Combination of Symbolic Manipulation and Numerics

Holmberg, Ulf; Lilja, Mats; Mattsson, Sven Erik

1987

Document Version:
Publisher's PDF, also known as Version of record

CODEN: LUTFD2/(TFRT-3186)/1–018/(1987)

Combination of Symbolic Manipulation and Numerics

Ulf Holmberg
Mats Lilja
Sven Erik Mattsson

STU project 85–4809

Department of Automatic Control
Lund Institute of Technology
March 1987

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
Report

Date of issue
March 1987

Document Number
CODEN: LUTFD2/(TFRT-3186)/1–018/(1987)

Author(s)
Ulf Holmberg
Mats Lilja
Sven Erik Mattsson

Supervisor

Sponsoring organisation
The National Swedish Board of Technical Development (STU contract 85-4809)

Title and subtitle
Combination of Symbolic Manipulation and Numerics

Abstract

Symbolic manipulation will play an important role in future tools for Computer Aided Control Engineering (CACE). The purpose of this project has been to experiment with and gain experiences of using a program for symbolic manipulation. As the tool we used the symbolic manipulation program MACSYMA and as the application analysis of multivariable systems was selected. A small package for analysis of multivariable linear systems is constructed inside MACSYMA. Manipulations of polynonial matrices are given as example. Also it is shown how it is possible to transfer results from MACSYMA to numerical programs, like the matrix-manipulation program CTRL-C and the differential equation program Simnon. By combining symbolic and numeric computations we can solve more complex and composit problems. This is demonstrated by solving for multivariable root locus.

Key words
Symbolic Manipulations; MACSYMA; Computer Aided Control Engineering; Multivariable Linear Systems; Polynomial Matrices.

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
18

Recipient's notes

Security classification

Contents

1. Introduction

Symbolic manipulation will play an important role in future CACE tools. Unfortunately, today's systems for Computer Aided Control Engineering (CACE) allow basically the user to perform numerical calculations. They do not support symbolic calculations. One important reason for this is that the tools were designed for computers with what is today considered as moderate computing power and symbolic manipulation calls for computing power. The increasing capacity of computers and workstations makes it now worthwhile to introduce symbolic calculations in CACE systems.

[Pavelle et al., 1981] give a popular scientific introduction to computer algebra. There are commercial general-purpose systems for symbolic systems available:

MACSYMA	developed at the MIT Laboratory for Computer Science, USA
REDUCE	developed at Stanford University, the University of Utah and the Rand Corporation, USA
Scratchpad	developed by IBM
SMP	developed at the California Institute of Technology, USA
Maple	developed at the University of Waterloo, Canada
muMATH	developed by the Soft Warehouse, Honolulu

The main purpose of the project "Combination of Symbolic Manipulation and Numerics" was to experiment with and gain experiences of using a program for symbolic manipulation. As the tool we used MACSYMA and as the application analysis of multivariable systems was selected. There is a framework for analysis and design of multivariable systems using polynomial matrices. A standard text book is [Kailath, 1980]. Unfortunately, these methods have poor numerical properties. Methods based on state space representations have better numerical properties. However, in many cases it is desirable to be able to work in the frequency domain. It may be easier to formulate and analyse properties of interest in the frequency domain than in the state space. MACSYMA is good at polynomials and rational functions.

Motives for supporting symbolic calculations are presented in Chapter 2. The package developed in MACSYMA for analysis of multivariable linear systems is described in Chapter 3. A Lisp function in MACSYMA can be used to establish interaction with other programs, such as Simnon and CTRL-C. This is presented in Chapter 4. In Chapter 5 a new method for calculating root loci is demonstrated. This serves as an example of the idea to combine symbolic and numerical computation and thereby solve more complex and composite problems. Conclusions are given in Chapter 6.

2. Motives

There are several good reasons for including symbolic manipulation in CACE tools. First, structure is important and an analytic answer may give a better insight. Second, the user interface could be improved, since the user's original formulation is usually not a computational procedure but rather equations and relations on symbolic form. Third, models could be multi-purpose and reusable independent of what is known and what should be computed. Forth, symbolic manipulation could be used to facilitate numerical solution. Below we will discuss these motives in more detail.

Insight is Desirable

You may think that the ultimate CACE program is an automatic procedure which outputs a VLSI chip that implements the optimal controller. Life is not that easy. You must at least specify your desires and requirements; a specification of what **you** think is optimal. Unfortunately, this may be a laborious and demanding task. Many problems are not that well-defined. If it is a new type of plant, it might be difficult to know which are the decisive requirements and which that are easy to fulfil. A given constraint may be totally decisive for the outcome of the control design. A designer may be willing to adjust the requirements to achieve other benefits, but he is not willing to consider every case or combination. He wants to work in an iterative way and be able to eliminate bad approaches early. He also wants to know why a certain approach fails. He wants to get insight. For example, it may be easy and favourable to remove a constraint by redesigning the plant. It is in most cases favourable to take the interaction between process design and control design into account and consider them simultaneously.

A designer is happy when he has a profound understanding of the system dynamics. He then knows the possibilities and the limitations and can make the proper compromises during the design. He can justify why it is not possible to make a better design according to the circumstances. In many cases insight is the key to design. If you can pinpoint the critical parts and if you understand the difficulties, you can often solve or avoid the problems and make a good design.

In real life most plants have significant non-linear behavior, while most available software for analysis and synthesis assumes linear models. It is difficult to analyse non-linear systems. The simulation model could be used for empirical studies concurrently with a mathematical analysis. Possibilities to include and exclude different features in the model by changing the model for one part or by making parameter changes are useful when studying their importance. To have some success with the analysis we are more or less forced to work mainly with linear models and to estimate the effects of nonlinearities. Linearization is tedious to do with paper and pen. A good formula manipulation program which takes the nonlinear equations and outputs the linearized ones would be a real time-saver. If there also was a program that took the linear model and intervals for the parameters and made proper approximations, the analysis would be even simpler to carry out. A nice thing with linear models is that they can be transformed into the frequency domain where many dynamical properties are easier to understand. When analysing a system it is useful to have different viewpoints and possibilities to transform back and forth between different representations.

Support Users' Concepts

It is important that a user can describe his problems on a for him natural form. The user interface of a CACE system could be viewed as consisting of a language and environment. The language should be more than just a means for instructing the computer to perform tasks. It should also serve as a framework within which we organize our ideas. It should be a high level problem solving language.

It is important that the user can give the mathematical description of a submodel on a natural form. When deriving models from first principles the result is often a system of differential algebraic equations (DAE):

$$g(t, \dot{x}, x, v, p, c) = 0$$

where t is the time, x and v vectors of unknown variables, p a vector of known parameters and c a vector of known constants. It is natural to require that interactive software for model development and simulation supports DAE systems. The prototype simulator Hibliz ([Elmqvist and Mattsson, 1986]; [Mattsson, Elmqvist and Brück, 1986]) which was developed in another CACE project (STU project 84-5069) accepts mathematical descriptions given as DAE systems. However, most simulation packages of today do not allow models given as DAE systems, but require assignment statements for derivatives and algebraic variables. The user must solve for the derivatives and put the model on the form

$$\dot{x} = f(t, x, p, c)$$

He is often allowed to introduce sequences of auxiliary variables and to give the assignment statements in any order:

$$\dot{x} = f_1(t, \dot{x}, x, v, p, c)$$
$$v = f_2(t, \dot{x}, x, v, p, c)$$

as long as it possible to sort them so that all derivatives and auxiliary variables are calculated before use. This means that the user has to manipulate his model manually. This is a non-trivial task. Errors may be introduced.

When DAE systems are supported, the model becomes more readable since the user can recognize fundamental relations as mass and energy balances and other phenomenological equations. It is easier to check that the model is entered correctly and the risk of introducing errors during manual transformation is reduced.

Multi-Purpose and Reusable Descriptions

It is a laborious and time-consuming task to develop good models of plants and various phenomena. Consequently, it is important that the investments in model development can be reused. A model can be used for different purposes as simulation, analysis and design. The status of a variable may vary. Sometimes it is considered to be known, while in other situations we want to solve for it. For example, when solving for a stationary operating point the derivatives are set to zero and the states are to be solved for. When a numerical ODE solver is used, the states are considered to be known and we should solve for the derivatives. When designing the plant or the control system, some of the parameters are considered to be unknown by the designer.

Furthermore, as thoroughly motivated by [Elmqvist, 1978], the equation form is the only reasonable representation for model libraries. With models on assignment form, it must be decided for each submodel which of its variables that are inputs (in other words are known) and which of its variables that are outputs (defined by the model). As a simple example consider a resistor. Ohm's law states $V_1 - V_2 = RI$, where V_1 and V_2 are the voltages at the ends of the resistor, I is the current through the resistor and R is the resistance. The model has three variables V_1, V_2 and I. The resistance R is in this model a given parameter. If we should write the model on assignment form there are three possibilities

$$I := (V_1 - V_2)/R$$
$$V_1 := V_2 + RI$$
$$V_2 := V_1 - RI$$

The first variant assumes that V_1 and V_2 are inputs and defines I. This model is appropriate if for example one end of the resistor is connected to a voltage source and the other to ground. The second and third variants assume that the current and the voltage at one end is known. These models are appropriate if the resistor is connected to a current source and ground. Consequently, for models on assignment form we need several different models for a resistor, depending on how it is connected to the environment. This makes both use and maintenance of a model library messy. Furthermore, other environments may result in algebraic loops so that equation systems with equations from several submodels must be solved to transform the model into assignment form. Two resistors connected in series between a voltage source and ground is a simple example of this. Submodels cannot be transformed into assignment form individually, but the transformation is a global problem.

Improve the Numerical Properties

It is favourable if a CACE system accepts problems on forms preferred by users. By exploring symbolic manipulation the problems can in many cases be simplified and transformed to a form more suitable for numeric solution.

As an example consider the problem of finding the optimum of a function. The numerical solution procedure could be made faster and more robust if analytic procedures for calculating the gradient and the Hessian are given. However, in many cases it is laborious for the user to provide these procedures. It is much more convenient for him if they are generated automatically.

You may say that a problem is ill-conditioned if a small perturbation in the equations can lead to a large deviation in the solution. The main question is, however, what perturbations we have to consider in a particular case. If we have a fully parameterized model, where all explicit numbers are exact (structural ones, zeros etc), the perturbations of interest are those described by perturbations in the parameters. If we want to perform a numerical calculation and substitute the parameters with numbers, then it is of interest to consider unstructured and random perturbations to model for example quantization. Then a larger class of problems becomes ill-conditioned. For a fully parametrized problem the condition number is not a problem invariant, but it depends on the formulation and may be decreased by symbolic manipulation.

It is important to consider the structural properties of a problem when deciding

whether it is well-posed or not. For example the problem $\epsilon\dot{y}+y=1$ where we know that the system is stable, is well-posed. The only perturbations that we have to consider are perturbations in ϵ which lead to an ϵ greater or equal zero. Even from a numerical view, it must be considered to be well-posed. It is a minimum demand that a non-negative number is represented by the computer as a non-negative number.

Possibilities to use symbolic manipulation to handle DAE systems are discussed in [Mattsson, 1986].

3. An Analysis Package in MACSYMA

The framework of polynomial matrices is useful for analysis of multivariable linear systems, see [Kailath, 1980]. However, polynomial matrices are not easily manipulated by hand. It is thus very important that good analysis tools for polynomial matrices are available. We have tried to fill the gap between theory and practice by implementing a package for analysis of multivariable linear systems in MACSYMA. The functions of this package is listed below. Then a MACSYMA demo with matrix fraction decompositions, co-prime factorizations, multivariable realizations, etc., will illustrate the beauty of symbolic manipulations. The examples are taken from [Kailath, 1980]. For further examples and details on the implementation including listings of the functions we refer to [Holmberg, 1986].

Available Functions

The following functions for analysis of multivariable linear systems have been implemented in MACSYMA.

Linearization

`LINEARIZE`	Linearizes the dynamical system $\dot{x} = f(x,u), \quad y = g(x,u)$

Stability analysis

`ROUTH`	Generates the stability conditions for a continuous time system
`JURY`	Gives the stability conditions of discrete time systems and the steady state output variance

Sampling

`SAMP`	Sampling from transfer function to pulse-transfer function
`SAMPSTATE`	Sampling from state space to state space

Geometry functions — state space

`HERMITE`	Gaussian elimination when applied to a constant matrix
`KER`	Computes the Kernel $\{X \mid AX = 0\}$
`INVERSE_IMAGE`	Calculates the inverse image $\{X \mid AX = B\}$ (A possibly singular)
`INTERSECTION`	Computes the intersection of two subspaces
`GRAM_SCHMIDT`	Calculates an orthogonal base for a subspace
`AINV`	Computes the maximal A-invariant subspace in a given subspace
`ABINV`	Computes the maximal (A,B)-invariant subspace in a given subspace

Factorization — Frequency domain

`SMITH`	Calculates the Smith form together with transformation matrices
`SMITH_McMILLAN`	Calculates the Smith-McMillan form with transformation
`HERMITE`	Calculates the Hermite form
`COLUMNREDUCE`	Makes a denominator polynomial matrix column reduced
`ROWREDUCE`	Makes a denominator polynomial matrix row reduced
`RMFD`	Right Matrix Fraction Decomposition (MFD) of a transfer matrix
`LMFD`	Left MFD of a transfer matrix
`RIGHTCOPRIME`	Gives a right coprime MFD from a noncoprime MFD
`LEFTCOPRIME`	Gives a left coprime MFD from a noncoprime MFD

SS2TF	State space to transfer function conversion
MAKESYS	Makes a list of the A, B, C, D matrices to represent a system

Multivariable Realizations

CONTROLLER	Calculates a controller form realization
OBSERVER	Calculates an observer form realization
CONTROLLABILITY	Calculates a controllability form realization
OBSERVABILITY	Calculates an observability form realization

Generation of the $\mathcal{S}(A, B, C, D)$-file

TOMIMO	Generates the file ABCD.MIM from A, B, C, and D.

Example—Polynomial Matrix Manipulations

The following example is a MACSYMA *Demo* that illustrate the use of polynomial matrices for analysis of multivariable linear systems. The cumbersome manipulations are done by the above functions. The Demo starts with a transfer function matrix, describing a multivariable linear system. The description is transfered into a matrix fraction decomposition, MFD, i.e. a polynomial matrix description. Extraction of different polynomial matrix factors of the MFD are made. Also, different multivariable realizations are presented. For terminology and a background the reader is refered to [Kailath], especially Chapter 6.

The file shown is a MACSYMA log file, output with the **typeset** switch **true**. The resulting Troff/EQN typesetting code has automatically translated to TEX by the program MacEQ2TEX (see [Mårtensson, 1986]).

```
(c1) load("login.mac")$
(c2) demo("realizations.dem");
/* This demo describes a couple of examples in Kailath, chapter 6.
Example 6.2-1. Alternative MFDs for a Transfer Function. p. 368-9.
Example 6.4-1. Controller-Form Realization of a Right MFD. p. 407-8.
Example 6.4-2. Observer-Form Realization of a Left MFD. p. 416
Example 6.4-6. Constructing Canonical Controllability Forms. p. 433-4. */
(c3) g:matrix([s/((s+1)*(s+2))^2,s/(s+2)^2],[-s/(s+2)^2,-s/(s+2)^2]);
```

(d3)
$$\begin{bmatrix} \frac{s}{(s+1)^2(s+2)^2} & \frac{s}{(s+2)^2} \\ -\frac{s}{(s+2)^2} & -\frac{s}{(s+2)^2} \end{bmatrix}$$

```
/* Example 6.2-1. Alternative MFDs for a Transfer Function. p. 368-9. */
(c4) rmfd(g);
```

(d4)
$$\left[dr = \begin{bmatrix} (s+1)^2(s+2)^2 & 0 \\ 0 & (s+2)^2 \end{bmatrix}, nr = \begin{bmatrix} s & s \\ -s(s+1)^2 & -s \end{bmatrix}\right]$$

```
(c5) ev(rightcoprime(dr,nr),%);
```

(d5)

$$\left[dr=\begin{bmatrix}(s+1)^2(s+2)^2 & -(s+1)^2(s+2)\\ 0 & s+2\end{bmatrix},nr=\begin{bmatrix}s & 0\\ -s(s+1)^2 & s^2\end{bmatrix},rr=\begin{bmatrix}1 & 1\\ 0 & s+2\end{bmatrix}\right]$$

```
(c6) ev(columnreduce(dr,nr),%);
```

(d6)

$$\left[dr=\begin{bmatrix}0 & -(s+1)^2(s+2)\\ (s+2)^2 & s+2\end{bmatrix},nr=\begin{bmatrix}s & 0\\ -s & s^2\end{bmatrix},u=\begin{bmatrix}1 & 0\\ s+2 & 1\end{bmatrix}\right]$$

```
/* Example 6.4-1. Controller-Form Realization of a Right MFD. p. 407-8. */
(c7) ev(real:controller(dr,nr),%);
```

(d7)

$$\left[a=\begin{bmatrix}-4 & -4 & 0 & -1 & -2\\ 1 & 0 & 0 & 0 & 0\\ 0 & 0 & -4 & -5 & -2\\ 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1 & 0\end{bmatrix},b=\begin{bmatrix}0 & 1\\ 0 & 0\\ -1 & 0\\ 0 & 0\\ 0 & 0\end{bmatrix},c=\begin{bmatrix}1 & 0 & 0 & 0 & 0\\ -1 & 0 & 1 & 0 & 0\end{bmatrix}\right]$$

```
/* Example 6.4-6. Constructing Canonical Controllability Forms. p. 433-4.
*/
/* Search by Crate 2 */
(c8) ev(controllability(a,b,c),%);
```

(d8)

$$\left[a=\begin{bmatrix}0 & 0 & -2 & 0 & 0\\ 1 & 0 & -5 & 0 & 0\\ 0 & 1 & -4 & 0 & 0\\ 0 & 0 & 2 & 0 & -4\\ 0 & 0 & 1 & 1 & -4\end{bmatrix},b=\begin{bmatrix}1 & 0\\ 0 & 0\\ 0 & 0\\ 0 & 1\\ 0 & 0\end{bmatrix},c=\begin{bmatrix}0 & 0 & 1 & 1 & -4\\ -1 & 4 & -12 & -1 & 4\end{bmatrix}\right]$$

```
/* Search by Crate 1 */
(c9) crate_nr:1;
```

(d9) 1

```
(c10) ev(controllability(a,b,c),real);
```

(d10)

$$\left[a=\begin{bmatrix}0 & 0 & 0 & -4 & 2\\ 1 & 0 & 0 & -12 & 5\\ 0 & 1 & 0 & -13 & 4\\ 0 & 0 & 1 & -6 & 1\\ 0 & 0 & 0 & 0 & -2\end{bmatrix},b=\begin{bmatrix}1 & 0\\ 0 & 0\\ 0 & 0\\ 0 & 0\\ 0 & 1\end{bmatrix},c=\begin{bmatrix}0 & 0 & 1 & -6 & 1\\ -1 & 4 & -12 & 32 & -1\end{bmatrix}\right]$$

```
/* Example 6.4-2. Observer-Form Realization of a Left MFD. p. 416 */
(c11) lmfd(g);
```

$$\text{(d11)}\qquad \left[dl=\begin{bmatrix}(s+1)^2(s+2)^2 & 0\\ 0 & (s+2)^2\end{bmatrix},nl=\begin{bmatrix}s & s(s+1)^2\\ -s & -s\end{bmatrix}\right]$$

```
(c12) ev(leftcoprime(dl,nl),%);
```

$$\text{(d12)}\qquad \left[dl=\begin{bmatrix}(s+1)^2(s+2)^2 & 0\\ (s+1)^2(s+2) & s+2\end{bmatrix},nl=\begin{bmatrix}s & s(s+1)^2\\ 0 & s^2\end{bmatrix},rl=\begin{bmatrix}1 & 0\\ -1 & s+2\end{bmatrix}\right]$$

```
(c13) ev(rowreduce(dl,nl),%);
```

$$\text{(d13)}\qquad \left[dl=\begin{bmatrix}0 & -(s+2)^2\\ (s+1)^2(s+2) & s+2\end{bmatrix},nl=\begin{bmatrix}s & s\\ 0 & s^2\end{bmatrix},u=\begin{bmatrix}1 & -(s+2)\\ 0 & 1\end{bmatrix}\right]$$

```
(c14) ev(observer(dl,nl),%);
```

$$\text{(d14)}\qquad \left[a=\begin{bmatrix}-4 & 1 & 0 & 0 & 0\\ -4 & 0 & 0 & 0 & 0\\ 0 & 0 & -4 & 1 & 0\\ 1 & 0 & -5 & 0 & 1\\ 2 & 0 & -2 & 0 & 0\end{bmatrix},b=\begin{bmatrix}1 & 1\\ 0 & 0\\ 0 & 1\\ 0 & 0\\ 0 & 0\end{bmatrix},c=\begin{bmatrix}0 & 0 & 1 & 0 & 0\\ -1 & 0 & 0 & 0 & 0\end{bmatrix}\right]$$

4. Generation of a MIMO System Text File

It will now be demonstrated how a MIMO system in MACSYMA can be transfered into a text file of a special form. The special form of the text file is chosen to be the same as the print format from CTRL-C. This makes it possible to load results form MACSYMA into CTRL-C. It should also be mentioned that there is a Pascal program written by [Mårtensson, 1986] that generates Simnon code from this text file representation. The generation of the text file from MACSYMA is made by the function TOMIMO. This is a LISP program and consequently we have to enter the LISP mode before we apply it to our MIMO system.

```
(c1) load("login.mac")$
(c2) a:matrix([1,2],[3,4])$
(c3) b:matrix([5,6],[7,8])$
(c4) c:matrix([9,10],[11,12])$
(c5) d:matrix([13,14],[15,16])$
(c6) sys:makesys(a,b,c,d);
```

$$\text{(d6)} \qquad \left[a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, b = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}, c = \begin{bmatrix} 9 & 10 \\ 11 & 12 \end{bmatrix}, d = \begin{bmatrix} 13 & 14 \\ 15 & 16 \end{bmatrix}\right]$$

```
(c7)
Break Entering Lisp:
<1>: (load "tomimo.l")
t
<1>: (tomimo $sys 'abcd.mim)
t
```

The MIMO system has now been written in the file `ABCD.mim`. This file looks as follows:

```
nmp =

2 2 2

a =

1 2
3 4

b =

5 6
7 8
```

```
c =

9 10
11 12

d =

13 14
15 16
```

---

5. Root Locus

In this section we will demonstrate a method for computing root loci by combining symbolic and numerical computations. Only a very brief description will be given. A fuller description is given in [Holmberg, Lilja, Mårtensson, 1986].

The naive way of plotting root loci of the type

$$A(s) + kB(s) = 0$$

where A and B are polynomials and k real, is to solve the equation for a number of equi-distant k-values and then mark the roots by an '$\times$' for each k. This method has severe disadvantages: Firstly it is rather time consuming and secondly it gives a very bad resolution near multiple roots. To be presentable, the plots also need heavy manual paste-up. In the following subsection, a method based on the implicit function theorem is suggested. A non-linear differential equation, that describes the root locus locally, is obtained by some manipulations of the transfer function (done in MACSYMA for example) and a package for solving the differential equation (e.g. Simnon) can then be utilized to compute and plot the root locus. This method is both faster and gives a better performance near multiple roots than the method mentioned above.

A general problem

This subsection proposes a method for plotting the locus of points s in the complex plane satisfying the equation

$$f(s,k) = 0, \qquad s,k \in \mathbb{C} \tag{♭}$$

where f is analytic in s and k and where k is restricted to the real axis. Several common control theory problems are covered in this formulation: Ordinary root loci, LQG root loci (k = control weighting), zeros of sampled systems (k = the sampling interval), etc.

The method is based on the implicit function theorem applied to (♭). The idea is the following: The problem is to compute $\{s | f(s,k) = 0, \quad k \in [a,b] \subset \mathbb{R}\}$. For this, compute the k's such that (♭) has multiple roots in s. Away from these, the branches $s_i(k)$ satisfies

$$\frac{d}{dk}s_i(k) = -\frac{\dfrac{\partial f}{\partial k}}{\dfrac{\partial f}{\partial s}} \tag{♯}$$

Implementation

The transfer function $G(s)$ is specified in MACSYMA and the closed loop characteristic polynomial $p(s,k)$ is calculated. The real and imaginary part of the right hand side of (♯) are then computed and written to a file using the function `print_ode`. To avoid divide overflow in Simnon when a multiple root is encountered one has to stop the integration before the multiple root. For each multiple root, one therefore has to find the local behavior of s with respect to k. A graphical method to do this is to plot a

Newton diagram. This is equivalent to making the substitution $s := bk^a$ in the characteristic polynomial and then finding pairs of dominating terms. The function `newton` implements this and returns two lists. The first list contains the possible k^a-alternatives and the second list gives the corresponding coefficients (expressed as a polynomial in b). The function `near_multiple_roots` uses `newton` for calculating the values of k for which $|\frac{d}{dk}s_i(k)| = d_{max}$. The function `print_kxy` uses `near_multiple_roots` to print out these k values and the corresponding solutions in s.

The differential equation for the real and imaginary parts of the root locus is written into the file `ode.rl`. The k-values specifying the intervals for which the root locus is to be plotted for are written (together with the corresponding initial values for the branches of the root locus) into the file `rootloc.rl`. These two files are then processed by a procedure written in the "editor language" TPU (Text Processing Utility) in VMS generating one Simnon system description file `ode.t` (the "dynamics" file) and one file `rootloc.t` containing the commands for setting initial values and integrating.

An Example

The following MACSYMA dialogue shows an example where the functions `print_ode` and `print_kxy` are used. In the example the interval for the gain k is chosen to $-2 \leq k \leq 2$ and the maximum derivative to $d_{max} = 100$.

```
(c1) load("rootloc.mac")$
(c2) g:matrix([1/s^2,1/s],[-1/s^2,0]);
```

(d2)
$$\begin{bmatrix} \frac{1}{s^2} & \frac{1}{s} \\ -\frac{1}{s^2} & 0 \end{bmatrix}$$

```
(c3) print_ode(g);
```

(d3) *ode.rl*

```
(c4) print_kxy(g,-2,2,100);
```

(d4) *rootloc.rl*

The resulting files `ode.rl` and `rootloc.rl` are then processed by the TPU file `rootloc.tpu` to get the Simnon system description file `ode.t` and the Simnon command file ("macro" file) `rootloc.t`. The Simnon commands required to plot the root locus are:

```
> syst ode
> axes h -2 2 v -2 2
> rootloc
```

The result is shown in Figure.

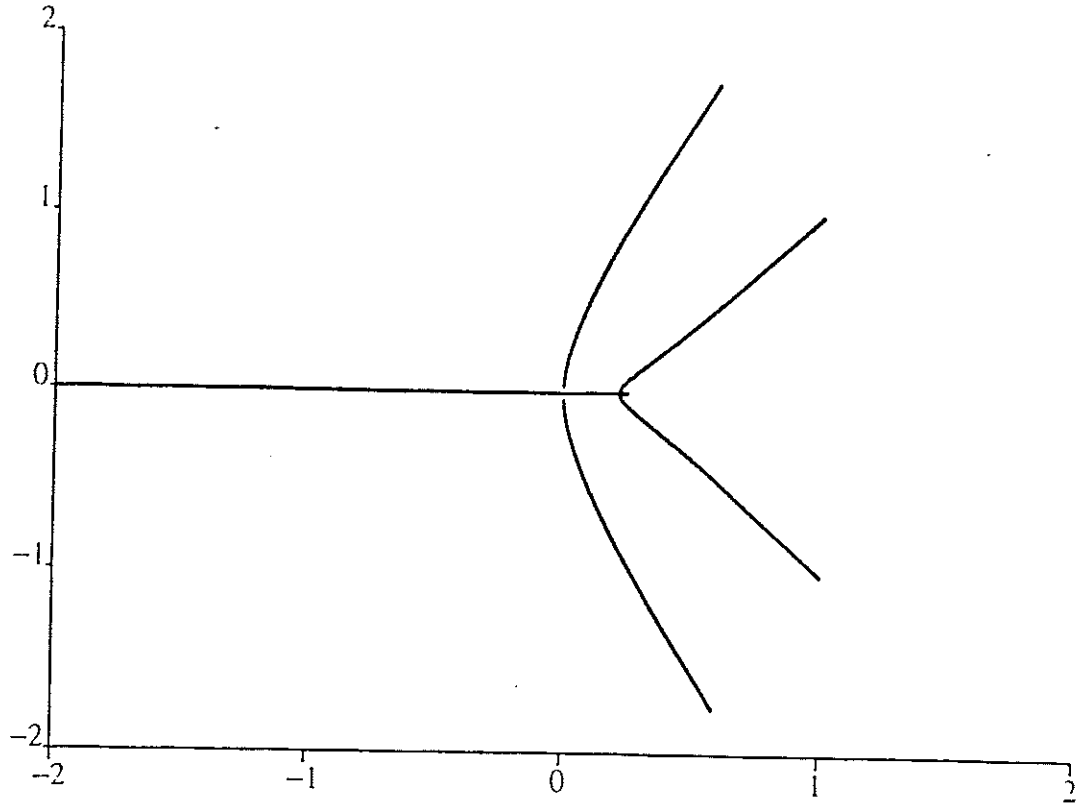

Figure: The root locus plot.

6. Conclusions

As motivated and illustrated symbolic manipulation could be very useful. The user interface could be improved by allowing the user to work on a higher level. He can present his problem on a for him suitable form. Results on analytic or symbolic form could give a better insight into structural properties than numerical tables. Even when it is not possible to carry the symbolic calculations all the way through, symbolic manipulation could be useful. Symbolic manipulation could simplify the problem and transform it to a form better suited for numerical solution. Symbolic manipulation could also be used for automatic generation of procedures for calculating gradients, Jacobians, Hessians etc. thereby relieving the user's burdon and hopefully decreasing the possibilities of introducing errors.

Our experiences of MACSYMA are that it is a powerful tool and can do a lot with proper guidance from the user. One advantage with with MACSYMA is that it is written in Lisp. This makes it possible to extend the program with Lisp functions. As you remember from Chapter 4, this enabled us to establish an interaction between MACSYMA and other programs, like Simnon and CTRL-C. The drawbacks are that it is a large program and that it consumes a lot of computer power. Unfortunately, MACSYMA is not modularized. For use in CACE systems it should be desirable to have modularized tools for symbolic calculation so that a user could select for him a proper set. We are eagerly searching for such a toolkit.

It is important to consider that there is a user in the loop. He can in many cases improve the manipulation by proposing substitutions and by informing the system on what kind of forms he want the answer. In many cases an equation system can be simplified considerably if it can be assumed that a parameter or a certain expression is zero. It is difficult for the user to anticipate all such cases in advance, but he may well be able to answer those questions interactively. Also, if the model is modified there should be facilities to take care of assumptions made before so he doesn't need to consider them once more when the manipulations are redone. To speed up the manipulation it is advisable to store the successful path and try it when the user has modified his model. The logging facility is also necessary for the explanation facility. If the numerical solution procedure fails, the error message should relate to the user's original formulation and not to the manipulated expressions.

Acknowledgements

The work has been supported by the Swedish Board of Technical Development in the project "Combination of formula manipulation and numerics" (STU–85–4809).

References

ELMQVIST, H. (1978): "A Structured Model Language for Large Continuous Systems," Ph.D-thesis TFRT-1015, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

ELMQVIST, H. and S.E. MATTSSON (1986): "A Simulator for Dynamical Systems Using Graphics and Equations for Modelling," *Proceedings of the IEEE Control Systems Society Third Symposium on Computer-Aided Control Systems Design (CACSD)*, Arlington, Virginia, September 24–26, 1986.

ELMQVIST, H., K.J.ÅSTRÖM and T. SCHÖNTHAL (1986): *Simnon—User's Guide for MS–DOS Computers*, Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden.

HOLMBERG U. (1986): "Some MACSYMA Functions for the Analysis of Multivariable Linear Systems," Report CODEN: LUTFD2/(TFRT-7333)/1–40/(1986), Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden.

HOLMBERG, U., LILJA, M., MÅRTENSSON, B. (1986): "Integrating Different Symbolic and Numeric Tools for Linear Algebra and Linear System Analysis," Report CODEN: LUTFD2/(TFRT-7338)/ 1–17/(1986), Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden.

KAILATH, T. (1980): *Linear Systems*, Prentice-Hall Information and System Sciences Series, Englewood Cliffs, New Jersey.

LILJA, M. (1986): "Symbolic and Numerical Computation of Implicit Functions, Especially Root Loci," Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden, to appear.

MÅRTENSSON, B. (1986): "MacEQ2TEX and *S*2TEX—Automatic TEX-code generation from MACSYMA and CTRL-C," Report CODEN: LUTFD2/(TFRT-7334)/1–11/(1986), Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden.

THE MATHLAB GROUP LABORATORY FOR COMPUTER SCIENCE (1983): *MACSYMA Reference Manual*, M.I.T., Cambridge, MA.

MATTSSON, S.E. (1986): "On Differential/Algebraic Systems," Report TFRT-7327, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

MATTSSON, S.E., H. ELMQVIST and D.M. BRÜCK (1986): "New Forms of Man-Machine Interaction," Report TFRT-3181, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

PAVELLE, R., M. ROTHSTEIN and J. FITCH (1981): "Computer Algebra," *Scientific American*, **245**, 6, 136–152.

SYSTEMS CONTROL TECHNOLOGY, INC., *CTRL-C, User's Guide*, 1801 Page Mill Road, Palo Alto, CA 94304.