Proceedings of the 34th
Conference on Decision & Control
New Orleans, LA - December 1995

Computer Assistance In Discovering Formulas And Theorems In System Engineering *

J. William Helton
Mark Stankus
Department of Mathematics, U. C. San Diego La Jolla, Cal 92093-0112
helton@osiris.ucsd.edu mstankus@osiris.ucsd.edu

ABSTRACT

If one reads a typical article on A,B,C,D systems in the control transactions, one finds that most of the algebra involved is non commutative rather than commutative. Thus, for symbolic computing to have much impact on linear systems research, one needs a program which will do noncommuting operations. Mathematica, Macsyma and Maple do not. We have a package, NCAlgebra, which runs under Mathematica which does the basic operations, block matrix manipulations and other things. The package might be seen as a competitor to a yellow pad. Like Mathematica the emphasis is on interaction with the program and flexibility.

The issue now is what types of "intelligence" to put in the package. **[HSW]** (CDC94) focused on procedures for simplifying complicated expressions automatically. In this talk we turn to a much more adventurous pursuit which is in a primitive stage. This is a highly computer assisted method for discovering certain types of theorems.

At the beginning of "discovering" a theorem, an engineering problem is often presented as a large system of matrix equations. The point is to isolate and to minimize what the user must do by running heavy algorithms. Often when viewing the output of the algorithm, one can see what additional hypothesis should be added to produce a useful theorem and what the relevant matrix quantities are.

Rather than use the word "algorithm", we call our method a strategy since it allows for modest human intervention. We are under the impression that many theorems in engineering systems might be derivable in this way.

1 What is a strategy?

We are under the impression that many theorems in engineering systems, matrix and operator theory amount to giving hypotheses under which it is possible to solve large collections of equations. (It is *not* our goal to reprove already proven theorems in engineering systems theory, but rather to develop technique which will be able to be used to discover new theorems.) Any method which assumes that all of the hypothesis can be stated algebraically and are known at the beginning of the computation will be of limited practical use.

What we shall describe is a method which allows one to add (algebraic) hypotheses as one proceeds with the computation. These hypotheses would be motivated by insights gained in the course of a computer session and the user would have to record and justify them independently of the computer run. However, we do want to be extremely systematic so we shall propose a structure which is algorithm like but a bit looser.

1.1 NCProcess

The approach which we will use to manipulate large collections of equations will be based largely upon a noncommutative Gröbner Basis Algorithm (GBA). A person can use this practical approach to performing computations and proving theorems *without knowing anything about* GBA's. Indeed, this article is a self-contained description of our method.

The program which we shall use which is based upon a GBA will be called NCProcess and will be described in §1.4.

The **input** to NCProcess is:

I1. A list of knowns.

I2. A list of unknowns (together with priorities for eliminating them).

I3. A collection of equations in these knowns and unknowns.

The **output** of NCProcess is a list of expressions which is presented to the user as

O1. Unknowns which have been solved for and equations which yield these unknowns.

O2. Equations selected or created by the user. [1] For example, in the context of S1 below, one would

*This work was partially supported by the Air Force Office for Scientific Research and by the National Science Foundation.

[1] These do not exist in the first run. A user-selected equation

0-7803-2685-7/95 $4.00 © 1995 IEEE

want to select the equation E_{17}. There are also times during a strategy when one wants to introduce new variables and equations. This is illustrated in §2.

O3. Equations involving no unknowns.

O4. Equations involving only one unknown.

O5. Equations involving only 2 unknowns. etc.

1.2 Strategy

The idea of a *strategy* is :

S1. Run NCProcess which creates a display of the output (see O1-O5 in §1.1) and look at the list of equations involving only one unknown (say a particular equation E_{17} contains only x_3).

S2. The user must now make a decision about equations in x_3 (e.g., E_{17} is a Riccati equation so I shall not try to simplify it, but leave it for Matlab). Now the user declares the unknown x_3 to be known and runs the GB algorithm again.

S3. The user must again make a decision and the process repeats.

S4. Knowing when a strategy stops is discussed in §1.3.

The above listing is, in fact, a statement of a *1-strategy*. Sometimes one needs a *2-strategy* in that the key is equations in 2 unknowns.

The point is to isolate and to minimize what the user must do. This is the crux of a strategy.

1.3 When to Stop

There are various criteria for stopping.

The digested equations (those in items O1, O2 and O3) often contain the hypotheses of the desired theorem and the main flow of its proof. If the starting equations follow as algebraic consequences of them then we should stop. This last statement is true iff the GB generated by the digested equations reduce (in a standard way) the set of starting equations to 0. Checking this on a computer is a purely mechanical process.

1.4 Redundant Equations

We mentioned earlier that we are using the Gröbner Basis algorithm (GBA). GBA and the formatted output (§1.1) alone are not enough to generate solutions to engineering or math problems. This is because they generate too many equations. It is our hope and our experience that these equations which it generates contain all of the equations essential to solution of whatever problem you are treating. On the other hand, it contains equations derived from these plus equations derived from those derived from these as well as precursor equations which are no longer relevant. That is, a GB contains a few jewels and lots of garbage. In technical language a GB is almost never a minimal basis for an ideal and what a human seeks in discovering a theorem is a minimal basis for an ideal. Thus we have algorithms and substantial software for finding small (or smallest) sets of equations associated to a problem. The process of running GBA followed by an algorithm for finding small sets of equations is what constitutes NCProcess.

is a polynomial equation which the user has selected. An effect which this has is to make the algorithm described in §1.5 treat these equations as "digested". This, for example, implies that they are given the highest priority in eliminating other equations when NCProcess runs. For example, equations which one knows can be solved by Matlab can be selected.

1.5 Summary

We have just given the basic ideas. As a strategy proceeds, more and more equations are digested by the user and more and more unknowns become knowns. Thus we ultimately have two classes of knowns: original knowns $\mathcal{K}_0$ and user designated knowns $\mathcal{K}_U$. Often a theorem can be produced directly from the output by taking as hypotheses the existence of knowns $\mathcal{K}_U \cup \mathcal{K}_0$ which are solutions to the equations involving only knowns.

Assume that we have found these solutions. To prove the theorem, that is to construct solutions to the original equations, we must solve the remaining equations. Fortunately, the digested equations often are in a block triangular form which is amenable to backsolving. This is one of the benefits of "digesting" the equations.

An example, makes all of this more clear.

2 A Simple Example

We derive a theorem due to Bart, Gohberg, Kaashoek and Van Dooren. The reader can skip the statement of this theorem (§2.1) if he wishes and go directly to the algebraic problem statement.

2.1 Background

Theorem([BGKvD]) A minimal factorization

$$\longleftarrow \underset{\text{state dim} = n_1}{[a, b, c, 1]} \longleftarrow \underset{\text{state dim} = n_2}{[e, f, g, 1]} \longleftarrow$$

of a system $[A, B, C, 1]$ correspond to projections P_1 and P_2 satisfying $P_1 + P_2 = 1$,

$$AP_2 = P_2AP_2 \quad (A - BC)P_1 = P_1(A - BC)P_1 \quad (1)$$

provided the state dimension of the $[A, B, C, 1]$ system is $n_1 + n_2$. (which has the geometrical interpretation

that A and $A - BC$ have complimentary invariant subspaces).

We begin discussing how one might derive this theorem by giving the algebraic statement of the problem. Suppose that these factors exist. By the statespace isomorphism theorem (Youla -Tissi), there is map

$$(m_1, m_2): \text{ Statespace of the product } \longrightarrow \text{ Statespace of the original}$$

which intertwines the original and the product system. Also minimality of the factoring is equivalent to existence of left and right inverses $(im_1^T, im_2^T)^T$ to (m_1, m_2). These requirements combine to imply that each of the following expressions is zero.

2.2 The Problem

$$(FAC) \quad \begin{array}{cc} Am_1 - m_1 a - m_2 fc & Am_2 - m_2 e \\ B - m_1 b - m_2 f & -c + C m_1 \\ im_1 m_1 - 1 & im_2 m_2 - 1 \\ im_1 m_2 & im_2 m_1 \\ -g + C m_2 & m_1 im_1 + m_2 im_2 - 1 \end{array}$$

Each of these expressions must equal 0. Indeed minimal factors exist iff there exist solutions m_1, m_2, im_1, im_2, a, b, c, e, f and g to these equations. Here A, B, C are known.

The problem is to solve these equations. That is we want a constructive theorem which says when and how they can be solved.

2.3 Solution via a Strategy

We now apply a strategy to see how one might discover this theorem.

We run NCProcess. The input is the equations (FAC), together with declaration of A, B, C as knowns and the remaining variables as unknowns. Here is the output displayed in what we call a spreadsheet. The "**" below denotes matrix multiplication and -> can be read as an equal sign.

```
=================================================
========== YOUR SESSION HAS DIGESTED ===========
========== THE FOLLOWING RELATIONS =============
=================================================
THE FOLLOWING VARIABLES HAVE BEEN SOLVED FOR:
{a,b,c,e,f,g}

The corresponding rules are the following:
a->im1**A**m1    b->im1**B    c->C0**m1
e->im2**A**m2    f->im2**B    g->C0**m2
```

```
=================================================
========= USER CREATIONS APPEAR BELOW ==========
=================================================
<Empty>
=================================================
=================================================
========= SOME RELATIONS WHICH APPEAR BELOW ====
=============== MAY BE UNDIGESTED ==============
=================================================
=================================================
THE FOLLOWING VARIABLES HAVE NOT BEEN SOLVED FOR:
{A,B,C0,m1,m2,im1,im2}

THE EXPRESSIONS WITH 1 UNKNOWNS ARE THE FOLLOWING.
-------------------------------------------------
The expressions with unknown variables
{m1,im2} and knowns {A,B,C0}
---------------------------------------
im2 ** m1 -> 0
im2 ** B ** C0 ** m1 -> im2 ** A ** m1

The expressions with unknown variables
{m2,im2} and knowns {}
---------------------------------------
im2 ** m2 -> 1

THE EXPRESSIONS WITH 2 UNKNOWNS ARE THE FOLLOWING.
-------------------------------------------------
The expressions with unknown variables
{m1,im1,im2} and knowns {A,B,C0}
---------------------------------------
im1 ** m1 -> 1

m1 ** im1 ** B ** C0 ** m1 -> m1 ** im1 **
A ** m1 + B ** C0 ** m1 - A ** m1        <===

The expressions with unknown variables
{m2,im1,im2} and knowns {A}
---------------------------------------
im1 ** m2 -> 0
im1 ** A ** m2 -> 0

THE EXPRESSIONS WITH 4 UNKNOWNS ARE THE FOLLOWING.
-------------------------------------------------
The expressions with unknown variables
{m1,m2,im1,im2} and knowns {}
---------------------------------------
m2 ** im2 -> -1 m1 ** im1 + 1      <===
```

The unknowns a,b,c,e, f and g are solved for. There are no equations in 1 unknown. There are 4 categories of equations in 2 unknowns. There is one category of equations in 4 unknowns. A user must observe that the first equation which we marked with <=== is an equation in the unknown quantity m_1 ** im_1. One makes the assignment:

$$P_1 = m_1 \, im_1 \, . \tag{2}$$

Then the user may notice that the second equation marked with <=== in an equation in only one unknown quantity m_2 ** im_2 once the above assignment has been made and P_1 is considered known[2]. These

[2] If the user does not notice it at this point, it will become very obvious with an additional run of NCProcess.

observations lead us to "select" (see footnote corresponding to O2 in §1.1) the equations $m_1\, im_1 - P_1$, $im_1\, m_1 - 1$, $m_2\, im_2 - 1 + P_1$ and $im_2\, m_2 - 1$.

Run NCProcess again with (2) added and P_1 declared known as well as A,B and C declared known.

```
=============================================
========== YOUR SESSION HAS DIGESTED ==========
========== THE FOLLOWING RELATIONS ============
=============================================
THE FOLLOWING VARIABLES HAVE BEEN SOLVED FOR:
{a,b,c,e,f,g}
The corresponding rules are the following:
a->im1**A**m1    b->im1**B    c->C0**m1
e->im2**A**m2    f->im2**B    g->C0**m2

THE EXPRESSIONS WITH 0 UNKNOWNS ARE THE FOLLOWING.
-------------------------------------------------
The expressions with unknown variables
{} and knowns {A,B,C0,P1}
---------------------------------------
P1^2 -> P1
P1 ** A ** P1 -> P1 ** A
P1 ** B ** C0 ** P1 -> B ** C0 ** P1 + P1 ** A
  - A ** P1
=============================================
========= USER CREATIONS APPEAR BELOW =========
=============================================
m1**im1->P1
im1**m1->1
im2**m2->1
m2**im2-> -m1 ** im1 + 1

=============================================
=============================================
========= SOME RELATIONS WHICH APPEAR BELOW ====
================ MAY BE UNDIGESTED ============
=============================================
=============================================
THE FOLLOWING VARIABLES HAVE NOT BEEN SOLVED FOR:
{A,B,C0,P1,m1,m2,im1,im2}

THE EXPRESSIONS WITH 2 UNKNOWNS ARE THE FOLLOWING.
-------------------------------------------------
The expressions with unknown variables
{m1,im1} and knowns {P1}
---------------------------------------
im1 ** m1 -> 1 is a user select.
m1 ** im1 -> P1 is a user select.
The expressions with unknown variables
{m2,im2} and knowns {P1}
---------------------------------------
im2 ** m2 -> 1 is a user select.
m2 ** im2 -> -1 P1 + 1 is digested.
```

Note that the equations in the above display which are in the undigested section (i.e., below the lowest line of equal signs) are repeats of those which are in the digested section (i.e., above the lowest line of equal signs). We relist these particular equations simply as a convenience for categorizing them. We will see how this helps us in §3. Since all equations are digested, we have finished using NCProcess (see S4). As we shall see, this output spreadsheet leads directly to the theorem about factoring systems.

3 The End Game

Note that all equations in the spreadsheet are necessary for a factoring to exist, since they are implied by the original equations. The equations involving only knowns play a key role. In particular, they say precisely that, there must exist a projection P_1 such that

$$P_1\, A\, P_1 = P_1\, A \text{ and } P_1\, B\, C\, P_1 = P_1\, A - A\, P_1 + B\, C\, P_1 \tag{3}$$

are satisfied.

The converse is also true and can be verified with the assistance of the above spreadsheet. We now prove this. We begin by assuming a projection P_1 exists which satisfies (3).

- First, solve the two equations in the category $\{im_1, m_1\}$ for the matrices m_1 and im_1.
- Next, solve the expressions in the category $\{im_2, m_2\}$ for the matrices m_2 and im_2.
- Finally, one uses the computed formulas to solve for a, b, c, e, f and g.

Here we used the fact that we are working with matrices and not elements of an abstract algebra. Thus by backsolving through the spreadsheet, we have constructed the factors of the original system $[A, B, C, I]$. This proves

Theorem:[BGKvD] The system A, B, C, I can be factored if and only if there exists a projection P_1 such that $P_1\, A\, P_1 = P_1\, A$ and $P_1\, B\, C\, P_1 = P_1\, A - A\, P_1 + B\, C\, P_1$ are satisfied.

Note the known equations can be neatly expressed in terms of P_1 and $P_2 = 1 - P_1$. Indeed, it is easy to check with a little algebra that these are equivalent to (1). It is a question of taste, not algebra, as to which form one chooses.

Our efforts are in a primitive stage and the brevity of this presentation suppresses some of the advantages and some of the difficulties. For example, one might not instantly have all of the insight which lead to the second spreadsheet. In practice a session in which someone "discovers" this theorem might use many spreadsheets. All that matters is that one makes a little bit of progress with each step.

REFERENCES

[HSW] J. W. Helton, M. Stankus and J. Wavrik: "Computer simplification of engineering systems formulas," Conf. on Decision and Control, Orlando Florida December 1994.

[TMora] T. Mora, "An introduction to commutative and noncommutative Gröbner Bases", Theoretical Computer Science, Nov 7,1994, vol. 134 N1:131-173.

[NCA] J.W. Helton, R.L. Miller and M. Stankus, "NCAlgebra: A Mathematica Package for Doing Non Commuting Algebra" available from ncalg@ucsd.edu