Nonlinear control system analysis and design with Maple

References

[1] C. Brezinski, Accélération de la Convergence en Analyse Numérique (Springer, Berlin, 1977).

[2] C. Brezinski, Algorithmes d'Accélération de la Convergence - Étude Numérique (Éditions Technip, Paris, 1978).

[3] J. Wimp, Sequence Transformations and Their Applications (Academic, New York, 1981).

[4] J-P. Delahaye, Sequence Transformations (Springer, Berlin, 1988).

[5] E. J. Weniger, Comput. Phys. Rep. 10 (1989) 189.

[6] D. Shanks, J. Math. and Phys. (Cambridge, Mass.) 34 (1955) 1.

[7] P. Wynn, Math. Tables Aids Comput. 10 (1956) 91.

[8] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, S. M. Watt, Maple V Language Reference Manual (Springer, New York, 1991).

[9] L. F. Richardson, Phil. Trans. Roy. Soc. London A 226 (1927) 229.

[10] P. Wynn, Proc. Camb. Phil. Soc. 52 (1956) 633.

[11] C. Brezinski, C. R. Acad. Sc. Paris 273 (1971) 727.

[12] B. Germain-Bonne, Estimation de la Limite des Suites et Formalisation de Précédes d'Accélération de Convergence (Thesè, Université de Lille, 1978).

[13] D. Levin, Int. J. Comput. Math. B3 (1973) 371.

[14] A. Sidi, J. Comput. Appl. Math. 7 (1981) 37.

[15] C. Brezinski, Numer. Math. 35 (1980) 176.

[16] T. Håvie, BIT 19 (1979) 204.

[17] J. Grotendorst, Comput. Phys. Commun. 59 (1990) 289.

[18] A. C. Aitken, Proc. Roy. Soc. Edinburgh 46 (1926) 289.

[19] R. Bhattacharya, D. Roy and S. Bhowmick, Comput. Phys. Commun. 55 (1989) 197.

[20] T. Fessler, W. F. Ford and D. A. Smith, ACM Trans. Math. Software 9 (1983) 346.

[21] E. J. Weniger and E. O. Steinborn, in M. Defranceschi and J. Delhalle (eds.), Numerical determination of the electronic structure of atoms, diatomic and polyatomic molecules (Kluwer, Dordrecht, 1989), pp. 341-346.

[22] J. Grotendorst, Comput. Phys. Commun. 67 (1991) 325

Artificial Intelligence, Expert Systems and Symbolic Computing
E.N. Houstis and J.R. Rice (Editors)
Elsevier Science Publishers B.V. (North-Holland)

Nonlinear Control System Analysis and Design with Maple

Bram de Jager

Department of Mechanical Engineering, WH 2.137, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.
Email: jag@wfw.wtb.tue.nl

Abstract
For the analysis and design of nonlinear control systems non-numerical methods are available. The required analytical computations are mostly too tedious to be done error free in a reasonable time by hand, so the use of symbolic computation programs can be of advantage. To show that the symbolic computation program MAPLE can be used to solve simple but representative analysis and design problems, four "benchmark" cases are used. Problems encountered are discussed. Recommendations for the enhancement of MAPLE are given.

1 INTRODUCTION

To fulfill the increasing requirements on the dynamic behavior of systems, it is necessary to control the system. For the controller design and implementation a model of the system is used. A nonlinear model is often necessary to describe the system adequately. Models can be characterized by structural properties. For the design of controllers these structural properties can be used to advantage. Among the interesting properties of a nonlinear model, from the viewpoint of control system design, are

- controllability and observability,
- the relative degree,
- the zero dynamics of the model.

Their computation for linear models is straightforward. For nonlinear models a numerical computation usually only gives an approximation of the properties of the model (along a trajectory or in a working point).

Standard methods for analysis and design of linear control systems are based on linear algebra packages like LINPACK, EISPACK and LAPACK, or interactive programs like MATLAB and MATRIXx. Analysis for nonlinear systems mostly means simulation, i.e., numerically solving the nonlinear differential equations representing the system. Design is almost always done for a linearized system. This is not adequate for the design of high performance control systems, e.g., fast and accurate robot control, or for structural problems in nonlinear system control and identification.

In the literature several methods are proposed for the exact solution of analysis and design problems in nonlinear control [1, 2]. For small scale test problems these methods already require a fair amount of computation. For industrial scale problems these methods can only be applied when the computations are assisted by SC (Symbolic Computation) programs. These programs are, however, not specifically designed for this kind of problems, so it is not clear beforehand

whether the current crop of SC programs can successfully be used for these problems. Even if a program can be used, but requires a lot of user programming, it is not of much value.

For the analysis and design of control systems SC programs must provide facilities for

- computation of differential operators, i.e., differentiation and multiplication of functions, such as polynomials, rational functions and transcendentals,
- rank determination,
- matrix inversion and multiplication,
- solving partial differential equations,
- solving sets of nonlinear equations.

The use of the SC program MACSYMA for implementing algorithms for controller and observer analysis and design problems for nonlinear control systems is reported by [3, 4, 5, 6]. REDUCE has also been used [7]. Both SC programs have some definite disadvantages. The programs are based on Lisp, which is quite uncommon in the control engineering community. This introduces problems when the programs have to be used by students and industry. Also, MACSYMA is known for its bulky resource requirements. Our experiences with REDUCE are that it is not straightforward to program user functions. Therefore alternative SC programs are investigated. Among the programmable SC programs MAPLE and MATHEMATICA seems to be the most promising. Here we used MAPLE. The reasons for this choice are: availability of source code, easy programming, low run times, and previous experience.

To assess the suitability of the SC program MAPLE, test cases are worked out. Four cases are presented, which were solved with MAPLE. Preliminary results are reported in [8, 9].

The first case illustrates the analysis problem of local observability, the second the design of a linearizing state-feedback for robotic systems, the third the design of nonlinear observers. Finally, the last case discusses the computation of the zero dynamics. The experiences with these cases lead to some suggestions.

2 LOCAL OBSERVABILITY

Given the model of an autonomous nonlinear dynamic system

$$\dot{x} = f(x), \qquad y = h(x) \tag{1}$$

with state $x(t) \in \mathbb{R}^n$, output $y(t) \in \mathbb{R}^l$, and appropriate vector functions $f \in C_n^\infty$ and $h \in C_l^\infty$. Roughly stated, the problem is whether knowledge of the output $y(\tau)$ for $0 < \tau < t$, permits the determination of the state $x(t)$, $t > 0$. A way to check this, is to verify whether the $nl \times n$ matrix

$$\begin{bmatrix} \frac{\partial}{\partial x} h(x) \\ \frac{\partial}{\partial x} L_f h(x) \\ \cdots \\ \frac{\partial}{\partial x} L_f^{n-1} h(x) \end{bmatrix}, \tag{2}$$

with the Lie derivative $L_f h(x)$ defined as $L_f h(x) = \frac{\partial h(x)}{\partial x} f(x)$ and $L_f^i h(x)$ defined recursively, has full rank n [10]. Then (1) is locally observable.

A MAPLE procedure `observ` is written, based on (2). Local observability is tested in a generic way, i.e., for lower dimensional manifolds in the state space local observability can be hampered. Initially this procedure used the standard MAPLE version 4.3 function `rank` which is based on the procedure `ffgausselim`. In `ffgausselim` the function `divide` is used to check whether an entry of the matrix is zero. This setup caused some problems:

- the memory and time usage were large,
- due to "intermediate expression blow up" some problems could not be solved (due to segmentation fault errors, caused by pointer overflows),
- the entries in (2) could not be rational functions (in MAPLE V the `gausselim` procedure works for rational functions), so, initially, all entries in a row were multiplied with their smallest common denominator, which made the blow up problem worse.

To remedy those problems, we modified the `ffgausselim` procedure. In `observ` the rank of a submatrix of (2) is checked, another relevant row is added to the submatrix and the rank is checked again. The modified procedure `gaussrow` takes advantage of this and only tries to eliminate one additional row. Doing this, the rank is checked of matrices with maximal dimension $n \times n$, much smaller than (2). Also, instead of the `divide` function, the `normal` or `expand` functions are used, so rational functions are permitted. This influenced the efficiency of the procedure (see Table 1). The computations performed in `observ` are based on procedures for

- computation of differential operators,
- rank determination.

One of the problems solved with `observ` stems from an extended kalman filter problem in biochemics [11]. The equations for the system are

$$\dot{x} = \begin{bmatrix} \frac{-x_8 x_1 x_2}{x_1 + k_s} \\ \frac{(x_6 - x_7) x_9 x_4 x_2}{(6 + \frac{1}{4} a_1 x_6)(x_4 + k_o)} + \frac{x_7 x_8 x_1 x_2}{x_1 + k_s} \\ \frac{-(2 + \frac{1}{2} a_1 x_7) x_9 x_4 x_2}{(6 + \frac{1}{4} a_1 x_6)(x_4 + k_o)} + \frac{(2 + \frac{1}{2} a_1 x_7) x_8 x_1 x_2}{x_1 + k_s} \\ \frac{-x_9 x_4 x_2}{x_4 + k_o} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3}$$

$$y = \begin{bmatrix} x_5 \\ x_4 \\ x_3 \\ \frac{-x_9 x_4 x_2 x_5}{x_4 + k_o} \\ \frac{(4 - x_6 + \frac{1}{6} a_2 x_7) x_9 x_4 x_2 x_5}{(6 + \frac{1}{4} a_1 x_6)(x_4 + k_o)} + \frac{(2 - \frac{1}{6} a_2 x_7) x_8 x_1 x_2 x_5}{x_1 + k_s} \end{bmatrix}. \tag{4}$$

This system is locally observable in a generic sense. In principle, it is possible to check this numerically. Choose a specific state x, linearize (3) and (4) and check the rank of the observability matrix. Due to numerical inaccuracies, this gives an incorrect answer when a standard rank function, based on the singular value decomposition, is used. The procedure `observ` gives the correct answer.

function	memory [MB]	time [s]
`gaussrow`	3.7	157
`gausselim`	7.3	478
`gaussrow & divide`	> 24.0	> 9000

Table 1: Memory and time needed for test case.

To show the difference in efficiency between several versions of `observ`, performance results are presented in Table 1, MAPLE V on a Sun 4/280 with 64 MB RAM was used, by permission

of CAN (Computer Algebra in the Netherlands). The left column shows the procedures used to perform gaussian elimination. Both `gaussrow` and `gausselim` use the function `normal` to check if an entry is zero. If in `gaussrow` the `divide` function is used, as in `ffgausselim`, the procedure `observ` halts after 9000 [s] with the message `system error, object too large`, when a large part of core memory (> 24 MB) is already allocated.

Remark 1: It is easy to extend the `observ` procedure to a non-generic observability test, by checking if and when the pivots used in the gauss elimination part of the rank determination vanish. This requires the solution of an, in general nonlinear, algebraic equation and determines a manifold in the n dimensional state space.

Remark 2: As suggested by M. B. Monagan, a procedure based on a "modulo a prime" rank determination will require less time and memory. Disadvantages of such a procedure are

- it is a probabilistic method,
- it is only applicable for a specific state,
- it cannot be extended to a non-generic observability test.

3 LINEARIZING STATE-FEEDBACK

Freund [12] proposes a method to control robotic systems. This method gives a linear decoupled controlled system, with poles which can be arbitrarily placed, at least when there is a perfect model. To illustrate the computations which are performed, a sketch of a simplified version of the design algorithm follows. See [12] for further details. The starting point is a constant model of a robotic system, given by

$$\dot{x} = f(x) + g(x)u, \qquad y = h(x) \tag{5}$$

with state $x(t) \in \mathbb{R}^{2m}$, input $u(t)$ and output $y(t) \in \mathbb{R}^m$, while $f \in C^\infty_{2m}$, $h \in C^\infty_m$ are vector functions and $g \in C^\infty_{2m\times m}$ is a matrix function. In contrast to (1) this model is not autonomous. The aim is to give the system a desired dynamic behavior by selecting a suitable $u(t)$. Therefore the following quantities are computed

$$D(x) = \begin{bmatrix} \frac{\partial}{\partial x} L_f h_1(x) \\ \cdots \\ \frac{\partial}{\partial x} L_f h_m(x) \end{bmatrix} g(x)$$

$$C(x) = \begin{bmatrix} L_f^2 h_1(x) \\ \cdots \\ L_f^2 h_m(x) \end{bmatrix}$$

$$M(x) = \begin{bmatrix} \alpha_{01} h_1(x) + \alpha_{11} L_f h_1(x) \\ \cdots \\ \alpha_{0m} h_m(x) + \alpha_{1m} L_f h_m(x) \end{bmatrix}.$$

Using these quantities, the control input vector is given by

$$\begin{bmatrix} u_1(t) \\ \cdots \\ u_m(t) \end{bmatrix} = -D^{-1}(x) \left\{ C(x) + M(x) - \Lambda \begin{bmatrix} w_1(t) \\ \cdots \\ w_m(t) \end{bmatrix} \right\}. \tag{6}$$

Substitution of (6) in (5) results in a linear system with the new input vector $w(t) \in \mathbb{R}^m$. The control design parameters α_{ij} and the $m \times m$ diagonal matrix Λ should be selected to obtain the desired dynamics of the controlled system. Normally, the designer aims for a stable system with small tracking errors and large stability robustness. The computations required for this design algorithm are

- differentiation,
- matrix inversion,
- matrix vector multiplication,
- simplification of expressions.

The algorithm has been implemented as a short MAPLE procedure. The only input required is a model of the robotic system, the MAPLE procedure handles the complete design. It is applied on the two examples in [12]. The results are the same. The only problem encountered is the simplification of the resulting control input $u(t)$, which is difficult to automate. The control input $u(t)$ is therefore not always obtained in a form appreciated by a human being.

4 NONLINEAR OBSERVER

Keller [13] proposes a method for the design of nonlinear observers, based on a pole placement procedure for a linearized observer error equation. This method has already been implemented in MACSYMA [4] and partially in REDUCE [7]. It is also partially implemented in the MAPLE procedure `obsdesign`. The part of the design method which could not be fully implemented (or perhaps only after more user programming) is to find the solution of a partial differential equation. The problem area is shown in the following.

The system considered is of the form (5) but now with state $x(t) \in \mathbb{R}^n$, input $u(t) \in \mathbb{R}^m$ and output $y(t) \in \mathbb{R}$. It differs from (5) in that the output y is scalar and the number n of states x is not twice the number m of inputs u. Again, the functions f, g and h are sufficiently differentiable and of appropriate dimensions. Under some conditions the system can be transformed to

$$\dot{\bar{x}} = \bar{f}(\bar{x}) + \bar{g}(\bar{x})u, \qquad y = \bar{x}_1, \tag{7}$$

where $\bar{f}$ and $\bar{g}$ have a special form, with the inverse of the nonlinear transformation

$$\bar{x} = \begin{bmatrix} h(x) \\ L_f h(x) \\ \cdots \\ L_f^{n-1} h(x) \end{bmatrix}. \tag{8}$$

In the canonical form (7), the output equation is quite simple. To obtain the equations of the linearized observer error dynamics, further transformations are necessary. For the appropriate mapping functions, the partial differential equation

$$\bar{L}_{\bar{f}}^n c(y) + \bar{L}_{\bar{f}}^{n-1} a_{n-1}(y) + \cdots + \bar{L}_{\bar{f}} a_1(y) + a_0(y) = 0 \tag{9}$$

has to be solved for $c(y)$ and $a_i(y)$, $i = 0, \ldots, n-1$. A solution for $n = 2$ is given in [13] and has been implemented in `obsdesign`. A general solution of (9), if it exists, can in general not be computed with MAPLE. This is a pity, because (9) is a typical form of a partial differential equation, often appearing in analysis and design problems for nonlinear control systems. Other computations to be performed are the computation of the inverse of (8), to obtain (7). Therefore, the symbolic computation program should be capable of

- computation of differential operators,
- solving a specific partial differential equation,
- solving sets of nonlinear equations (for the inverse mappings).

The `obsdesign` procedure is used for the problem in [13]. The result agrees with the results reported in [13, 4, 7].

5 THE ZERO DYNAMICS

As stated in the introduction, the zero dynamics of a dynamic system gives information which is useful for the design of controllers for the system. For instance, the linear equivalent of a model with asymptotically stable zero dynamics is a minimum phase model.

One of the first uses of the zero dynamics concept is in [14] and the minimum phase property is used recently in [15, 16]. This recent activity in the development of nonlinear control theory indicates the importance of the zero dynamics concept, and therefore of its computation for practical systems.

The theoretical foundation of the computation of the zero dynamics is given in, e.g., [1, 2]. Here, we discuss the computation of the zero dynamics of nonlinear models, with emphasis on the implementation of the algorithms within MAPLE.

First, we give an overview of the underlying theory, followed by some remarks about the actual implementation of the algorithms in the ZERODYN procedure. Then the results are given for a case, representative for small scale industrial applications, that we solved with ZERODYN.

5.1 Theory of the zero dynamics

The theory in [1] is the basis for the following discussion. The *zero dynamics* of the nonlinear model (using the notation of [1])

$$\dot{x}(t) = f(x(t)) + \sum_{i=1}^{m} g_i(x(t))u_i(t) \quad \text{with} \quad x \in \mathbb{R}^n,\ u_i \in \mathbb{R}$$

and

$$y(t) = h(x(t)) \quad \text{with} \quad y \in \mathbb{R}^l$$

is the dynamics of the submodel, which results after choosing the inputs $u_i(t)$ and initial conditions x° such that the outputs $y(t)$ of the model are 0 for all t. Alternatively, the zero dynamics is the dynamics of the submodel of maximal dimension that can be made unobservable by state-feedback.

For simplicity the remaining discussion is restricted to a single-input ($m = 1$), single-output ($l = 1$) model

$$\dot{x} = f(x) + g(x)u, \quad y = h(x). \tag{10}$$

The computation of the zero dynamics of this model at $x = x^\circ$ consists of two steps

- bring the model in a *local normal form* with a nonlinear invertible change of coordinates $z = \Phi(x)$,
- extract the zero dynamics equations from this form.

First, compute r components of Φ as

$$\begin{aligned} \phi_1(x) &= h(x) \\ \phi_2(x) &= L_f h(x) \\ &\cdots \\ \phi_r(x) &= L_f^{r-1} h(x) \end{aligned}$$

where r is the *relative degree* at x°, i.e., the smallest r for which

- $L_g L_f^k h(x) = 0, \quad \forall x$ around $x^\circ, \quad \forall k < r-1$,
- $L_g L_f^{r-1} h(x^\circ) \neq 0$.

When there is no r fulfilling these conditions, r is sometimes defined as $r = \infty$, and the normal form cannot be derived.

Next, choose the remaining $n - r$ new coordinates z_i, $i = r+1, \ldots, n$ such that

$$\begin{aligned} &(1)\ \ \Phi(x) \text{ is invertible at } x^\circ \\ &(2)\ \ L_g z_i = L_g \phi_i(x) = 0, \quad \forall x \text{ around } x^\circ, \quad \forall r+1 \leq i \leq n. \end{aligned} \tag{11}$$

This is alway possible [2], but it is sometimes difficult to compute the required Φ.

The normal form in the new coordinates z is then particularly simple

$$\begin{aligned} \dot{z}_1 &= z_2 \\ &\cdots \\ \dot{z}_{r-1} &= z_r \\ \dot{z}_r &= b(z) + a(z)u \\ \dot{z}_{r+1} &= q_{r+1}(z) \\ &\cdots \\ \dot{z}_n &= q_n(z) \\ y &= z_1 \end{aligned}$$

where $b(x) = L_f^r h(x)$, $a(x) = L_g L_f^{r-1} h(x)$ and $q_i(x) = L_f \phi_i(x)$, $i = r+1, \ldots, n$. Use the relation

$$x = \Phi^{-1}(z) \tag{12}$$

to express $b(x)$, $a(x)$ and $q_i(x)$, $i = r+1, \ldots, n$ as functions of z.

When it is too tedious to compute the components of Φ fulfilling (11), choose ϕ_i, $i = r+1, \ldots, n$ such that Φ is invertible, then the normal form will be as above, but the $\dot{z}_i$, $i = r+1, \ldots, n$ will not only depend on $q_i(z)$, but are also affine in u, like $\dot{z}_r$.

To get $y = 0$, choose the input u as $-b(z)/a(z)$ and the initial conditions x° so that $z_i = 0$, $i = 1, \ldots, r$. By definition, the zero dynamics of the model is then given by the equations for $\dot{z}_{r+1}, \ldots, \dot{z}_n$, with the coordinates $z_1, \ldots, z_r$ set to 0. With $\xi = [z_1, \ldots, z_r]'$, $\eta = [z_{r+1}, \ldots, z_n]'$ the zero dynamics can be written as

$$\dot{\eta} = \begin{bmatrix} q_{r+1}(0, \eta) \\ \cdots \\ q_n(0, \eta) \end{bmatrix}.$$

For multivariable models the theory is more involved.

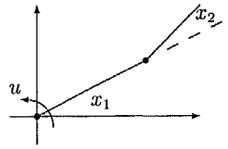
5.2 Computation of the zero dynamics

The solution of the partial differential equations (11) and of the system of nonlinear equations (12) is the main problem in the computation of the zero dynamics. General symbolic computation programs like MAPLE do not always find a solution, although theoretically a computable solution exists. The class of problems for which a solution can be obtained is therefore limited. The availability of a program that can tackle more problems will be welcomed.

The actual code of the MAPLE procedure ZERODYN is a straightforward implementation of the algorithm discussed in [1] and the previous theory, although not for the general multivariable case. We do not present it here. Work to expand the ZERODYN procedure to compute the zero dynamics for square multivariable systems is in progress.

5.3 Application

The analysis procedure ZERODYN is applied to several models. The model presented here is a mechanical system with two rotary joints [1], the first driven by a motor and with a spring for the second joint.



The equations of state for this model are

$$f(x) = \begin{bmatrix} x_3 \\ x_4 \\ \frac{bc(x_3+x_4)^2 \sin x_2 + (b+c\cos x_2)Kx_2 + c^2 x_3^2 \sin x_2 \cos x_2}{d-(c\cos x_2)^2} \\ -\frac{(b+c\cos x_2)(x_4+2x_3)cx_4 \sin x_2 + (a+2c\cos x_2)(cx_3^2 \sin x_2 + Kx_2)}{d-(c\cos x_2)^2} \end{bmatrix}$$

$$g(x) = \begin{bmatrix} 0 \\ 0 \\ \frac{b}{d-(c\cos x_2)^2} \\ \frac{-(b+c\cos x_2)}{d-(c\cos x_2)^2} \end{bmatrix}$$

$$h(x) = x_1.$$

The model parameters are a, b, c, d and K, with $d - c^2 > 0$ and

$$d = b(a - b). \tag{13}$$

The relative degree r is 2. The standard MAPLE function `dsolve` was unable to solve the partial differential equation (11). A solution for ϕ_3, ϕ_4 is computed by an extended version of `dsolve` which first rewrites (11) as a set of ordinary differential equations and recursively solves this set, resulting in the following map Φ, invertible for all x°

$$\left\{ z_1 = x_1, z_2 = x_3, z_3 = x_2, z_4 = x_4 + (1 + \frac{c}{b}\cos x_2)x_3 \right\}.$$

The inverse map Φ^{-1} is

$$\{x_1 = z_1, x_2 = z_3, x_3 = z_2, x_4 = -(z_2 b - z_4 b + z_2 c \cos z_3)/b\}.$$

The normal form of the system is

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= b(z) + a(z)u \\ \dot{z}_3 &= q_3(z) \\ \dot{z}_4 &= q_4(z) \end{aligned}$$

with

$$a(z) = b/(d - (c\cos z_3)^2)$$

$$b(z) = \{bz_2^2 c \sin z_3 - 2z_2 c \sin z_3 (z_2 b - z_4 b + z_2 c \cos z_3) + \frac{c \sin z_3 (z_2 b - z_4 b + z_2 c \cos z_3)^2}{b} + Kz_3 b + Kz_3 c \cos z_3 + z_2^2 c^2 \sin z_3 \cos z_3\}/\{d - (c\cos z_3)^2\}$$

$$q_3(z) = -(z_2 b - z_4 b + z_2 c \cos z_3)/b$$

$$q_4(z) = \{\frac{z_2 c \sin z_3 (z_2 b - z_4 b + z_2 c \cos z_3)(d - (c\cos z_3)^2)}{b} + b^2 z_2^2 c \sin z_3 + z_2^2 (c\cos z_3)^2 c \sin z_3 - baz_2^2 c \sin z_3 - Kz_3 ba + Kz_3 b^2 + Kz_3 (c\cos z_3)^2\}/\{b(d - (c\cos z_3)^2)\}.$$

The zero dynamics is now given by the equations for q_3 and q_4 with z_1 and z_2 set to 0. Using the relation between the model parameters (13) we obtain

$$\begin{aligned} \dot{z}_3 &= q_3(0, 0, z_3, z_4) = z_4 \\ \dot{z}_4 &= q_4(0, 0, z_3, z_4) = \frac{-Kz_3}{b}. \end{aligned}$$

This is a nice small expression and it agrees with the result in [1]. Further analysis shows that the zero dynamics is not asymptotically stable.

6 CONCLUSIONS

The general symbolic algebra program MAPLE can be used as a starting point or first attempt for the implementation of algorithms for problems in the analysis and design of nonlinear control systems. The observability and controller design cases, requiring only explicit and recursive computations, are solvable by MAPLE. Initial problems with the efficiency of the `rank` function are solved by modifying standard MAPLE procedures. Implicit and non-recursive computations like solving sets of nonlinear equations or partial differential equations cannot always be performed by MAPLE, although solutions are known to exist. Human intervention may therefore be needed, especially when the model is complex, but precisely for those models we need the assistance of a SC program.

So, it is necessary that in the nearby future some limitations of MAPLE are removed. In particular

- enlarge the structure size to prevent pointer overflows,
- make the simplification functions more powerful to avoid human intervention in the simplification process,
- enlarge the class of sets of nonlinear equations that can be solved,
- enhance the capabilities for the solution of partial differential equations.

The availability of a symbolic algebra program with more powerful capabilities for solving sets of nonlinear equations and, especially, for solving partial differential equations should make it possible to tackle more problems.

Acknowledgements

The author likes to thank Harm van Essen for his implementation of the extended version of `dsolve`.

References

[1] A. Isidori, "Lectures on nonlinear control," course notes, Università di Roma "La Sapienza", Rome, July 1987.

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

[3] O. Akhrif and G. L. Blankenship, "Computer algebra for analysis and design of nonlinear control systems," in *Proceedings of the 1987 American Control Conference*, vol. 1, (Minneapolis, MN), pp. 547–554, IEEE, 1987.

[4] H. Bär, H. Fritz, and M. Zeitz, "Rechnergestützter Entwurf nichtlinearer Beobachter mit Hilfe einer symbolverarbeitenden Programmiersprache," *Automatisierungstechnik*, vol. 35, pp. 177–183, May 1987.

[5] J. Birk and M. Zeitz, "Computer-aided design of nonlinear observers," in *Nonlinear Control Systems Design: science papers of the IFAC Symposium*, (Capri, Italy), pp. 1–6, IFAC, Pergamon Press, Oxford, 1989.

[6] J. Birk and M. Zeitz, "Anwendung eines symbolverarbeitenden Programmsystems zur Analyse und Synthese von Beobachtern für nichtlineare Systeme," *Messen, Steuern, Regeln*, vol. 33, pp. 536–543, Dec. 1990.

[7] T. Mimpen, "Design of a nonlinear observer with REDUCE," trainee project, Eindhoven University of Technology, Department of Mechanical Engineering, Aug. 1990. Report WFW 90.028. In Dutch.

[8] B. de Jager, "Some nonlinear control system analysis and design with Maple," in *Proceedings of the 13th World Congress on Computation and Applied Mathematics, IMACS '91* (R. Vichnevetsky and J. J. H. Miller, eds.), vol. 1, (Dublin, Ireland), pp. 105–106, IMACS, July 1991.

[9] B. de Jager, "Symbolic calculation of zero dynamics for nonlinear control systems," in *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation, ISSAC '91* (S. M. Watt, ed.), (Bonn, Germany), pp. 321–322, ACM Press, New York, July 1991.

[10] J. Birk and M. Zeitz, "Extended Luenberger observer for non-linear multivariable systems," *Internat. J. Control*, vol. 47, pp. 1823–1836, June 1988.

[11] P. Bögels, "Modelling and model reference adaptive control for fed-batch bakers' yeast fermentation," tech. rep., Unilever Research Laboratory, Vlaardingen, May 1989. Restricted circulation.

[12] E. Freund, "Fast nonlinear control with arbitrary pole-placement for industrial robots and manipulators," *Internat. J. Robotics Res.*, vol. 1, pp. 65–78, Spring 1982.

[13] H. Keller, "Entwurf nichtlinearer, zeitvarianter Beobachter durch Polvorgabe mit Hilfe einer Zwei-Schritt-Transformation," *Automatisierungstechnik*, vol. 34, pp. 271–275, 326–331, July and Aug. 1986.

[14] C. I. Byrnes and A. Isidori, "A frequency domain philosophy for nonlinear systems, with applications to stabilization and to adaptive control," in *Proceedings of the 23rd IEEE Conference on Decision and Control*, vol. 3, (Las Vegas, NV), pp. 1569–1573, IEEE, Dec. 1988.

[15] C. I. Byrnes and A. Isidori, "Asymptotic stabilization of minimum phase nonlinear systems," *IEEE Trans. Automat. Control*, vol. 36, pp. 1122–1137, Oct. 1991.

[16] C. I. Byrnes, A. Isidori, and J. C. Willems, "Passivity, feedback equivalence, and the global stabilization of minimum phase nonlinear systems," *IEEE Trans. Automat. Control*, vol. 36, pp. 1228–1240, Nov. 1991.

Artificial Intelligence, Expert Systems and Symbolic Computing
E.N. Houstis and J.R. Rice (Editors)
Elsevier Science Publishers B.V. (North-Holland)

Some Applications of Maple to Mathematical, Scientific and Engineering Problems

T.C. Scott[a], M.B. Monagan[b], G.J. Fee[c] and R.M. Corless[d]

[a]Institute for Theoretical Atomic and Molecular Physics, Harvard-Smithsonian Center for Astrophysics, 60 Garden Street, Cambridge MA, U.S.A.

[b]Institute for Scientific Computation, ETH-Zentrum, CH 8092 Zurich, Switzerland

[c]Symbolic Computation Group, Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada

[d]Department of Applied Mathematics, University of Western Ontario, London, Ontario N6A 5B7, Canada

Abstract

In this article, we survey some applications of Maple[1] where it has proved useful as a problem-solving tool. The application areas include relativity, quantum theory, audio engineering, number theory, and asbestos fiber analysis. They should give the reader some idea of the potential capabilities and versatility of the Maple system. Most of the applications presented here include new approaches at the forefront of research. For the reader new to the field of symbolic computation, we hope to show how the symbolic computation tools available in Maple have provided new possibilities in various fields of research.

1. INTRODUCTION

It is impossible to present in this article an exhaustive survey of all the cases where the Maple system has proven itself useful as a problem-solver. Rather, we highlight some of the projects associated with the authors and consequently present only a sample of what can be done with symbolic computation. However, the ranges of applications presented here should give the reader some idea of the potential capabilities and versatility of the Maple system. The applications discussed here include past and current work with some hints at future projects. These are categorized according to their respective disciplines below.

2. PURE MATHEMATICS - The First Case of Fermat's Last Theorem

In about 1637, Fermat asserted, in the margin of his copy of the complete works of Diophantus, that it is not possible to find for a given integer $n > 2$ nonzero integers x, y, z such that

$$x^n + y^n = z^n.$$