Proceedings of the 35th
Conference on Decision and Control
Kobe, Japan • December 1996

Computer Assistance In Discovering Formulas And Theorems In System Engineering II *

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

ABSTRACT

[HSWcdc94] focused on procedures for simplifying complicated expressions automatically. [HScdc95] turned to the adventurous pursuit of developing a highly computer assisted method for "discovering" certain types of formulas and theorems.

It is often the case that some variables in the formulation of a problem are not the natural "coordinates" for solution of the problem. Gröbner Basis Algorithms, which lie at the core of our method, are very good at eliminating unknowns, but have no way of finding good changes of variables. This paper gives a way of incorporating changes of variables into our method.

As an example, we "discover" the DGKF equations of H^∞ control.

INTRODUCTION

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 noncommuting 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. As mentioned in the abstract, [HScdc95] turned to the adventurous pursuit of developing a highly computer assisted method for discovering certain types of formulas and 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.

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 useful for discovering 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.

Our method 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.

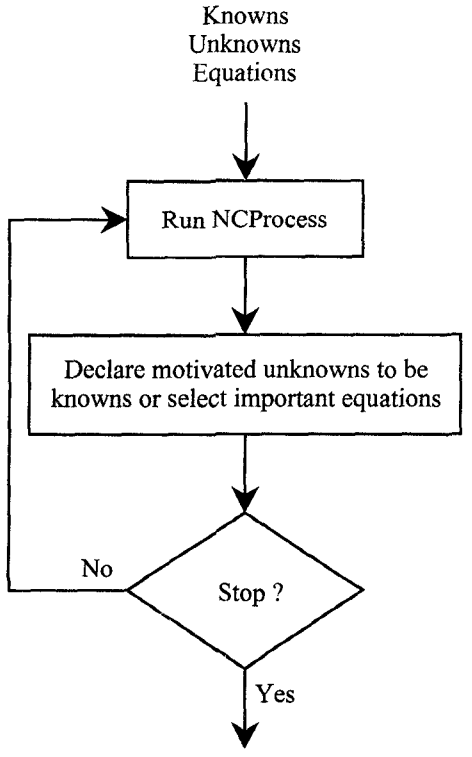
We begin by reviewing the basic method and then turn to our new changes of variable ideas [H-St].

1 What is a strategy?

At the highest level, a strategy consists of running a program called NCProcess which displays a sorted list of equations in LaTeX as its output. Then a person looks at the output and makes a decision which produces a new set of equations on which to run NCProcess again. A discussion of when the repetition ends is discussed in §1.4.

The flowchart for a strategy is:

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



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 discussed in §1.5.

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 the NCProcess commands is a list of equations which are mathematically equivalent to the equations in I3. These equations are presented to the user as

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

O2. Equations selected or created by the user. [1]

[1]These do not exist in the first run. A user-selected equation is a polynomial equation which the user has selected. When a user selects an equation, they are given the highest priority in eliminating other equations when NCProcess runs.

There are times during a strategy when one wants to introduce new variables and equations.

O3. Equations involving no unknowns.

O4. Equations involving only one unknown.

O5. Equations involving only 2 unknowns. etc.

We will call the formatted output described above a spreadsheet.

1.2 A simple example

We used our method to classify 3 projections in an algebra $\mathcal{A}$ whose sum is equal to a constant λ times the identity; strengthening an old result due to Joe Stampfli. Here we will just derive the solution for the case where $\lambda = 1$.

Note that A, B and C are projections whose sum is 1 if and only if the following four equations hold

$$A+B+C-1=0 \qquad AA-A=0$$
$$BB-B=0 \qquad CC-C=0\,.$$

The next consideration is which variables are to be considered known and which are unknowns. This problem is so simple that it does not matter how one selects knowns or unknowns. For illustrative purposes, we set A to be known and B and C to be unknown. The spreadsheet that NCProcess produces is the following.

YOUR SESSION HAS DIGESTED
THE FOLLOWING RELATIONS

THE FOLLOWING VARIABLES HAVE BEEN SOLVED FOR: $\{C\}$
The corresponding rules are the following:
$C \to 1 - A - B$

The expressions with unknown variables $\{\}$
and knowns $\{A\}$
$AA \to A$

USER CREATIONS APPEAR BELOW

SOME RELATIONS WHICH APPEAR BELOW
MAY BE UNDIGESTED

THE FOLLOWING VARIABLES HAVE NOT BEEN SOLVED FOR: $\{A, B\}$

The expressions with unknown variables $\{B\}$
and knowns $\{A\}$
$AB \to 0 \qquad BA \to 0 \qquad BB \to B$

This shows not only that A, B and C commute, but that the product of any two of them is zero.

1.3 Strategy

The idea of a *strategy* is :

S1. Run NCProcess which creates a display of the output (see O1-O5 in §1.1). Suppose that there is an equation which involves only one unknown (for example, a particular equation E_{17} contains only one unknown x_5) OR that there is an equation E_{18} which involves only one expression which contains unknowns (say $(x_1x_2 + x_3)$ — see §2).

S2. The user must now make a decision about equations in x_5 (e.g., E_{17} is a Riccati equation so I shall not try to simplify it, but solve it using Matlab) OR make a decision about equations in the equations which involve only one expression, $x_1x_2 + x_3$, which contains unknowns (I will introduce the new variable y and introduce the new equation $y = x_1x_2 + x_3$ to perform a change of variables – see §2.) Now the user declares the unknown x_5 (OR y) to be known.

S3. The process repeats.

S4. Knowing when a strategy stops is discussed in §1.4.

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 1 or 2 unknowns (OR equations involving 1 or 2 unknown expressions).

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

1.4 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 if and only if the Gröbner Basis 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.

1.5 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 usually they generate too many equations. It is our hope and our experience that the equations which it generates contain all of the equations essential to the 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. Our method addresses this problem in that 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 then followed by a command to be described in §2.3 for a "collecting" on knowns is what constitutes NCProcess.

1.6 Summary

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.

2 Changes of Variables and motivated unknowns

Changes of variables were introduced hastily in S1 and S2 of §1.3. Now we give more detail.

2.1 Decompose

Suppose that it can be shown algebraically that an expression, such as $x_1x_2 + x_3$, solved a Ricatti equation, e.g.,

$$(x_1x_2+x_3)a_1a_2(x_1x_2+x_3)+a_3(x_1x_2+x_3)+a_5a_6 = 0 \quad (1)$$

The left hand side of (1) depends on three unknowns x_1, x_2 and x_3 and so would not be an equation three unknowns, not one. It is natural, however, to view (1) as an equation in one new unknown y and to rewrite the left hand side of (1) as the composition

$$k(a_1, \ldots, a_6, q(a_1, \ldots, a_6, x_1, x_2, x_3))$$

where $q(a_1, \ldots, a_6, x_1, x_2, x_3) = x_1x_2 + x_3$ and

$$k(a_1, \ldots, a_6, y) = ya_1a_2y + a_3y + a_5a_6 \, .$$

In this example, we would call y a *motivated unknown.* Let *Decompose* denote the operation of of creating all non-trivial maximal compositions of a polynomial p. *Decompose*, therefore, produces motivated unknowns.

One can consider an analogous decomposition, called an ℓ-decomposition, into several variables and

thereby introduce new variables $y_1, \ldots, y_\ell$. Let ℓ-*Decompose* denote the operation of creating all nontrivial maximal ℓ-decompositions of a polynomial p.

A variant on 1-Decompose which we shall use frequently is called *symmetric 1-Decompose*. This applies in an algebra with involution, $w \to w^*$ for all w. For example, if $\mathcal{A}$ is a matrix algebra which is closed under transposes (or adjoints), then the operation transpose (or adjoint) is an involution on $\mathcal{A}$.

2.2 Left and Right Multiples

Suppose that a polynomial $p(a_1, \ldots, a_r, x_1, \ldots, x_s)$ appears on a spreadsheet and has the property that there are other monomials $L(a_1, \ldots, a_r, x_1, \ldots, x_s)$ and $R(a_1, \ldots, a_r, x_1, \ldots, x_s)$ for which LpR has a 1-decomposition

$$LpR = k(a_1, \ldots, a_r, q(a_1, \ldots, a_r, x_1, \ldots, x_s))$$

where k is a polynomial in one unknown. For mathematical reasons we do not describe, the polynomial LpR will *not* appear on the spreadsheet. We formalize this as follows.

Definition 2.1 *A polynomial p motivates an unknown y via the equation $y = q(a_1, \ldots, a_r, x_1, \ldots, x_s)$ if there exist monomials $L(a_1, \ldots, a_r, x_1, \ldots, x_s)$ and $R(a_1, \ldots, a_r, x_1, \ldots, x_s)$ and there exists a polynomial in one unknown $k(a_1, \ldots, a_r, y)$ such that $LpR = k(a_1, \ldots, a_r, q(a_1, \ldots, a_r, x_1, \ldots, x_s))$.*

Of course, from the perspective of finding zeros of collections of polynomials, if p has a zero, then LpR has a zero and so k has a zero. Since k is a polynomial in only one unknown variable, finding the zeros of k is bound to be easier than finding the zeros of p.

2.3 Implementation

The authors do not know how to fully implement the Decompose operation. Finding decompositions by hand can be facilitated with the use of a certain type of "collect" command. This "collect" command both assists the user in performing decompositions of a particular polynomial and helps in finding other polynomials in the ideal which would produce motivated unknowns.

This "collect" command "collects" knowns and products of knowns out of expressions. For example, suppose that A and B are knowns and X, Y and Z are unknowns. The collected form of

$$XABX+XABY+YABX+YABY+AX+AY \quad (2)$$

is

$$(X+Y)AB(X+Y)+A(X+Y). \quad (3)$$

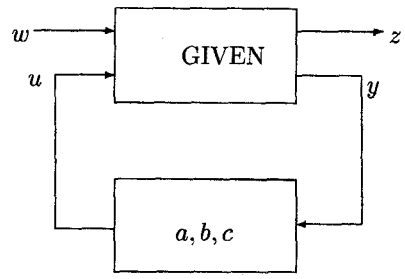
Clearly this suggests a decomposition of (2) and, indeed, the collect command helps find decompositions of much more complicated polynomials.

Next we give a demonstration of how collect enters the NCProcess commands.

3 Example:Solving the H^∞ Control Problem

In this section we give an example of solving a problem using a strategy.

A basic problem in systems engineering (H^∞ control) is to make a given system A, B_1, B_2, C_1, C_2, $D_{11} = 0 = D_{22}$, $D_{12} = 1 = D_{21}$ dissipative by designing a feedback law.



This corresponds to the existence of a nonnegative quadratic form (Ez, z) on the statespace of the closed loop system. It must satisfy a HJBI inequality which we abbreviate $H \leq 0$. Now we can only deal with equalities so we optimistically set out to find E which solves $H = 0$ and do not ask the computer to keep track of $E \geq 0$. Natural coordinates on the closed loop statespace are the statespace of the plant and of the compensator. In these coordinates E and H split into block 2 x 2 matricies. Setting $H = 0$ gives the following 4 equations below.

3.1 Problem statement

Let H_{xx}, H_{xz}, H_{zx} and H_{zz} be defined as follows.

$$\begin{aligned} H_{xx} &= E_{11}A + A^T E_{11} + C_1^T C_1 + E_{12}^T b C_2 + \\ &\quad C_2^T b^T E_{12}^T + E_{11} B_1 b^T E_{12}^T + \\ &\quad E_{11} B_1 B_1^T E_{11} + E_{12} b b^T E_{12}^T + E_{12} b B_1^T E_{11} \end{aligned}$$

$$\begin{aligned} H_{xz} &= E_{21}A + \frac{a^T(E_{21}+E_{12}^T)}{2} + c^T C_1 + E_{22} b C_2 + \\ &\quad c^T B_2^T E_{11}^T + \frac{E_{21} B_1 b^T (E_{21}+E_{12}^T)}{2} + \\ &\quad E_{21} B_1 B_1^T E_{11}^T + \frac{E_{22} b b^T (E_{21}+E_{12}^T)}{2} + \\ &\quad E_{22} b B_1^T E_{11}^T \end{aligned}$$

$$\begin{aligned} H_{zx} &= A^T E_{21}^T + C_1^T c + \frac{(E_{12}+E_{21}^T)a}{2} + E_{11} B_2 c + \\ &\quad C_2^T b^T E_{22}^T + E_{11} B_1 b^T E_{22}^T + E_{11} B_1 B_1^T E_{21}^T + \\ &\quad \frac{(E_{12}+E_{21}^T) b b^T E_{22}^T}{2} + \frac{(E_{12}+E_{21}^T) b B_1^T E_{21}^T}{2} \end{aligned}$$

$$\begin{aligned} H_{zz} &= E_{22} a + a^T E_{22}^T + c^T c + E_{21} B_2 c + \\ &\quad c^T B_2^T E_{21}^T + E_{21} B_1 b^T E_{22}^T + \\ &\quad E_{21} B_1 B_1^T E_{21}^T + E_{22} b b^T E_{22}^T + E_{22} b B_1^T E_{21}^T \end{aligned}$$

The math problem we address is:

(HGRAIL)

> *Let A, B_1, B_2, C_1, C_2 be matrices of compatible size be given. Solve $H_{xx} = 0$, $H_{xx} = 0$, $H_{xz} = 0$, $H_{zx} = 0$, and $H_{zz} = 0$ for a, b, c and for E_{11}, E_{12}, E_{21} and E_{22}. When can they be solved? If these equations can be solved, find formulas for the solution.*

Since we are trying to "discover" a result we proceed a bit like we would in a back of the envelope computation, e.g., we assume everything in sight, such as E_{ij}, is invertible. Later, after the main ideas have been discovered, the user can selectively relax them and thereby obtain more general results.

3.2 Solving (*HGRAIL*) using a Strategy

We now give a demonstration of how one discovers the algebraic part of the solution to this problem.

3.2.1 The Set-Up

The first step is to assemble all of the key polynomial equations in executable form:

The polynomials we shall input to NCProcess are naturally thought of in several groups. First, to enforce that the 2×2 matrix $(E_{i,j})_{i=1,2,j=1,2}$ is symmetric, we require each of the following polynomials to be zero:

$$\begin{array}{cccc} E_{11}^T - E_{11}, & E_{12}^T - E_{21}, & E_{21}^T - E_{12}, & E_{22}^T - E_{22}, \\ E_{11}^{-1T} - E_{11}^{-1}, & E_{12}^{-1T} - E_{21}^{-1}, & E_{21}^{-1T} - E_{12}^{-1}, & E_{22}^{-1T} - E_{22}^{-1} \end{array}$$

Since we assumed that E_{ij} is invertible for $i = 1, 2$ and $j = 1, 2$, we require the following polynomials to be zero:

$$\begin{array}{cccc} E_{11}^{-1}E_{11} - 1 & E_{12}^{-1}E_{12} - 1 & E_{21}^{-1}E_{21} - 1 & E_{22}^{-1}E_{22} - 1 \\ E_{11}^{T}E_{11}^{-1T} - 1 & E_{12}^{T}E_{12}^{-1T} - 1 & E_{21}^{T}E_{21}^{-1T} - 1 & E_{22}^{T}E_{22}^{-1T} - 1 \end{array}$$

Naturally we also assume the following polynomials are zero:

$$H_{xx}, \quad H_{xz}, \quad H_{zx}, \quad H_{zz}$$

The knowns in this example are A, A^T, B_1, B_1^T, B_2, B_2^T, C_1, C_1^T, C_2 and C_2^T. The unknowns in this problem are E_{12}, E_{12}^T, E_{21}, E_{21}^T, E_{22}, E_{22}^T, E_{11}, E_{11}^T, E_{12}^{-1}, E_{12}^{-1T}, E_{21}^{-1}, E_{21}^{-1T}, E_{22}^{-1}, E_{22}^{-1T}, E_{11}^{-1}, E_{11}^{-1T}, b, b^T, c, c^T a and a^T.

We ran NCProcess for 2 iterations with the "collect" operation (described above) applied to each equation.

The algorithm did not run the full two iterations but finished after one. Our program produced a message saying that, in fact, the output is a Gröbner Basis.

3.2.2 Step 1: Process and Collect

There is no point in listing the full output here. Indeed, the only nontrivial undigested polynomial equations are:

The expressions with unknown variables
$\{b^T, b, E_{21}^{-1}, E_{12}^{-1}, E_{11}\}$
and knowns $\{A, B_1, C_1, C_2, A^T, B_1^T, C_1^T, C_2^T\}$

$$b\,b^T + b\,C_2\,E_{21}^{-1} + E_{12}^{-1}\,C_2^T\,b^T + E_{12}^{-1}\,E_{11}\,A\,E_{21}^{-1} + E_{12}^{-1}\,E_{11}\,B_1\,b^T + E_{12}^{-1}\,A^T\,E_{11}\,E_{21}^{-1} + E_{12}^{-1}\,C_1^T\,C_1\,E_{21}^{-1} + (b + E_{12}^{-1}\,E_{11}\,B_1)\,B_1^T\,E_{11}\,E_{21}^{-1} = 0$$

The expressions with unknown variables
$\{c^T, c, E_{21}^{-1}, E_{12}^{-1}, E_{11}, E_{22}, E_{21}, E_{12}\}$
and knowns $\{A, B_1, B_2, C_1, A^T, B_1^T, B_2^T, C_1^T\}$

$$c^T\,c + (E_{21} - E_{22}\,E_{12}^{-1}\,E_{11})\,B_2\,c + c^T\,B_2^T\,(E_{12} - E_{11}\,E_{21}^{-1}\,E_{22}) - E_{22}\,E_{12}^{-1}\,A^T\,(E_{12} - E_{11}\,E_{21}^{-1}\,E_{22}) - E_{22}\,E_{12}^{-1}\,C_1^T\,(c - C_1\,E_{21}^{-1}\,E_{22}) - (E_{21} - E_{22}\,E_{12}^{-1}\,E_{11})\,A\,E_{21}^{-1}\,E_{22} + (E_{21} - E_{22}\,E_{12}^{-1}\,E_{11})\,B_1\,B_1^T\,(E_{12} - E_{11}\,E_{21}^{-1}\,E_{22}) - c^T\,C_1\,E_{21}^{-1}\,E_{22} = 0$$

3.2.3 Step 2: The user attacks

Now the reader must apply his expertise to the nontrivial polynomial equations left undigested by the NCProcess command. A key observation is that the first key polynomial equation contains b but not c and the second key polynomial equation contains c but not b. In other words, b and c appear in decoupled equations.

Observe that the first key polynomial from the above spreadsheet is quadratic in b. We could complete the square and put the polynomial in the form

$$(b + \mu)(b^T + \mu^T) + \nu \tag{4}$$

where μ and ν are expressions involving C_2, C_2^T, B_1, B_1^T, A, A^T, E_{21}^{-1}, E_{12}^{-1} and E_{11}. Since there are many unknowns in the problem, there is probably excess freedom. Let us investigate what happens when we take $b + \mu = 0$. This yields the polynomial equation

$$b = -E_{12}^{-1}\,C_2^T - E_{12}^{-1}\,E_{11}\,B_1 \tag{5}$$

which we could add to the input for NCProcess in Step 3. We can also complete the square for the expression in c and put that expression in the form

$$(c + \lambda)(c^T + \lambda^T) + \gamma\,. \tag{6}$$

We also assume that $c + \lambda = 0$. This defines c by the following equation

$$c = -B_2^T\,E_{12} + C_1\,E_{21}^{-1}\,E_{22} + B_2^T\,E_{11}\,E_{21}^{-1}\,E_{22}\,. \tag{7}$$

Since we have now solved for b and c, we can use these equations to solve for b^T and c^T.

As we shall see, this is the step which requires the most human intervention. The reason is that the original problem, has a nonunique solution and the user must make a decision which eliminates some of the degrees of freedom.

3.2.4 Step 3

The equations for this step will be the output from the first call to NCProcess (above) as well as the four new equations that we have just derived. We ran NCProcess for two iterations.

Once again we go directly to the spreadsheet which NCProcess created. There is no need to record all of it here, since at this stage we shall be concerned only with the undigested polynomial equations. There are only two undigested polynomial equations which are not banal.

The expressions with unknown variables $\{E_{11}\}$
and knowns $\{A, B_1, C_1, C_2, A^T, B_1^T, C_1^T, C_2^T\}$

$$E_{11}\,B_1\,C_2 \to E_{11}\,A + A^T\,E_{11} + C_1^T\,C_1 - C_2^T\,C_2 - C_2^T\,B_1^T\,E_{11}$$

The expressions with unknown variables
$\{E_{21}^{-1}, E_{12}^{-1}, E_{11}, E_{22}, E_{21}, E_{12}\}$
and knowns $\{A, B_1, B_2, C_1, A^T, B_1^T, B_2^T, C_1^T\}$

$$E_{22}\,E_{12}^{-1}\,A^T\,(E_{12} - E_{11}\,E_{21}^{-1}\,E_{22}) + (E_{21} - E_{22}\,E_{12}^{-1}\,E_{11})\,A\,E_{21}^{-1}\,E_{22} - (E_{21} - E_{22}\,E_{12}^{-1}\,E_{11})\,B_1\,B_1^T\,(E_{12} - E_{11}\,E_{21}^{-1}\,E_{22}) + (E_{21} - E_{22}\,E_{12}^{-1}\,E_{11})\,B_2\,B_2^T\,(E_{12} - E_{11}\,E_{21}^{-1}\,E_{22}) - E_{22}\,E_{12}^{-1}\,C_1^T\,B_2^T\,(E_{12} - E_{11}\,E_{21}^{-1}\,E_{22}) - (E_{21} - E_{22}\,E_{12}^{-1}\,E_{11})\,B_2\,C_1\,E_{21}^{-1}\,E_{22} = 0$$

3.2.5 Step 4

Now we analyze these two polynomial equations.

The first polynomial equation is a (Ricatti-Lyapunov) equation in E_{11}. Numerical methods for solving Ricatti equations are common. For this reason assuming that a Ricatti equation has a solution is a socially acceptable necessary condition throughout control engineering. Thus we can consider E_{11} to be known. (Indeed, it is the Y^{-1} Ricatti of DGKF.)

A first glance at the second equation reveals that the same products of unknowns appear over and over. Also we can see that this equation is symmetric. It would not take an experienced person long to realize that by multiplying this equation on the left by $E_{12}\,E_{22}^{-1}$ and on the right by $E_{22}^{-1}\,E_{21}$, we will have an equation in one unknown, $E_{11} - E_{12}\,E_{22}^{-1}\,E_{21}$.

Now we can replace $E_{11} - E_{12}\,E_{22}^{-1}\,E_{21}$ with a new variable X. There are two ways of proceeding at this point. The first is to compute that the above equations yields

$$-X\,A - A^T\,X + X\,B_2\,C_1 + C_1^T\,B_2^T\,X - X\,B_1\,B_1^T\,X + X\,B_2\,B_2^T\,X$$

Observe that this is an equation in the one unknown X. (Indeed, it is the X Ricatti of DGKF.) The second approach is to use NCProcess. In particular, create new knowns, X and Y (which at this point we take to be invertible) and add the equations

$$X - (E_{11} - E_{12}E_{22}^{-1}E_{21}) = 0 \qquad Y^{-1} - E_{11} = 0\,.$$

We then run NCProcess. The output of NCProcess contains the list of equations below. To save space in the presentation we go to the next step where we turn on the strongest algorithm for removing redundant equations inside of NCProcess (see §1.5) and obtain

---

YOUR SESSION HAS DIGESTED
THE FOLLOWING RELATIONS

THE FOLLOWING VARIABLES HAVE BEEN SOLVED FOR:
$\{a, b, c, E_{11}, E_{11}^{-1}, a^T, b^T, c^T, E_{11}^T, E_{12}^T, E_{21}^T, E_{22}^T, E_{11}^{-1T}, E_{12}^{-1T}, E_{21}^{-1T}, E_{22}^{-1T}\}$

The corresponding rules are the following:

$$a \to -E_{12}^{-1}\,A^T\,E_{12} + E_{12}^{-1}\,C_1^T\,B_2^T\,E_{12} + E_{12}^{-1}\,C_2^T\,B_1^T\,E_{12} + E_{12}^{-1}\,E_{11}\,B_2\,B_2^T\,E_{12} - E_{12}^{-1}\,C_1^T\,C_1\,E_{21}^{-1}\,E_{22} - E_{12}^{-1}\,E_{11}\,B_2\,C_1\,E_{21}^{-1}\,E_{22} - E_{12}^{-1}\,C_1^T\,B_2^T\,E_{11}\,E_{21}^{-1}\,E_{22} - E_{12}^{-1}\,E_{11}\,B_2\,B_2^T\,E_{11}\,E_{21}^{-1}\,E_{22}$$

$$b \to -E_{12}^{-1}\,C_2^T - E_{12}^{-1}\,E_{11}\,B_1$$

$$c \to -B_2^T\,E_{12} + C_1\,E_{21}^{-1}\,E_{22} + B_2^T\,E_{11}\,E_{21}^{-1}\,E_{22}$$

$$E_{11} \to Y^{-1}$$

$$E_{11}^{-1} \to Y$$

$$a^T \to -E_{21}\,A\,E_{21}^{-1} + E_{21}\,B_1\,C_2\,E_{21}^{-1} + E_{21}\,B_2\,C_1\,E_{21}^{-1} + E_{21}\,B_2\,B_2^T\,E_{11}\,E_{21}^{-1} - E_{22}\,E_{12}^{-1}\,C_1^T\,C_1\,E_{21}^{-1} - E_{22}\,E_{12}^{-1}\,E_{11}\,B_2\,C_1\,E_{21}^{-1} - E_{22}\,E_{12}^{-1}\,C_1^T\,B_2^T\,E_{11}\,E_{21}^{-1} - E_{22}\,E_{12}^{-1}\,E_{11}\,B_2\,B_2^T\,E_{11}\,E_{21}^{-1}$$

$$b^T \to -C_2\,E_{21}^{-1} - B_1^T\,E_{11}\,E_{21}^{-1}$$

$$c^T \to -E_{21}\,B_2 + E_{22}\,E_{12}^{-1}\,C_1^T + E_{22}\,E_{12}^{-1}\,E_{11}\,B_2$$

$$E_{11}^T \to E_{11} \quad E_{12}^T \to E_{21} \quad E_{21}^T \to E_{12} \quad E_{22}^T \to E_{22}$$
$$E_{11}^{-1T} \to E_{11}^{-1} \quad E_{12}^{-1T} \to E_{21}^{-1} \quad E_{21}^{-1T} \to E_{12}^{-1} \quad E_{22}^{-1T} \to E_{22}^{-1}$$

The expressions with unknown variables $\{\}$
and knowns
$\{A, B_1, B_2, C_1, C_2, X, Y, X^{-1}, Y^{-1}, A^T, B_1^T, B_2^T, C_1^T, C_2^T\}$

$$X\,X^{-1} \to 1 \quad X^{-1}X \to 1 \quad Y\,Y^{-1} \to 1 \quad Y^{-1}Y \to 1$$

$$Y^{-1}\,B_1\,C_2 \to Y^{-1}\,A + A^T\,Y^{-1} + C_1^T\,C_1 - C_2^T\,C_2 - C_2^T\,B_1^T\,Y^{-1}$$

$$X\,B_2\,B_2^T\,X \to X\,A + A^T\,X - X\,B_2\,C_1 - C_1^T\,B_2^T\,X + X\,B_1\,B_1^T\,X$$

USER CREATIONS APPEAR BELOW

$$E_{11}^{-1} \to Y$$

$$E_{12}\,E_{22}^{-1}\,E_{21} \to E_{11} - X$$

We have left out the undigested portion of the spreadsheet, which contains only equations which give the definition of the inverses.

3.3 End game

Now let us compare what we have found to the well known solution of (*HGRAIL*). In that theory there are two Ricatti equations due to Doyle, Glover, Khargonekar and Francis. These are the DGKF X and Y equations. One can read these off from the equations involving knowns.

Indeed what we have discovered, up to assuming that many things are invertible, is that if (HGRAIL) has a solution and if b and c are given by formulas (5) and (7), then

(1) the DGKF X and Y^{-1} equations must have a solution

(2) X and Y are self-adjoint

Now we turn to the converse. The straightforward converse of the above italicized statement would be: If items (1), (2) and invertibility hold, then (*HGRAIL*) has a solution and b and c are given by formulas (5) and (7). There is no reason to believe (and it is not the case) that b and c *must* be given by the formulas (5) and (7). These two formulas came about in §3.2.3 and were motivated by "excess freedom" in the problem. A plausible converse is:

Proposed Converse 3.2 *If items (1) and (2) above hold and Y, X, $Y^{-1} - X$ are invertible, then (HGRAIL) has a solution.*

One can use the above spreadsheet to heavily assist in proving the proposed converse. Space limitations prevent us from doing that carefully here. The point is that the groupings in the last spreadsheet correspond to the sequence in which we must back solve the system of equations.

3.4 Retrospective

Now we discuss the level of difficulty of the discovery of the DGKF equations.

The first spreadsheet featured an equation in the single unknown b (and its transpose) and an equation in the single unknown c (and its transpose) and so is the most complicated. For example, the expression (4) decomposes as

$$p = q_1^T q_1 + q_2 \tag{8}$$

where $q_1 = b + E_{12}^{-1} C_2^T + E_{12}^{-1} E_{11} B_1$ and q_2 is a symmetric polynomial which does not involve b. This forces us to say that the proof of the necessary side was done with a symmetrized 2-strategy.

A more aggressive way of selecting knowns and unknowns would allow us to obtain this same result with a symmetrized 1-strategy.

Thanks

We would like to thank the referee for pointing out that the idea that commutative ideal theory (Ritt and Gröbner bases) could be applied to Theorem Proving in geometry and differential analysis goes back to [**T1**] and [**T2**].

REFERENCES

[DGKF] J. C. Doyle, K. Glover, P. P. Khargonekar and B. A. Francis, "State–space solutions to standard H_2 and H_∞ control problems", IEEE Trans. Auto. Control 34 (1989), 831–847.

[HScdc95] J. W. Helton and M. Stankus: "Computer Assistance In Discovering Formulas and Theorems In System Engineering", Conf. on Decision and Control, New Orleans, December 1995.

[H-St] J. W. Helton and M. Stankus: "Computer Assistance for "discovering" formulas in system engineering and operator theory", preprint pp. 1-88.

[HSWcdc94] J. W. Helton, M. Stankus and J. Wavrik: "Computer simplification of engineering systems formulas," Conf. on Decision and Control, Orlando Florida December 1994.

[NCA] J.W. Helton, R.L. Miller and M. Stankus, "NCAlgebra: A Mathematica Package for Doing Non Commuting Algebra" available from ncalg@ucsd.edu or visit http://math.ucsd.edu/~mstankus or http://math.ucsd.edu/~helton

[NCGBDoc] J.W. Helton and M. Stankus, 'NonCommutative Gröbner Basis Package" available from ncalg@ucsd.edu or visit http://math.ucsd.edu/~mstankus or http://math.ucsd.edu/~helton

[T1]Wu Wen-Ts¥: "On the decision problem and the mechanization of theorem proving in elementary geometry", Scientia Sinica 21 (1978) 159–172

[T2] Wu Wen-Ts¥: "Some recent advances in mechanical theorem proving of geometry", AMS Contemporary Math. 29 235–241.