SRC TR 87-52

Computer Algebra for Analysis and Design of Nonlinear Control Systems

by

O. Akhrif, and G.L. Blankenship

COMPUTER ALGEBRA FOR ANALYSIS AND DESIGN OF NONLINEAR CONTROL SYSTEMS

O. Akhrif G. L. Blankenship *

Electrical Engineering Department & Systems Research Center
University of Maryland, College Park, Maryland 20742

Abstract

A rich collection of analytical tools based on differential geometric methods has been developed for the analysis and design of nonlinear control systems. The concept of feedback equivalence among nonlinear systems is used to linearize and control certain classes of nonlinear control systems. The left and right invertibility of nonlinear systems is used to solve the output tracking problem. Using computer algebra programming methods, a software system has been developed which makes these analytical procedures available to users who need not have an extensive knowledge of differential geometry. This work may be viewed as a component of an expert system for the treatment of stochastic and deterministic linear and nonlinear control problems. Examples of the use of this system are reported.

1. Introduction

The main object of our work is the creation of a general-purpose tool for analysis and design of nonlinear control systems in the form of a software system using computer algebra and symbolic manipulations.

In the first part of this report we review some methods for analysis of deterministic servo-problems for nonlinear systems affine in control, i.e, systems which in local coordinates are described by:

$$\frac{dx}{dt} = f(x) + \sum_{i=1}^{m} u_i(t) g_i(x) \tag{1}$$

In recent years, the differential geometric approach to nonlinear control problems has been developing fast. The differential geometric setting allows the generalization of many known classical results in linear systems theory to the nonlinear case. For the purpose of dynamic control of nonlinear affine systems (output tracking, stabilization ...), we will use two concepts from differential geometric system theory. The first is the concept of *feedback equivalence* among nonlinear systems. The second is the concept of *left-invertibility* of nonlinear control systems.

Feedback equivalence is an equivalence relation (transitive, symmetric and reflexive) among systems and it generalizes the concept of linear feedback group which plays a role in linear system theory (*Wonham* [12]) leading, among other things, to the Brunovsky canonical form and the definition of controllability indices.

* This research supported in part by NSF Grant CDR-85-00108

The second concept used is the concept of *invertibility* of nonlinear systems. A control system is invertible when the corresponding input-output map is injective. If this is the case, then it is possible to reconstruct uniquely the input acting on the system from knowledge of the corresponding output. The main application of this is in the output tracking problem where we try to control a system so that its output follows some desired path. The inverse system is used to generate the required control given the desired output function.

In the second part of the report, we present CONDENS, a software package that implements these two concepts for nonlinear control systems using a differential geometric approach.

Equipped with the theoretical concepts presented above, this system can, given a nonlinear system affine in control, answer questions such as: Is this system feedback-equivalent to a controllable linear one? If so, can we construct the diffeomorphism that makes this equivalence explicit? Is the nonlinear system invertible? What is the relative order of this system? In case the system is invertible, can we construct a left-inverse for it? Given a real analytic function $y(t)$, can $y(t)$ appear as output of the nonlinear system? If so, what is the required control?

CONDENS tries to answer these questions and uses this knowledge to solve a design problem: The output tracking problem for the given nonlinear control system. If we are interested in simulation results, the system can automatically generate numerical programs for that purpose.

In Section 2, the concepts of state equivalence and feedback equivalence among systems are defined and necessary and sufficient conditions for state and feedback equivalence to linear controllable systems are recalled from the literature. The construction of state and feedback transformations is also reviewed. We then show how we can use feedback linearization to design an output tracking controller for the nonlinear system (1). This design scheme was first proposed by *G.Meyer* at NASA Ames Research Center [9]. He used the feedback transformations of nonlinear control systems to Brunovsky canonical form in the design of model following automatic pilots for vertical and short take-off aircraft. One important aspect of this design is that all regulation is done on the canonical form and the regulator never sees the more complicated original system.

In Section 3, necessary and sufficient conditions for the invertibility of nonlinear control systems of the form $\dot{x} = f(x) + \sum_{i=1}^{m} g_i(x)u_i, y = h(x)$ are given. The relative order of the system is defined and a prefilter or left-inverse system is constructed. These results are used to study the question of right-invertibility or functional controllability for nonlinear systems where the problem is to determine functions $f(t)$ which can be realized as the outputs of the nonlinear system driven by a suitable input function. The left and right invertibility is used to design an output tracking controller for our nonlinear system. The left-inverse, if given as input the desired trajectory, will naturally generate the required control.

In Section 4, we present all the programming work. We show how the system can make available to the design engineer some design methods that rely on more or less sophisticated mathematical tools. A brief description of the different modules of the program is given. In the last section, we present some examples with simulation results to illustrate the performance of the system.

Acknowledgement: We would like to thank C.I. Byrnes for suggesting this research area and J.P. Quadrat for introducing us to the potential of MACSYMA.

2. Feedback Equivalence among nonlinear systems

Krener (1973), *Brockett* (1978), *Jacubczyck* and *Respondek* (1980), *Hunt et al* (1983) and *Su* (1982) introduced the concept of feedback linearization or feedback equivalence for nonlinear systems and characterized via necessary and sufficient conditions, systems which are feedback equivalent to linear controllable ones. This so-called feedback linearization is based on three operations: Change of coordinates in the state space, change of coordinates in the control space and feedback.

Definition

The nonlinear system affine in control:

$$\dot{x} = f(x) + \sum_{i=1}^{m} g_i(x)u_i \qquad x \in R^n \tag{2}$$

is said to be *feedback linearizable* in a neighborhood U of the origin ($f(0) = 0$) if there exists a transformation (T, S) (called feedback or F-transformation) consisting of:

- A diffeomorphism $T = (T_1, \ldots, T_n) : U \longrightarrow T(U)$, $T(0) = 0$ and $z = T(x)$, (state space change of coordinates).
- State feedback and affine change of coordinates of the control space R^m over the ring of smooth functions $C^\infty(U)$, $S = (S_1, \ldots, S_m) : U \times R^m \longrightarrow R^m$, where $v = S(x, u) = \alpha(x) + \beta(x)u$.

  and the $m \times m$ matrix $[\frac{\delta S_i}{\delta u_j}(x)]$ invertible for all $x \in U$.

such that in the new coordinates (z, v), the nonlinear system (2) becomes a controllable linear system in Brunovsky canonical form:

$$\dot{z} = Az + Bv \qquad z \in R^n \tag{3}$$

with *Kronecker* indices $k_1, \ldots, k_m$.

The *Kronecker* indices k_i play an important role since, being invariant under feedback transformations (see [6]), characterize the equivalence classes of systems that are feedback equivalent. The algorithm for the computation of these indices was proposed by *Hunt-Su* [4] and is very similar to the one known for linear systems.

The following result (*Hunt, Su* and *Meyer* 1983) gives necessary and sufficient conditions under which a transformation of the type we consider exists.

Theorem 1 (Hunt-Su [5]) *A multi-input control system $\dot{x} = f(x) + \sum_{i=1}^{m} g_i(x)u_i$ is F-transformable to a controllable linear system in Brunovsky form in a neighborhood of the origin in R^n if and only if:*

1. *The set $C = \{g_1, [f, g_1], \ldots, ad_f^{k_1-1}(g_1), g_2, [f, g_2], \ldots, ad_f^{k_2-1}(g_2), \ldots, g_m, [f, g_m], \ldots, ad_f^{k_m-1}(g_m)\}$ spans an n-dimensional space.*
2. *The sets $C_j = \{g_1, [f, g_1], \ldots, ad_f^{k_j-2}(g_1), g_2, [f, g_2], \ldots, ad_f^{k_j-2}(g_2), \ldots, g_m, [f, g_m], \ldots, (ad_f^{k_j-2}(g_m)\}$ are involutive for $j = 1, 2, \ldots, m$ and*

3. The span of each C_j is equal to the span of $C_j \cap C$.

where $k_1 \geq k_2 \geq \cdots \geq k_m$ are the Kronecker indices of both the nonlinear system and the linear system.

We remark that block triangular systems (*Meyer* and *Cicolani* 1980) are an interesting subset of the class of all systems satisfying the hypotheses of this theorem. Indeed, to construct an F-transformation for such systems is an easy task. In the single input case, *Brockett* (1978) considered only feedbacks of the form $v = u + \alpha(x)$. This is an interesting case since it facilitates the construction of the F-transformation but it makes the conditions of transformability more restrictive.

In the most general case, from the fact that only change of coordinates and feedback are allowed, the components of the transformation (T, S) satisfy the following set of first order partial differential equations:

$$\begin{aligned} \prec dT_1, ad_f^i(g_j) \succ &= 0 \qquad i = 0, 1, \ldots, k_1 - 2 \\ \prec dT_{\sigma_1+1}, ad_f^i(g_j) \succ &= 0 \qquad i = 0, 1, \ldots, k_2 - 2 \\ &\vdots \\ \prec dT_{\sigma_{m-1}+1}, ad_f^i(g_j) &= 0 \qquad i = 0, 1, \ldots, k_m - 2 \end{aligned} \tag{4}$$

where $j = 1, \ldots, m$
and $\sigma_1 = k_1, \sigma_2 = k_1 + k_2, \ldots, \sigma_m = k_1 + \cdots + k_m = n$

As one can see, solution of these equations is not always easy or even possible. In [4], *Hunt* and *Su* gave a procedure to solve this set of first order partial differential equations by solving n sets of n ordinary differential equations where the solution of each system depends on the solution of the previous system. This algorithm is implemented in CONDENS.

The other components are found by:

$$\begin{aligned} T_{i+1} &= \prec dT_i, f \succ \\ &\quad i = 1, \ldots, \sigma_1 - 1, \sigma_1 + 1, \ldots, \sigma_{m-1} - 1, \\ &\quad \sigma_{m-1} + 1, \ldots, \sigma_m - 1 = n - 1 \\ S_1 &= \prec dT_{\sigma_1}, f + \sum_{i=1}^{m} u_i g_i \succ \\ &\vdots \\ S_m &= \prec dT_{\sigma_m}, f + \sum_{i=1}^{m} u_i g_i \succ \end{aligned}$$

Application: Design of output tracking controllers

The design technique is to build a controller for the nonlinear system by designing one for the equivalent linear canonical system. The design proceeds in three steps. First, the given nonlinear system is transformed into a constant, decoupled, controllable linear representation. Second, standard linear design techniques, such as pole placement, are used to design an output tracking control for this simple representation, forcing the output to follow some desired trajectory. Third, the resulting control law is transformed back out into the original coordinates to obtain the control law in terms of the available controls.

G.Meyer at NASA Ames Research Center [9] proposed this scheme in the design of exact model following automatic pilots for vertical and short take-off aircraft and has applied it to several aircraft of increasing complexity.

3. Invertibility of nonlinear systems

In this section, we review some of the results on left and right invertibility for nonlinear systems that are implemented in our software system. Even though CONDENS handles the multivariable case, only results for single input single output case will be reported for simplicity (see [13] for the multivariable case).

We consider systems of the following form:

$$\begin{aligned} \dot{x}(t) &= f(x(t)) + u(t)g(x(t)) \qquad x(0) = x_0 \in M \\ y(t) &= h(x(t)) \end{aligned} \tag{5}$$

where the state space M is a connected real analytic manifold, f, g are analytic vector fields on M, h is a real analytic mapping, and $u \in U$, the class of real analytic functions from $[0, \infty)$ into R. If $x_0 \in M$ and u is an admissible control, we denote the resulting solution to the above differential equation by $x(t, u, x_0)$ and denote $h(x(t, u, x_0))$ by $y(t, u, x_0)$.

The left-invertibility problem is the problem of injectivity of the input-output map of system (5). If the input-output map is invertible from the left, then it is possible to reconstruct uniquely the input acting on the system from knowledge of the corresponding output. Since, as we know, the input-output map of a nonlinear system depends on the initial state x_0, one has to incorporate the dependence on the initial state into a precise definition of invertibility.

Definition

The nonlinear system (5) is said to be ***left-invertible at*** $x_0 \in M$ if whenever u and $\hat{u}$ are distinct admissible controls,

$$y(t, u, x_0) \neq y(t, \hat{u}, x_0)$$

for at least a value of $t \geq 0$.

The system (5) is ***strongly invertible at*** $x_0 \in M$ if there exists an open neighborhood V of x_0 such that for all $x \in V$, the system is left-invertible at x.

The system (5) is ***strongly invertible*** if there exists an open and dense submanifold M_0 of M such that for all $x_0 \in M_0$ the system is strongly invertible at x_0.

If a system is strongly invertible, it is natural to look for a second system which acts as a left-inverse for the original system. The inverse system is a nonlinear system which, when driven by appropriate derivatives of $y(., u, x_0)$ produces $u(.)$ as its output. The left-inverse provides a practical method for determining u, and has many applications, e.g, the tracking problem.

Before stating the theorem that gives the necessary and sufficient conditions of strong invertibility of system (5), let us define the relative order α of system (5) as defined by ***Hirschorn*** in [3].

Definition

The ***relative order*** α of the nonlinear system (5) is the first nonnegative integer k such that $L_g L_f^{k-1} h \not\equiv 0$ on M and $L_g L_f^j h \equiv 0 \quad \forall\, 0 \leq j < k-1$.

or $\alpha = \infty$ if $L_g L_f^k h \equiv 0 \quad \forall k \geq 0$.

Theorem 2 (Hirschorn [3]) *The nonlinear system (5) is strongly invertible if and only if* $\alpha < \infty$.

To see how the relative order of the system is related to its strong invertibility, we have to remember that our aim is to solve for the control $u(t)$ as a function of the state and the output $y(t)$.

To solve the output equation $y(t) = h(x(t))$ for u, it will be necessary to differentiate y:

$$\begin{aligned} \frac{dy}{dt} &= \frac{\partial h}{\partial x}\dot{x} \\ &= L_f h(x) + L_g h(x).u(t) \end{aligned}$$

If $L_g h \not\equiv 0$ then $\alpha = 1$.
If $L_g h \equiv 0$ then we get :

$$\frac{dy}{dt} = L_f h(x)$$

therefore we should differentiate once more, we can go on like this until we reach the relative order α, we obtain then:

$$\frac{d^\alpha y}{dt^\alpha} = L_f^\alpha h(x) + L_f L_g^{\alpha-1} h(x).u(t)$$

where $L_f L_g^{\alpha-1} h$ is $\not\equiv 0$.

Let then $M_\alpha = \{x \in M / L_f L_g^{\alpha-1} h(x) \neq 0\}$.

M_α is called *the inverse submanifold* of the system. Because of the analyticity of the function $L_f L_g^{\alpha-1} h$, M_α will be an open dense subset of M, hence a submanifold of M. M_α will provide the state space for the left-inverse system.

Therefore, suppose that the nonlinear system (5) is strongly invertible with relative order α, initial state $x_0 \in M$, and inverse submanifold M_α, then the system:

$$\begin{aligned} \dot{z} &= F(z) + G(z)v \qquad z(0) = x_0 \\ w &= H(z) + K(z)v \end{aligned} \tag{6}$$

where $z \in M_\alpha$,

$$K(z) = \frac{1}{L_f L_g^{\alpha-1} h(z)} \qquad H(z) = -\frac{L_f^\alpha h(z)}{L_f L_g^{\alpha-1} h(z)}$$

$$F(z) = f(z) - \frac{g(z) L_f^\alpha h(z)}{L_f L_g^{\alpha-1} h(z)} \qquad G(z) = \frac{g(z)}{L_f L_g^{\alpha-1} h(z)}$$

acts as a *left-inverse* for the original system (5).

The problem of right-invertibility is the problem of determining functions $f(t)$ which can be realized as output of the nonlinear system (5) driven by a suitable input function. While the left-invertibility is related to the *injectivity* of Input/Output map of the nonlinear system (5), the right-invertibility is related to the *surjectivity* of the Input/Output map.

Theorem 3 (Hirschorn [3]) *Consider the nonlinear system (5) with relative order α.*

If $\alpha < \infty, x_0 \in M_\alpha$ and $f \in C^w(R)$ then there exists $u \in U$ such that $y(.,u,x_0) = f(.)$ if and only if :

$$f^{(k)}(0) = L_f^k h(x_0) \qquad \text{for } k = 0, 1, \ldots, \alpha - 1$$

Application: Design of output tracking controllers

It is easy to see how the notion of left and right invertibility can be applied to the output tracking problem. The application of the inversion algorithm to the nonlinear system (see [13] for the algorithm in the multivarible case) gives rise to a left-inverse system which, when driven by appropriate derivatives of the output y, produces $u(.)$ as its output. The question of trajectory following by the output is related to right-invertibility of the nonlinear input-output map, and the ability of the nonlinear system to reproduce the reference path as its output. To obtain robustness in the control system under perturbations, design of a servocompensator around the inner loop using servomechanism theory is suggested.

4. CONDENS: A software package using symbolic manipulations

In this section, CONDENS, a software package using symbolic manipulations is presented. The main aim of CONDENS is to implement the theory presented in previous chapters and help the user in some symbolic calculations in differential geometry and its applications to control, with emphasis on the design of controllers for the output tracking problem.

The impact of the computer on mathematics and its related fields is well-known. Perhaps, less well-known is the recent progress of the application of symbolic calculations in the more continuous parts of mathematics; such as mathematical analysis, differential equations, differential geometry and its applications in nonlinear control theory.

An obvious but nonnegligible consequence is the fact that using symbolic calculations, one may carry out easy, "long and tedious" calculations with the computer, thus avoiding elementary mistakes, such as wrong signs, missing brackets, omitted symbols,...etc.

CONDENS presents an other feature: It can automatically generate numerical programs for the solution of problems posed in symbolic form or for simulation purposes. Given a new control problem, the time involved in analyzing the theoretical results and then writing the Fortran code to execute it is eliminated. The mistakes and the time required to test and debug the Fortran code is also eliminated.

Most importantly, the system allows the engineer to interact with the computer for design at the symbolic manipulation level. In this way, he can modify his analysis or design problem by modifying the symbolic functional form of the model. The Fortran subroutines that he might have to modify by hand to accomplish this in conventional design procedures are written automatically for him.

There are several systems designed for symbolic calculations, e.g, FORMAC, MACSYMA, REDUCE. We have chosen the language MACSYMA as a basis for our developments. It may be implemented on any computer system supporting LISP, it is easily available and, consequently, widespread. Last, but not least, we are charmed by the interactive facilities of MACSYMA.

Description of CONDENS

Based on the theoretical concepts presented in previous chapters, the system, can treat feedback linearization and tracking control problems for certain classes of nonlinear systems.

The package contains a set of user-defined functions collected in a special file ("initfile.mac") with file addresses for all the functions. Once this file is loaded into MACSYMA, a user-defined function, not previously defined, will be automatically loaded into MACSYMA when it is called. This will be done by a login file using the `setup-autoload` command. So the first thing the user has to do once in MACSYMA, is to load the login file (load "initfile.mac";). He can then start the package using the command `condens()`; which will give him some hints on how to use CONDENS.

The programs that form CONDENS can be considered by order of complexity, in three different categories.

First we have a set of user-defined functions that perform some differential geometric computations. Straightforward computations such as *Lie brackets* and *Lie derivatives* and more complex computations such as *Kronecker indices* or *Relative order* of nonlinear systems are included.

MENU: returns a list of all the user-defined functions contained in CONDENS.

HELP(fun-name): generates an information text describing the function, its syntax, how to enter its arguments as well as an example.

JACOB(f): computes the *Jacobian* matrix of f.

LIE(f,g): computes the *Lie brackets* of the vector fields f and g:

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

ADJ(f,g,k): computes the k $\overset{th}{-}$ adjoint of f and g:

$$\begin{aligned} ad_f^k g &= [f, ad_f^{k-1} g] \\ ad_f^0 g &= g \end{aligned}$$

LIDEV(f,h): computes the *Lie derivative* of the real valued function h along the direction defined by the vector field f: $\mathbf{L_f h}$.

NLIDEV(f,h,k): computes the k^{th} *Lie derivative* of h along f:

$$\begin{aligned} L_f^k h &= L_f \left(L_f^{k-1} h \right) \\ L_f^0 h &= h \end{aligned}$$

KRONECK(f,g): used for multivariable systems,where g represents the set of vector fields $g_1, \ldots, g_m$. This function is useful when the nonlinear system is transformable to a linear controllable system canonical form. It computes the *Kronecker indices* of the equivalent linear system the original nonlinear system is transformed to. It returns a set of numbers: $k_1 \geq k_2 \geq \cdots \geq k_m$ with $k_1 + k_2 + \cdots + k_m = n$.

BTRIANG(f,g): This function checks if the nonlinear system $\dot{x} = f(x) + \sum_{i=1}^{m} g_i(x)u_i$ is in *block triangular form* . The argument g of the function represents the m vector fields $g_1, \ldots, g_m$.

RELORD(f,g,h): computes the *relative order* (see section 3 for definition) of the single-input single-output nonlinear system (5).

Second, we present two more sophisticated modules TRANSFORM and INVERT that use the user-defined functions presented above two theoretical problems: feedback linearization and invertibility of nonlinear control systems.

TRANSFORM (f,g): treats the feedback linearization problem presented in Section 2, that is, it investigates the existence of a one to one transformation (consisting of a change of coordinates and feedback) which transforms system $\dot{x} = f(x) + \sum_{i=1}^{m} g_i(x)u_i$ to a controllable linear system in canonical form. Moreover, in case the transformation, say T, exists, TRANSFORM generates the set of partial differential equations that T satisfies. In some cases, the module TRANSFORM can solve these partial differential equations and returns the transformation T and its inverse.

INVERT(f,g,h): Given the nonlinear control system:

$$\begin{aligned} \dot{x} &= f(x) + \sum_{i=1}^{m} g_i(x)u_i \\ y &= h(x) \end{aligned} \tag{7}$$

INVERT tries to check if this system is strongly invertible by investigating the relative order of the system. In case it is invertible (that is in case the relative order is finite), INVERT returns this relative order, furthermore, it computes the left-inverse to system (7) and returns: $A_{inv}(x), B_{inv}(x), C_{inv}(x)$,
$D_{inv}(x)$ where:

$$\begin{aligned} \dot{x} &= A_{inv}(x) + B_{inv}(x)\hat{y} \\ u &= C_{inv}(x) + D_{inv}(x)\hat{y} \end{aligned}$$

Third, we have FENOLS and TRACKS which are two design packages contained in CONDENS. Their purpose is essentially to use, respectively, the two modules TRANSFORM and INVERT to design a control law that forces the output of a nonlinear system to follow some desired trajectory.

They are both user-friendly, ask progressively for all the data they need, and are capable of automatically generating, upon request, Fortran codes for simulation purposes.

FENOLS:

- Takes as input the nonlinear dynamics $f, g_1, \ldots, g_m$, the desired path $x_d(t)$ and the desired eigenvalues for the linear regulator.
- Uses the package TRANSFORM to check if the system is transformable to a controllable linear one, and constructs the nonsingular transformation T and its inverse T^{-1}.

- Once the canonical linear system is obtained, FENOLS uses the desired eigenvalues (entered as inputs) to design by the pole placement method a linear feedback control. This control solves the output tracking problem for the equivalent linear system.
- In case the user is interested in simulations results, all he has to do is answer "yes" to a question posed by FENOLS at the end: "Are you interested in simulation results?"

TRACKS:

- Takes as input the nonlinear dynamics $f, g_1, \ldots, g_m, h_1, \ldots, h_m$ and the desired trajectory $y_d(t)$.
- Investigates the left and right invertibility of the nonlinear system.

  Left-invertibility: by computing the relative order of the system and making sure that it is finite.

  Right-invertibility: by checking if the desired trajectory $y_d(t)$ is trackable by the system, that is, if $y_d(t)$ satisfies the conditions of Theorem 3.
- Uses the package INVERT to construct the left-inverse to the system.
- Feeds the left-inverse system obtained with $y_d(t)$ to obtain, as output, the required control $u_d(t)$.
- Generates upon request a Fortran program for simulation purposes.

5. Examples

In this section, two examples are treated using CONDENS.

EXAMPLE 1: Basic industrial robot

This is the example of a *basic industrial robot.* It has one rotational joint and a translational joint in the (x, y) plane.

Using a polar coordinate system for modelling the robot, the kinetic equations are:

$$\begin{aligned} \ddot{r} &= r\dot{\varphi}^2 - \frac{m_R l}{2(m_R+m_L)}\varphi^2 + \frac{K_r}{m_R+m_L} \\ \ddot{\varphi} &= \frac{(-2(m_R-m_L)r + ml)\dot{r}\dot{\varphi} + M_\varphi}{k - m_R l r + (m_R+m_L)r^2} \\ y_1 &= r \qquad y_2 = \varphi \end{aligned}$$

with $r(t)$ as translational motion and $\varphi(t)$ as the angle of rotation. y_1 and y_2 are the outputs that have to be controlled. $K_r(t)$ and M_φ are the drives corresponding to $r(t)$ and $\varphi(t)$.

If the state variables and the inputs are chosen to be:

$$\begin{aligned} x_1(t) &= r(t) & x_2(t) &= \dot{r}(t) \\ x_3(t) &= \varphi(t) & x_4(t) &= \dot{\varphi}(t) \\ u_1(t) &= K_r(t) & u_2(t) &= M_\varphi(t) \end{aligned}$$

then the system can be described by a state space representation of the form:

$$\begin{aligned}
\dot{x_1} &= x_2 \\
\dot{x_2} &= x_1 x_4^2 - \frac{m_R l}{2(m_R + m_L)} x_4^2 + \frac{u_1}{m_R + m_L} \\
\dot{x_3} &= x_4 \\
\dot{x_4} &= \frac{-2[(m_R + m_L)x_1 - m_R \frac{l}{2}]x_2 x_4 + u_2}{k - m_R l x_1 + (m_R + m_L)x_1^2}
\end{aligned}$$

We are interested in designing a tracking controller to track the trajectories:

$$\begin{aligned}
y_1(t) &= x_1(t) = exp(t) \\
y_2(t) &= x_3(t) = t
\end{aligned}$$

Using the first technique: Exact Linearization.

with $m_R = m_L = l = 2, k = 5$

Using the first technique: Exact Linearization.

with $m_R = m_L = l = 2, k = 5$

```
(c2) load("initfile.mac");

(d2)                                initfile.mac

(c3) fenols();

Hello,FENOLS tries to solve the problem:                                    #

      Given the non linear system:

                                  m
                                 ====
                          dx     \
                          -- =    >     u  g (x) + f(x)
                          dt     /       i  i
                                 ====
                                 i = 1

find a non singular transformation that takes this system to a controllable li#

near system:

                                  dz
                                  -- = b . v + a . z
                                  dt

enter dimension of the state space
4;

enter dimension of the control space
2;

enter the values of f in the form[f1(x),f2(x),.....,fn(x)] followed by ,
[x2,x1*x4**2-x4**2/2,x4,(4*x2*x4-8*x1*x2*x4)/(4*x1**2-4*x1+5)];

enter g (x) in the form [ g     (x) .... g     (x) ]
       1               1, 1           1, 4
[0,1,0,0];

enter g (x) in the form [ g     (x) .... g     (x) ]
       2               1, 1           1, 4
[0,0,0,4/(4*x1**2-4*x1+5)];

Hello,TRANSFORM tries to solve the problem:                                  #

        Given the non linear system:

                          dx
                          -- = f(x) + u  g (x) + u  g (x)
                          dt           2  2       1  1

find a non singular transformation that takes this system to a controllable

linear system:

                                  dz
                                  -- = b . v + a . z
                                  dt

checking if the system is in block triangular form....

system in block triangular form

the transformation is easy to construct
```

The new state variables are:

z[1]= x1

z[2]= x3

z[3]= x2

z[4]= x4

The new control variables are:

$$v[1] = x1\ x4^2 - \frac{x4^2}{2} + u1$$

$$v[2] = \frac{4\ x2\ x4 - 8\ x1\ x2\ x4}{4\ x1^2 - 4\ x1 + 5} + \frac{4\ u2}{4\ x1^2 - 4\ x1 + 5}$$

enter the desired trajectory in the form [xd(1),...xd(n)]
[exp(t),exp(t),t,1];

enter the desired eigenvalues of the linear controller in the form [delta(1),.#

..,delta(n)] followed by a ,
[-2,-2,-2,-2];

The tracking controller is :

$$u(x,xd)[1] = -\frac{(2\ x1 - 1)\ x4^2 - 2\ (-4\ x3 - 4\ x1 + 5\ \%e^t + 4\ t)}{2}$$

$$u(x,xd)[2] = ((8\ x1\ x2 - 4\ x2)\ x4 + 4\ x1^2\ (-4\ x4 - 4\ x2 + 4\ \%e^t + 5) - 4\ x1\ (-4\ x4 - 4\ x2 + 4\ \%e^t + 5) + 5\ (-4\ x4 - 4\ x2 + 4\ \%e^t + 5))/4$$

Are you interested in simulation results?(answer y or n)
y;

enter filename of fortran code(without adding'.f')
exampl;

enter initial time you would like the simulation to start from
0.0;

enter final time tf
5.0;

enter step size h
0.01;

enter initial condition in the form[xo[1],...xo[n]]
[0,0,0,0];

```
      dimension x( 4 ),dx( 4 ),datad(1000, 4 ),data(1000,
     1 4 ),u( 2 ),y( 4 )
c     set no of equations
      n= 4
      m= 2
c     set initial conditions
      x(1) = 0
      x(2) = 0
      x(3) = 0
      x(4) = 0
      y(1) = x(1)
      y(2) = x(2)
      y(3) = x(3)
      y(4) = x(4)
c     set initial & final time
      t= 0.0
      tf= 5.0
c     set step size
      h= 0.01
```

```
c     desired trajectory
      no= 500
      do 15 i=1,no
      datad(i,1) = exp(t)
      datad(i,2) = exp(t)
      datad(i,3) = t
      datad(i,4) = 1
      t=t+h
 15   continue
c     store initial values for plotting
      do 25 i=1, 4
      data(1,i)=y(i)
 25   continue
      t=0.0
      call control(x,t,u)
c     initialize k & mm
      k=0
      mm=1
      print *,t,datad(mm,1),data(mm,1)
c     write down the differential equations
 1    n= 4
      dx(1) = x(2)


      dx(2) = x(1)*x(4)**2-x(4)**2/2.0+u(1)
      dx(3) = x(4)
      dx(4) = (4*x(2)*x(4)-8*x(1)*x(2)*x(4))/(4*x(1)**2-4*x(1)+5)+4*u(2)
     1   /(4*x(1)**2-4*x(1)+5)
      call runta(n,k,ii,x,dx,t,h,u)
      go to (1,2),ii
 2    mm=mm+1
      y( 1 )= x(1)
      y( 2 )= x(2)
      y( 3 )= x(3)
      y( 4 )= x(4)
      do 30 i=1, 4
      data(mm,i)=y(i)
 30   continue
      print *,t,datad(mm,1),data(mm,1)
      if(t.le.tf) go to 1
      stop
      end

      subroutine runta(n,k,ii,x,dx,t,h,u)
      dimension yf( 4 ),z( 4 ),x( 4 ),dx( 4 ),u( 2 )
      k=k+1
      go to (1,2,3,4,5),k
 2    do 10 j=1,n
      z(j)=dx(j)
      yf(j)=x(j)
 10   x(j)=yf(j)+0.5*h*dx(j)
 25   t=t+0.5*h
      call control(x,t,u)
 1    ii=1
      return
 3    do 15 j=1,n
      z(j)=z(j)+2.0*dx(j)
 15   x(j)=yf(j)+0.5*h*dx(j)
      call control(x,t,u)
      ii=1
      return
 4    do 20 j=1,n
      z(j)=z(j)+2.0*dx(j)
 20   x(j)=yf(j)+h*dx(j)
      go to 25
 5    do 30 j=1,n
 30   x(j)=yf(j)+(z(j)+dx(j))*h/6.0
      call control(x,t,u)
      ii=2
      k=0
      return
      end

      subroutine control(x,t,u)
      dimension u( 2 ),x( 4 )
      u(1) = -((2*x(1)-1)*x(4)**2-2*(5*exp(t)+4*t-4*x(3)-4*x(1)))/2.0
      u(2) = (4*x(1)**2*(4*exp(t)-4*x(4)-4*x(2)+5)-4*x(1)*(4*exp(t)-4*x(
     1   4)-4*x(2)+5)+5*(4*exp(t)-4*x(4)-4*x(2)+5)+(8*x(1)*x(2)-4*x(2))*
     2   x(4))/4.0
      return
      end

(d3)                          done
```

Using the second technique: Inverse System.

```
(c2) load("initfile.mac");

(d2)                                initfile.mac

(c3) tracks();

Hello,TRACKS tries to solve the problem:                                    #

        Given the non linear system:

                                    m
                                   ====
                              dx   \
                              -- = >     u  g (x) + f(x)
                              dt   /      i  i
                                   ====
                                   i = 1


                                 y(t)=h(x)

where      x is an n-dim vector
           u is an m-dim vector
           y is an m-dim vector

find the inverse system that takes as input the desired trajectory yd(t) and  #

generates the required control ud(t)

enter dimension of the state space
4;

enter dimension of the control space
2;

enter the values of f in the form [ f (x) ... f (x) ] followed by ,
                                    1          4
[x2,x1*x4**2-x4**2/2,x4,(4*x2*x4-8*x1*x2*x4)/(4*x1**2-4*x1+5)];

enter g (x) in the form [ g     (x) .... g     (x) ] followed by ,
       1                  1, 1           1, 4
[0,1,0,0];

enter g (x) in the form [ g     (x) .... g     (x) ] followed by ,
       2                  2, 1           2, 4
[0,0,0,4/(4*x1**2-4*x1+5)];

enter the values of h in the form [ h (x) ... h (x) ] followed by ,
                                    1          2
[x1,x3];

enter the initial state x0 in the form [ x0 (x) ... x0 (x) followed by ,
                                          1           4
[1,1,0,1];

enter the desired trajectory you want the output of your system to track



yd[ 1 ](t)=
exp(t);

yd[ 2 ](t)=
t;


relative order of system = 2

The inverse system to our non linear control system is :
```

$$\frac{dx}{dt} = binv(x)\ .\ yi(t) + ainv(x)$$

$$u(t) = Cinv(x) + Dinv(x)\ .\ yi(t)$$

where $yi(t) = \left[\frac{d^2 y1}{dt^2}, \frac{d^2 y2}{dt^2}\right]$

$$Ainv(x) = \begin{bmatrix} x2 \\ 0 \\ x4 \\ 0 \end{bmatrix}$$

$$Binv(x) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$Cinv(x) = \begin{bmatrix} -\frac{2\ x1\ x4^2 - x4^2}{2} \\ 2\ x1\ x2\ x4 - x2\ x4 \end{bmatrix}$$

$$Dinv(x) = \begin{bmatrix} 1 & 0 \\ 0 & \frac{4\ x1^2 - 4\ x1 + 5}{4} \end{bmatrix}$$

Checking if yd(t) is trackable by our system (right-invertibility)

yd(t) can be tracked by the output y(t)

The control ud(t) that makes y(t) track yd(t) is:

$$ud(t)[\ 1\ ] = \left[-\frac{2\ x1\ x4^2 - x4^2 - 2\ \%e^t}{2},\ 2\ x1\ x2\ x4 - x2\ x4\right]$$

$$ud(t)[\ 2\ ] = \left[-\frac{2\ x1\ x4^2 - x4^2 - 2\ \%e^{-t}}{2},\ 2\ x1\ x2\ x4 - x2\ x4\right]$$

Are you interested in simulation results ? (answer y or n)
y;

enter filename of fortran code
fort;

enter initial time you would the simulation to start from
0.0;

enter final time tf
5.0;

enter step size h
0.05;

enter initial condition in the form [xo[1],...,xo[4]
[0,0,0,0];

```
      dimension x( 4 ),dx( 4 ),datad(1000, 2 ),data(1000,
     1 2 ),u( 2 ),y( 2 )
c     set no of equations
      n= 4
      m= 2
c     set initial conditions
      x(1) = 0
      x(2) = 0
      x(3) = 0
      x(4) = 0
      y(1) = x(1)
      y(2) = x(3)
c     set initial & final time
      t= 0.0
      tf= 5.0
c     set step size
      h= 0.05
c     desired trajectory
      no= 100
      do 15 i=1,no
      datad(i,1) = exp(t)
      datad(i,2) = t
      t=t+h
 15   continue

c     store initial values for plotting
      do 25 i=1, 2
      data(1,i)=y(i)
 25   continue
      t=0.0
      call control(x,t,u)
c     initialize k & mm
      k=0
      mm=1
      print *,t,datad(mm,1),data(mm,1)
c     write down the differential equations
 1    n= 4
      dx(1) = x(2)
      dx(2) = x(1)*x(4)**2-x(4)**2/2.0+u(1)
      dx(3) = x(4)
      dx(4) = (4*x(2)*x(4)-8*x(1)*x(2)*x(4))/(4*x(1)**2-4*x(1)+5)+4*u(2)
     1   /(4*x(1)**2-4*x(1)+5)
      call runta(n,k,ii,x,dx,t,h,u)
      go to (1,2),ii
 2    mm=mm+1
      y( 1 )= x(1)
      y( 2 )= x(3)
      do 30 i=1, 2
      data(mm,i)=y(i)
 30   continue
      print *,t,datad(mm,1),data(mm,1)
      if(t.le.tf) go to 1
      stop
      end

      subroutine runta(n,k,ii,x,dx,t,h,u)
      dimension yf( 4 ),z( 4 ),x( 4 ),dx( 4 ),u( 2 )
      k=k+1
      go to (1,2,3,4,5),k
 2    do 10 j=1,n
      z(j)=dx(j)
      yf(j)=x(j)
 10   x(j)=yf(j)+0.5*h*dx(j)
 25   t=t+0.5*h
      call control(x,t,u)
 1    ii=1
      return
 3    do 15 j=1,n
      z(j)=z(j)+2.0*dx(j)
 15   x(j)=yf(j)+0.5*h*dx(j)
      call control(x,t,u)
      ii=1
      return
 4    do 20 j=1,n
      z(j)=z(j)+2.0*dx(j)
 20   x(j)=yf(j)+h*dx(j)
      go to 25
 5    do 30 j=1,n
 30   x(j)=yf(j)+(z(j)+dx(j))*h/6.0
      call control(x,t,u)
      ii=2
      k=0
      return
      end

      subroutine control(x,t,u)
      dimension u( 2 ),x( 4 )
      u(1) = exp(t)-x(1)*x(4)**2+x(4)**2/2.0
      u(2) = -(4*x(2)*x(4)-8*x(1)*x(2)*x(4))/4.0
      return
      end
```

Example 2

Check if the following nonlinear system is feedback-linearizable to a controllable linear system and if so, find the F-transformation.

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \end{pmatrix} = \begin{pmatrix} \sin(x_2) \\ \sin(x_3) \\ x_4^3 \\ x_5 + x_4^3 - x_1^{10} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

```
(c2) load("initfile.mac");
(d2)                                initfile.mac
(c3) f:[sin(x2),sin(x3),x4**3,x5+x4**3-x1**10,0];
                                    3          3     10
(d3)          [sin(x2), sin(x3), x4 , x5 + x4  - x1  , 0]
(c4) g:[[0,0,1,0,0],[0,0,0,0,1]];
(d4)                  [[0, 0, 1, 0, 0], [0, 0, 0, 0, 1]]
(c5) transform(f,g);

Hello,TRANSFORM tries to solve the problem:                                    #

          Given the non linear system:

                          dx
                          -- = f(x) + u  g (x) + u  g (x)
                          dt           2  2       1  1

find a non singular transformation that takes this system to a controllable

linear system:

                                  dz
                                  -- = b . v + a . z
                                  dt

checking if the system is in block triangular form....
system not in block triangular form ==> trying the general method...
the system is multi-input ==> computing first the Kronecker indices of the equ#

ivalent controllable linear system

Kronecker indices are:
k[ 1 ]= 3
k[ 2 ]= 2
checking the first condition of transformability.....
checking the second condition of transformability...
checking the third condition of transformability....
```

span of c1 =span of c1 /C

span of c2 =span of c2 /C

All the conditions are satisfied

trying to construct (if possible) the transformation....

The new state variables are :

z[1]= x1

z[2]= sin(x2)

z[3]= cos(x2) sin(x3)

z[4]= - x4

$$z[\ 5\ ]= - x5 - x4^{3} + x1^{10}$$

The new control variables are :

$$v[\ 1\ ]= \cos(x2)\ \cos(x3)\ (x4^{3} + u1) - \sin(x2)\ \sin^{2}(x3)$$

$$v[\ 2\ ]= - 3\ x4^{2}\ (x5 + x4^{3} - x1^{10}) + 10\ x1^{9}\ \sin(x2) - u2$$

(d5) done

References

[1] BOOTHBY, W. M., 1975, *An Introduction to Differentiable Manifolds and Riemannian Geometry*, Academic Press, Inc.m New York.

[2] BROCKETT, R. W., 1978, *Feedback Invariants for Nonlinear Systems*, IFAC Congress, Helsinki. p. 1115-1120.

[3] HIRSCHORN, R. M., 1979, *Invertibility of nonlinear control systems*, SIAM J Control Optim, 17, pp. 289-297.

[4] HUNT, R. L., SU, R., and MEYER, G., 1983, *Design for multi-input systems*, Differential Geometric Control Theory, edited by R. Brockett, R. Millman and H. J. Sussman, Birkhauser, Boston, vol. 27, pp. 268-298.

[5] HUNT, R. L., SU, R., and MEYER, G., 1983, *Global Transformations of Nonlinear Systems*, IEEE Trans Aut Control, AC-28, No. 1, pp. 24-31.

[6] JAKUBCZYK, B., and RESPONDEK, W., 1980, *On linearization of control systems*, Bull Acad Polon Sci, Ser Sci Math Astronom phys, 28, pp.517-522.

[7] KRENER, A. J., 1973, *On the equivalence of control systems and the linearization of nonlinear systems*, SIAM J Control, 11, pp. 670-676.

[8] MEYER, G., and CICOLANI, L., 1980, *Application of nonlinear system inverses to automatic flight control design-system concepts and flight evaluations*, AGARDograph 251 on Theory and Applications of Optimal Control in Aerospace Systems, P.Kent, ed., reprinted by NATO.

[9] MEYER, G., 1981, *The design of exact nonlinear model followers*, Proceedings of Joint Automatic Control Conference, FA3A.

[10] SU, R., 1982, *On the linear equivalents of nonlinear systems*, Systems and Control Letters, 2, No 1, pp. 48-52.

[11] THE MATLAB GROUP LABORATORY FOR COMPUTER SCIENCE, 1983, *MACSYMA Reference Manual*, M.I.T., Cambridge, Mass.

[12] WONHAM, W. M., 1979, *Linear Multivariable Control: A Geometric Approach*, (New York: Springer-Verlag).

[13] AKHRIF, O., 1987, *Using Computer Algebra for Design of Nonlinear Control Systems*, M.S. Thesis, Univ. of Maryland, College Park, MD.