1993 Internat. Symp. on Nonlinear Theory and its Applications.* Vol. 4 IEICE. Honolulu, Hawaii. pp. 1205–1208.

de Jager, Bram (1994*a*). Computer aided hybrid analysis and design of nonlinear control systems. In: *Application of Multivariable System Techniques (AMST94)* (R. Whalley, Ed.) IMC. London: MEP. Bradford, UK. pp 247–254.

de Jager, Bram (1994*b*). Symbolic solution for a class of partial differential equations In: *Proc. of the Rhine Workshop on Computer Algebra* (Jacques Calmet, Ed.). Karlsruhe, Germany. pp. 84–88.

de Jager, Bram (1995*a*). Symbolics for control: Maple used in solving the exact linearization problem. In: *Computer Algebra in Industry 2: Problem Solving in Practice* (Arjeh M. Cohen, Leendert van Gastel and Sjoerd Verduyn Lunel, Eds.). pp. 291–311. John Wiley & Sons, Ltd.. Chichester.

de Jager, Bram (1995*b*). The use of symbolic computation in nonlinear control: Is it viable? *IEEE Trans. Automat. Control* **40**(1), 84–89.

Fisher, J. P. J. (Gosé) (1994). NONLINCON: Symbolic analysis and design package for nonlinear control systems. MSc thesis. Eindhoven University of Technology. Fac. of Mechanical Engineering. Report WFW 94.095.

Isidori, A. (1989). *Nonlinear Control Systems: An Introduction.* 2nd ed.. Springer-Verlag. Berlin.

Kwatny, H. G. and G. L. Blankenship (1994). 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.* IEEE. Benevento, Italy.

Møller-Pedersen, Jens and Martin Pagh Petersen (1995). Control of nonlinear plants - Volumes I and II. MSc thesis. Mathematical Institute, Technical University of Denmark. Lyngby, Denmark.

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

Post, B. A. (1995). Computer algebra analysis of nonlinear discrete-time systems. MSc thesis. Dept. of Applied Mathematics, University of Twente. Enschede, The Netherlands.

van Essen, H. (1992). Symbols speak louder than numbers: Analysis and design of nonlinear control systems with the symbolic computation system MAPLE. MSc thesis. Eindhoven University of Technology. Fac. of Mechanical Engineering. Report WFW 92.061.

van Essen, Harm and Bram de Jager (1993). 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.