Computer Simplification of Formulas in Linear Systems Theory

J. William Helton, Mark Stankus, John J. Wavrik

Abstract— **Currently, the three most popular commercial computer algebra systems are Mathematica, Maple and MACSYMA. These systems provide a wide variety of symbolic computation facilities for commutative algebra and contain implementations of powerful algorithms in that domain. The Gröbner Basis Algorithm, for example, is an important tool used in computation with commutative algebras and in solving systems of polynomial equations.**

On the other hand, most of the computation involved in linear control theory is performed on matrices, and these do not commute. A typical issue of IEEE TAC is full of linear systems and computations with their coefficient matricies A B C D's or partitions of them into block matrices. Mathematica, Maple and MACSYMA are weak in the area of non-commutative operations. They allow a user to declare an operation to be non-commutative, but provide very few commands for manipulating such operations and no powerful algorithmic tools.

It is the purpose of this article to report on applications of a powerful tool: a non-commutative version of the Gröbner Basis Algorithm. The commutative version of this algorithm is implemented in most major computer algebra packages. The noncommutative version is relatively new [FMora].

. INTRODUCTION

Part One of the paper introduces Gröbner Bases (GB) and lays the foundation for simplification of complicated algebraic expressions from engineering and other applications. In Part Two we shall describe the Gröbner Bases for several elementary situations which arise in systems theory. These GB give (in a sense to be made precise) a "complete" set of simplifying rules for formulas which arise in these situations. We have found that this process provides a practical means of simplifying expressions.

We begin Part Three with an illustration of how the simplification rules from Part One apply in system theory. Section IV illustrates the use of Gröbner Basis with an application to the Doyle-Glover-Khargonekar-Francis theory of H^∞ control. The rest of Part Three explores other facets of the Gröbner Basis machinery.

The simplification process and the Gröbner Basis depend on the choice of an ordering on variables. In Section V we will examine the effect of changing the ordering on the GB which arise in connection with Lyapunov equations .

In addition to providing an approach to simplifying complex expressions, the Gröbner Basis Algorithm can be used to generate new and (sometimes) interesting equations from equations that are the statements of the basic assumptions. For example, the Youla-Tissi formulas involving the intertwining of the controllability and observability operators of a system arise as a subset of the Gröbner bases studied in Section VI.

Department of Mathematics, University of California, San Diego

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

The research required the use of software suited for computing with non-commuting symbolic expressions. Most of the research was performed using a special-purpose system developed by J. Wavrik. This system uses a new approach to the creation of support software for mathematical research. It provides the flexibility needed for experimentation with algorithms, data representation, and data analysis.

In an effort to make available computational facilities for work in non-commutative algebras to a wider audience, the other authors have written a collection of packages for Mathematica called NCAlgebra (available from ncalg@osiris.ucsd.edu). NCAlgebra has a number of commands for manipulating non-commuting expressions which are named and designed to be noncommutative analogs of Mathematica's built in commands. We have incorporated in these packages many of the results on simplification obtained from this research.

PART ONE: SIMPLIFYING EXPRESSIONS

I. Simplification

The problem of simplification involves a collection of expressions and a notion of equivalence among expressions. A goal is to obtain expressions equivalent to a given expression which are simpler in some way. Another goal is to find a unique representation for equivalent expressions.

When matrix expressions are simplified by hand, they are scanned for subexpressions which can be replaced by something which is equivalent and simpler. For example, the expression $1 - x^{-1}x + x^2$ simplifies to x^2 because the subexpression $x^{-1}x$ can be replaced by 1, and then $1 - 1$ can be replaced by 0.

In the case above, the occurrence of a matrix expression next to its inverse leads to a rather obvious simplification. Simplification can also use less obvious replacements. For example, $(1 - yx)^{-1} - x^{-1}(1 - xy)^{-1}x$ simplifies to 0 even though no subexpression consists of a matrix adjacent to its inverse. Here we use the fact that $(1-xy)^{-1}x$ is equivalent to $x(1 - yx)^{-1}$ so that

$$\begin{aligned}(1 - yx)^{-1} - x^{-1}(1 - xy)^{-1}x \\ \rightarrow (1 - yx)^{-1} - x^{-1}x(1 - yx)^{-1} \\ \rightarrow (1 - yx)^{-1} - (1 - yx)^{-1} = 0\end{aligned}$$

The simplification rule $(1-xy)^{-1}x \rightarrow x(1-yx)^{-1}$ is well known – but it is not quite as obvious. We will show that this rule and others of this sort are generated automatically by the simplification technology introduced in this paper.

We will examine several classes of matrix expressions. In each case we will start with a few simple matrix expressions and variables like x, x^{-1}, $(1-x)^{-1}$, and $(1-xy)^{-1}$ which we will call elementary or **atomic** expressions. **Polynomial expressions** are those which can be obtained by repeatedly performing arithmetic operations (addition, subtraction, multiplication and multiplication by scalars) on these atomic expressions. [1] Very complex expressions can be obtained in this way. It is quite possible for two expressions which look quite different to be equivalent in the sense that they represent the same matrix. We would like to find the simplest possible among these representations.

We are drawing a distinction among, for example, the polynomial expressions xx^{-1}, $x^{-1}x$, and 1. We regard these as different expressions because they are formed differently in terms of atomic expressions and operations. They look different. While they are not identical expressions, they are **(strongly) equivalent**[2] in the sense that, for any choice of invertible matrix x they assume the same value.

Simplification depends on a concept of simplicity and a concept of equivalence. A **simplifier** is a procedure, $\mathcal{S}$, which takes any expression E to an equivalent and simpler expression $\mathcal{S}(E)$. Thus we have $\mathcal{S}(E) \sim E$ (indicating that $\mathcal{S}(E)$ is equivalent to E) and $\mathcal{S}(E) \leq E$ (indicating that $\mathcal{S}(E)$ is simpler, in some sense, than E). In this paper we will discuss a simplifier which is based on a non-commutative version of the Gröbner Basis Algorithm. It can be implemented on a computer and shows evidence of being a very valuable tool. The precise notions of simplicity and equivalence which we use are discussed in Section I-C. We have been able to show, in the cases we have examined, that this simplifier is a **canonical simplifier** in the sense that $E_1 \sim E_2 \Longleftrightarrow \mathcal{S}(E_1) = \mathcal{S}(E_2)$. The expression $\mathcal{S}(E)$ is a **canonical form** for E. In other words, equivalence of expressions can be tested by reducing them to the canonical form. A test for equivalence of expressions is a major application for simplification machinery.

The problem of automatically simplifying matrix expressions and our use of the non-commutative Gröbner technology for this purpose are quite new. Simplification plays a fundamental role in computer algebra. A treatment of the general theory of simplification including an extensive set of references is found in the survey article of Buchberger and Loos [**BL**]. A good general reference on commutative Gröbner Basis is [**CLS**] and on non-commutative Gröbner Basis is [**TMora**].

A. Replacement Rules

For a collection of atomic expressions, there is usually a collection of replacement rules coming from obvious relations among the expressions. Thus we have rules like $xx^{-1} \to 1$ and $x^{-1}x \to 1$ which allow us to replace a matrix which occurs next to its inverse. The initial set of rules is insufficient for producing major simplifications. Crucial to our simplification procedures is a mechanism for extending a set of simplification rules.

A **replacement rule** consists of a left hand side (LHS), which will always be a monomial, and a right hand side (RHS) which will always be a polynomial. A replacement rule is applied to an expression by scanning its terms to find a match for LHS. If we find a term which has LHS as a factor, we replace the factor by RHS. Our notation for a rule is $LHS \to RHS$. Thus, for example, we have the replacement rule $xx^{-1} \to 1$.

Naturally we are unwilling to substitute RHS for LHS unless these are equivalent. Thus we require that

$$LHS - RHS \text{ becomes } 0$$

whenever matrices are substituted for the matrix variables that occur in the atomic expressions. $xx^{-1} \to 1$ is a valid replacement rule since $xx^{-1} - 1$ becomes 0 whenever x is replaced by an invertible matrix and x^{-1} by its inverse. $(1-xy)^{-1}x \to x(1-yx)^{-1}$ is also a valid replacement rule (since $(1-xy)^{-1}x - x(1-yx)^{-1}$ becomes 0 whenever x and y are replaced by matrices for which this makes sense)[3].

Simplification of a polynomial expression, p, using a list, $\mathcal{L}$, of replacement rules involves repeatedly applying rules in the list until we arrive at an expression which is **irreducible** (no further rules on the list are applicable). In Section I-C.2 we will place an ordering on the terms of polynomial expressions. Our replacement rules will always have the property that LHS is greater than any of the terms in RHS in this ordering. This will guarantee that (1) repeated application of the rules eventually leads to an irreducible expression and that (2) the irreducible expression is *simpler* in the sense of having terms of smaller order than the original. If h is irreducible and obtained from p by applying reduction rules in list $\mathcal{L}$, we will say that h is a **normal form** of p and write $h = NForm(p, \mathcal{L})$ or $p \to_{\mathcal{L}} h$. In general, the normal form is not unique[4].

We will now look at an example of simplification. This will illustrate the process and show what can occur. Here is a list of simplification rules based on the definition of inverse for the atomic expressions x, x^{-1} and $(1-x)^{-1}$:

Rule 1	$xx^{-1} \to 1$
Rule 2	$x^{-1}x \to 1$
Rule 3	$x(1-x)^{-1} \to (1-x)^{-1} - 1$
Rule 4	$(1-x)^{-1}x \to (1-x)^{-1} - 1$

Example 1: We now apply these rules to the expression:

$$x^{-1}x(1-x)^{-1} - x^{-1}(1-x)^{-1} - (1-x)^{-1} \qquad (2)$$

We first apply Rule 3 to the first term. This produces

$$x^{-1}((1-x)^{-1} - 1) - x^{-1}(1-x)^{-1} - (1-x)^{-1} \qquad (3)$$

[1] For example, $x^2(1-x)^{-1}$ and $2+x+3x^{-1}$ are polynomial expressions in the atomic expressions x, x^{-1} and $(1-x)^{-1}$, while $sin(x)$ and $x/(2+x)$ are not polynomial expressions in x, x^{-1} and $(1-x)^{-1}$.

[2] We will just say equivalent until we need to distinguish this from another concept of equivalence.

[3] As mentioned above, this classical rule can be automatically derived from simpler rules. See Section I-D.

[4] If the reduction rules are applied in different sequences, different normal forms can be obtained (unless $\mathcal{L}$ has special properties). Even though the normal form is not uniquely defined, the notation $h = NForm(p, \mathcal{L})$ is commonly used.

which, after expanding, rearranging, and cancelling terms, produces the result:

$$-x^{-1} - (1-x)^{-1}\,. \tag{4}$$

None of the rules apply to this expression and, therefore, it is irreducible. Thus (4) is a normal form for (2).

B. Complete lists of rules

We come to one of the more basic points which is the concept of 'completeness' of a list of rules.

The reader may have noticed that there are other possibilities for applying the replacement rules to (2). If we first apply Rule 2 then we get

$$(1-x)^{-1} - x^{-1}(1-x)^{-1} - (1-x)^{-1}\,. \tag{5}$$

This, after rearranging and cancelling terms, becomes

$$-x^{-1}(1-x)^{-1}\,. \tag{6}$$

We obtain two different expressions, (4) and (6), just by changing the sequence in which rules are applied. As a result, we obtain two expressions which are equivalent but which cannot be reduced to a common irreducible form. The difference of these two expressions is

$$x^{-1}(1-x)^{-1} - x^{-1} - (1-x)^{-1}\,. \tag{7}$$

It is equivalent to 0 but cannot be reduced to 0 by repeated application of the rules. A set of rules will be called **complete** if it is sufficient for simplifying to 0 all expressions which are actually equivalent to 0. The set of Rules 1-4 is not complete because they are not enough to simplify the expression (7) to 0.

This problem can be handled by enlarging the set of rules. The expression (7) does not reduce to 0 using the current set of rules, so we add it to the list of rules. The two rules we obtain in this way are:

Rule 5 $\quad x^{-1}(1-x)^{-1} \to x^{-1} + (1-x)^{-1}$
Rule 6 $\quad (1-x)^{-1}x^{-1} \to x^{-1} + (1-x)^{-1}$

Incidentally, Rules 5 and 6 are often called the resolvent identities.

It will follow from Section II-A that this expanded list has several important special properties. First of all, the expanded set of rules Rule 1-6 is **complete**[5]. Such a complete set of rules corresponds to something called a Gröbner basis (or GB) for the relations on x, x^{-1} and $(1-x)^{-1}$ (with respect to the given ordering); this will be discussed later. Secondly, if the full list is of rules is applied repeatedly to any polynomial p in x, x^{-1} and $(1-x)^{-1}$, then one obtains a particular irreducible polynomial q. The same polynomial q is obtained regardless of the sequence in which the rules are applied. We can show, in this case, that q is a canonical form for p with respect to algebraic equivalence[6]: decisions about algebraic equivalence of expressions p can be made by comparing canonical forms q.

We have illustrated the idea of expanding a set of simplification rules to find a complete set for a particular example. We now provide a more formal description of this process in general.

C. Formal Description

It can become very confusing if we sometimes regard $xx^{-1}$and $x^{-1}x$ as different, and, at other times, treat them as the same. We can understand what is at issue here by introducing a bit of formalism. This section will also make precise the concepts of simplicity and equivalence used in this work.

C.1 Polynomials

We will make a polynomial ring with one (noncommuting) variable for each of our atomic expressions. Let's continue with the example from the previous section where the atomic expressions are x, x^{-1} and $(1-x)^{-1}$. We introduce three polynomial variables a, b and c. Since the variables do not commute, ab and ba are different polynomials. Now take a polynomial in a, b and c and substitute a matrix M for a, M^{-1} for b and $(1-M)^{-1}$ for c. The result is a matrix. The result of substituting into ab is MM^{-1} while the result of substituting into ba is $M^{-1}M$. The polynomials are different but the resulting matrices are the same. We obtain, in this way, one notion of equivalence on polynomials: two polynomials are **(strongly) equivalent** if, upon any meaningful substitution, they produce the same matrix. Here are some polynomials that are equivalent to 0 with the substitution as in this example:

$$ab-1,\ ba-1,\ ac-c+1,\ cb-c-b \tag{8}$$

because, on substitution, they become

$$\begin{aligned} &MM^{-1}-1,\ M^{-1}M-1,\\ &M(1-M)^{-1}-(1-M)^{-1}+1,\\ &(1-M)^{-1}M^{-1}-(1-M)^{-1}-M^{-1} \end{aligned}$$

all of which are 0 for any matrix M for which they make sense. A polynomial which is equivalent to zero (in the context of association of the polynomial's variables with atomic expressions) is said to be a **relation** on the variables. Therefore, in our example, $ab-1$, $ba-1$, $ac-c+1$ and $cb-c-b$ are relations on a, b and c.

Let $\mathcal{P}$ denote all polynomials in three noncommuting variables a, b and c. Define

$$p_1 = ab-1,\ p_2 = ba-1,\ p_3 = ac-c+1,\ p_4 = ca-c+1 \tag{9}$$

and observe that if we substitute M for a, M^{-1} for b and $(1-M)^{-1}$ for c, then these polynomials become 0 and in fact constitute the definitions of the expressions x^{-1} and $(1-x)^{-1}$. These relations correspond to the simplification rules Rules 1-4. In practice it is only easy to determine that some very simple polynomials are relations. p_1, p_2, p_3 and p_4 are relations which result from the definition of inverse. We often show that a more complicated polynomial is a

[5]This is not obvious. It follows from the fact that the Mora Algorithm terminates in this case (see Section I-D).

[6]We will discuss algebraic equivalence in Section I-C.3. It depends on the choice of starting rules.

relation by showing that it is an "algebraic consequence" of known relations. For example, $p = bc - b - c$ is a relation because:

$$p = bc - b - c = p_2 c - b p_3 \tag{10}$$

Notice that any matrices substituted for a,b, and c which make p_2 and p_3 zero will also make p zero. Thus p is a relation since p_2 and p_3 are relations and p is an "algebraic consequence" of p_2 and p_3.[7]

Since our work involves the notion of "algebraic consequence" we will introduce some terminology to make it precise. Let $\mathcal{P}$ be the set of polynomials in a fixed finite collection of non-commuting variables. Recall that an **ideal** of $\mathcal{P}$ is a subset I of $\mathcal{P}$ such that whenever p_1 and p_2 are in I and whenever q_1 and q_2 are in $\mathcal{P}$, both $p_1 + p_2$ and $q_1 p q_2$ are in I. The **ideal generated by a set of polynomials** S is the smallest ideal of $\mathcal{P}$ containing S. This ideal consists of finite sums of the form $\sum r_i p_i s_i$ where r_i, s_i are any polynomials and $p_i \in S$.

Suppose that S is a finite set of polynomials,

$$S = \{p_1, ..., p_n\},$$

I is the ideal generated by S and $f \in I$. If any set of matrices M_j satisfy the equations $p_i(M_1, ..., M_m) = 0$ then they also satisfy $f(M_1, ..., M_m) = 0$. If the p_i are relations on some atomic expressions, then f will also be a relation on these expressions. We say that f is an **algebraic consequence** of $p_1, ..., p_n$ if f is in the ideal generated by the p_i. An ideal is the set of all algebraic consequences of a starting set of polynomials.

Notation: Using the strict polynomial notation, as we have above, makes it hard to remember which atomic expression is associated with which variable. We have found it convenient to use the associated atomic expressions as names for the polynomial variables. Thus, in the case above, we would use x rather than a and x^{-1} rather than b. x should be thought of as a variable for which matrices can be substituted. $xx^{-1} - 1$ should be thought of as a polynomial in two variables which is not zero (as a polynomial) but which becomes zero when any matrix and its inverse are substituted for the variables. We will always specify in advance which atomic expressions are being used.

C.2 Ordering

A replacement rule $LHS \rightarrow RHS$ gives rise to a relation $LHS - RHS$. A relation, on the other hand, could give rise to several possible replacement rules. For example, the definition of $(1-x)^{-1}$ gives the relation

$$x(1-x)^{-1} - (1-x)^{-1} + 1 \tag{11}$$

The 3 replacement rules we could associate to this are

$$\begin{aligned} 1 &\rightarrow (1-x)^{-1} - x(1-x)^{-1} \\ (1-x)^{-1} &\rightarrow x(1-x)^{-1} + 1 \\ x(1-x)^{-1} &\rightarrow (1-x)^{-1} - 1 \end{aligned}$$

We wish to use the rule to make expressions less complicated so we choose the last rule which replaces the "most complicated" monomial in (11) by a sum of simpler ones. A choice of a particular replacement rule for any relation is made by placing an ordering on the terms in expressions. The ordering will be chosen so that expressions which we subjectively regard as complicated tend to be higher in the order than those which we think of as simpler. Once an ordering is imposed, each relation has a term of highest order. We associate to a relation that replacement rule for which LHS is the term of highest order.

Let us assume that the variables for the polynomial ring $\mathcal{P}$ are $a, b, c, ...$ (one letter for each atomic expression). The monomials of $\mathcal{P}$ are words in the letters $a, b, c....$ We place an ordering on these monomials by:

$M \leq N$ if and only if

either	$length(M) < length(N)$
or	$length(M) = length(N)$ and
	M comes before N in the dictionary

This is called **graded lexicographic ordering** of the monomials.

If $a < b < c$ then the monomials of degree 3 are ordered

$$\begin{aligned} &aaa < aab < aac < aba < abb < abc < aca < acb < acc < \\ &baa < bab < bac < bba < bbb < bbc < bca < bcb < bcc < \\ &caa < cab < cac < cba < cbb < cbc < cca < ccb < ccc \end{aligned}$$

These are all taken to be bigger than any monomial of degree 2.[8]

Every polynomial p has a unique term whose monomial part [9] is of highest order. This is called the leading term of p and is denoted $LT(p)$. A polynomial relation is converted to a simplification rule, by setting LHS to be $LT(p)$ and RHS to be $LT(p) - p$. One polynomial is **simpler** than another if the terms of the first polynomial are smaller in this ordering than the largest term of the second. Our simplification rules decrease the order of the terms.

C.3 Definition of Gröbner Basis

Definition 12: Let P be a polynomial ring and O an ordering on the terms of P. A set G of polynomials corresponds (using the ordering) to a set $\mathcal{L}$ of replacement rules. Let h and p be polynomials and let h be obtained from p by applying rules in list $\mathcal{L}$ until no further rules apply. We will say that h is a **normal form** of p and write $h = NForm(p, G)$ or $p \rightarrow_G h$.

Definition 13: Let P be a polynomial ring, I an ideal of P, and O an ordering on the terms of P. A set $G \subset I$ is

[7] We call attention to the way that p is obtained from p_2 and p_3 namely $p = p_2 c - b p_3$. This shows that p can be obtained from p_2 and p_3 by applying certain algebraic operations.

[8] This ordering is intended to capture our notion of simplicity. When we apply this machinery, we assign variables higher in the ordering (alphabet) to atomic expressions which seem more complicated. Terms having fewer factors are automatically regarded as simpler than terms with more factors.

[9] in the term $2xyx$, 2 is the coefficient and xyx the monomial part.

called a **Gröbner Basis** for I if it generates I and if $p \in I$ implies $p \rightarrow_G 0$.

We will also speak of a set of simplifying rules as a Gröbner Basis when the associated set of polynomials is a GB. If G is a Gröbner Basis, then $NForm(p, G)$ is independent of the order in which replacement rules are applied. It is also a canonical form for **algebraic equivalence** (that is, p and q are algebraically equivalent if $p-q$ is in the ideal I). [10]

D. A Gröbner Basis Algorithm

Here is a simplified version of Mora's Algorithm used to extend a set, G, of generators for an ideal I to a larger (and potentially complete) set of generators.

Simplified Basis Algorithm

Let $\mathcal{S} := \{(g_1, g_2) \mid g_1, g_2 \in G\}$
While $\mathcal{S} \neq \emptyset$
  Choose $(g_1, g_2) \in \mathcal{S}$
  $S := S - \{(g_1, g_2)\}$
  For all $f = SPoly(g_1, g_2)$
    Let $h = NForm(f, G)$
    If $h \neq 0$ then
      $\mathcal{S} := \mathcal{S} \cup \{(g, h) \mid g \in G\}$
      $G := G \cup \{h\}$

An $SPoly(g_1, g_2)$ is a combination of the form

$$a_1 l_1 g_1 r_1 - a_2 l_2 g_2 r_2$$

where the a_i are numbers and the l_i and r_i are monomials. These are chosen so that (1) the leading term of $a_1 l_1 g_1 r_1$ equals the leading term of $a_2 l_2 g_2 r_2$ and (2) this occurs in a "minimal" way [11]. $NForm(f, G)$ may depend on the sequence in which the simplification rules are applied. h can be chosen to be any normal form of f.

Notice that the algorithm is an iterative process which adds new polynomials to G. Notice also that every new element h which is added to G is in the ideal I. So the elements of G at any stage in this algorithm are all algebraic consequences of the starting G. Thus, if the original G consists of relations among a set of matrix expressions, all the elements of G (at any stage) will also be relations on the matrix expressions.

A criterion for a set G to be a Gröbner Basis is $SPoly(g_1, g_2) \rightarrow_G 0$ for all $g_1, g_2 \in G$. If the algorithm terminates, then the criterion shows that the (finite) resulting set G is automatically a Gröbner Basis for I. In the case of polynomials in commuting variables, the algorithm always terminates and so always produces a Gröbner Basis.

[10] If p and q are *algebraically* equivalent and if the generators of I are relations (i.e., become zero upon substitution), then p and q are also *strongly* equivalent. Thus, for most of our Gröbner Bases, which are obtained from somewhat evident starting relations, two expressions which simplify to the same normal form are equivalent in the usual sense.

[11] In the case of commuting variables, there is a unique minimal match – and so a unique $SPoly(g_1, g_2)$. In the non-commutative case, there may be none or several minimal matches for a given pair (g_1, g_2). See [**HW**], [**FMora**] or [**TMora**] for details.

D.1 Comments on Rules and Notation

In general, the process of applying rules to simplify a given expression is very quick once a Gröbner Basis has been computed. The process of computing a Gröbner Basis is usually labor intensive and there is a big advantage to computing it and storing it, once and for all, for a given set of atomic expressions. As we have noted, if the Mora Algorithm terminates yielding a finite basis, then this basis is automatically a Gröbner Basis [**FMora**].

In some of our examples, the Gröbner Basis is infinite. In this case, the Mora Algorithm is interrupted after it has produced sufficiently many new polynomials to indicate the ultimate result. The truncated output can be quite useful. In practice we have found that it can provide a list of rules which has considerable simplifying power. We have also found that, in analyzing the output, we could sometimes obtain recursive formulas for parametrized families of relations. Application of the SPoly criterion has allowed us to assert that the infinite families discussed in this paper are actually Gröbner bases. See Section II-C for examples of this.

Parametrized families of rules can be applied almost as readily as a finite set of rules. Thus, the use of an infinite set of rules can be quite practical. We are using an ordering which depends on the number of factors in a term. If we wish to simplify a particular expression, f, the only rules which will be applicable are those whose LHS has a smaller number of factors than the leading term of f. Thus a finite subset of the rules will be sufficient to simplify all expressions up to a certain complexity. An infinite set of rules has been implemented by storing all rules up to a sufficiently high degree to handle most situations, generating any instances of yet higher order rules as needed.

PART TWO: LISTS OF GRÖBNER BASES

In this part we will give lists of simplification rules which arise in settings of increasing complexity. We will provide examples of complete bases which are finite and also some which are infinite. We conclude with a formulation which provides a powerful general summary of many of our simplification rules.

II. Simplification Rules for some Common Settings

In this section we list Gröbner Bases for reducing polynomials in

(RESOL)	x, x^{-1} and $(1-x)^{-1}$
(EB)	x, y, x^{-1}, y^{-1}, $(1-xy)^{-1}$ and $(1-yx)^{-1}$
(preNF)	x, y, x^{-1}, y^{-1}, $(1-x)^{-1}$, $(1-y)^{-1}$, $(1-xy)^{-1}$ and $(1-yx)^{-1}$
(NF)	x, y, x^{-1}, y^{-1}, $(1-x)^{-1}$, $(1-y)^{-1}$, $(1-xy)^{-1}$, $(1-yx)^{-1}$, $(1-xy)^{1/2}$ and $(1-yx)^{1/2}$

The names (RESOL), (EB), (preNF) and (NF) are explained in the following sections even though the names are irrelevant to what we are doing here.

Here, as in the rest of this paper, we have adopted the convention that atomic expressions will be used as the

names of polynomial variables (sometimes called indeterminates). Thus, for example, x and x^{-1} are not matrices; they are variables for which matrices can be substituted. If we substitute a matrix for x, we must substitute the inverse of that matrix for x^{-1}.

There are many ways to impose orders on the monomials in the expressions we have listed above. The choice of an ordering for monomials is arbitrary, but the Gröbner Basis may depend on the particular order chosen. In this paper we use a graded lexicographic order which is determined by an ordering of the atomic expressions. We have selected orderings which reflect our subjective notion of which expressions are more complicated than others.

For example, the ordering $x < x^{-1} < (1-xy)^{-1}$ of variables is consistent with this intuitive idea of increasing complexity. Specifying an order on the three variables imposes a unique graded lexicographic order on the monomials in these variables. For example, when we use this graded lexicographic order, the following monomials are ordered as indicated:

$$\begin{aligned} xx &< xx^{-1} < x(1-xy)^{-1} < x^{-1}x < (1-xy)^{-1}x \\ &< (1-xy)^{-1}x(1-xy)^{-1} < (1-xy)^{-1}x^{-1}x \end{aligned}$$

A. A Gröbner Basis for Resol

The first list, called RESOL Rules, is a generalization of the example presented in Part One which involves expressions in x, x^{-1} and $(1-x)^{-1}$. The following list of rules[12] involves expressions in x, $(\lambda - x)^{-1}$ and $(\mu - x)^{-1}$.

$$\begin{aligned}
&(RESOL_0) && (\lambda-x)^{-1}x \to \lambda(\lambda-x)^{-1} - 1 \\
&(RESOL_1) && x(\lambda-x)^{-1} \to \lambda(\lambda-x)^{-1} - 1 \\
&(RESOL_2) && (\mu-x)^{-1}x \to \mu(\mu-x)^{-1} - 1 \\
&(RESOL_3) && x(\mu-x)^{-1} \to \mu(\mu-x)^{-1} - 1 \\
&(RESOL_4) && (\mu-x)^{-1}(\lambda-x)^{-1} \to \\
& && \quad \tfrac{1}{\mu-\lambda}(\lambda-x)^{-1} + \tfrac{1}{\lambda-\mu}(\mu-x)^{-1} \\
&(RESOL_5) && (\lambda-x)^{-1}(\mu-x)^{-1} \to \\
& && \quad \tfrac{1}{\mu-\lambda}(\lambda-x)^{-1} + \tfrac{1}{\lambda-\mu}(\mu-x)^{-1}
\end{aligned}$$

for all operators x on a Hilbert space $\mathcal{H}$ and distinct complex numbers λ and μ. The following theorem is an easy generalization of a corresponding result from [**HW**].

Theorem 14: The list RESOL Rules is complete (where λ and μ are distinct complex numbers).

Proof: If one uses the ordering

$$x \;<\; (\lambda-x)^{-1} \;<\; (\mu-x)^{-1}$$

and the polynomials corresponding to
$(RESOL_0)$, $(RESOL_1)$, $(RESOL_2)$ and $(RESOL_3)$ together with the fact that scalars λ and μ commute with everything
as starting relations for Mora's algorithm, then the algorithm terminates giving $(RESOL_0)$ through $(RESOL_5)$ as output. Thus by Section I-D, this is a GB.

[12]We use graded lexicographic order consistent with the order in which the symbols are listed.

The name RESOL reflects the fact that operator theorists call $(\lambda - x)^{-1}$ the resolvent of x.

B. A Gröbner basis for EB

The indeterminates which are used in EB and the ordering which we use is as follows:

$$x < y < x^{-1} < y^{-1} < (1-xy)^{-1} < (1-yx)^{-1} \; .$$

The set of relations of EB is the set of defining relations of
x^{-1}, y^{-1}, $(1-xy)^{-1}$ and $(1-yx)^{-1}$
(EB_0 through EB_7 below) [13]. This set of relations is not a Gröbner basis. The following theorem shows that one can extend this list of relations to obtain a Gröbner basis.

Theorem 15: The following relations constitute a finite Gröbner basis for EB.

$$\begin{aligned}
EB_0 &= x^{-1}x - 1 \\
EB_1 &= xx^{-1} - 1 \\
EB_2 &= y^{-1}y - 1 \\
EB_3 &= yy^{-1} - 1 \\
EB_4 &= xy(1-xy)^{-1} - (1-xy)^{-1} + 1 \\
EB_5 &= yx(1-yx)^{-1} - (1-yx)^{-1} + 1 \\
EB_6 &= (1-xy)^{-1}xy - (1-xy)^{-1} + 1 \\
EB_7 &= (1-yx)^{-1}yx - (1-yx)^{-1} + 1 \\
EB_8 &= (1-yx)^{-1}x^{-1} - y(1-xy)^{-1} - x^{-1} \\
EB_9 &= (1-xy)^{-1}y^{-1} - x(1-yx)^{-1} - y^{-1} \\
EB_{10} &= x^{-1}(1-xy)^{-1} - y(1-xy)^{-1} - x^{-1} \\
EB_{11} &= y^{-1}(1-yx)^{-1} - x(1-yx)^{-1} - y^{-1} \\
EB_{12} &= (1-yx)^{-1}y - y(1-xy)^{-1} \\
EB_{13} &= (1-xy)^{-1}x - x(1-yx)^{-1}
\end{aligned}$$

Proof: Mora's algorithm terminates producing this set.

We express this GB as a list of polynomials rather than as a list of replacement rules. We will use the convention that polynomials are written with terms in descending order. Thus the first term of a polynomial will be the LHS when it is converted to a replacement rule. Note that EB_6 and EB_7 can be reduced to 0 using the other rules and so EB_0 through EB_6 together with EB_9 through EB_{13} is a GB. They have been included in this list because they are in the starting set of relations and we find it helpful to keep the starting relations visible for reference.

The relations which form the GB for (EB) are of interest because they underlie energy balance equations in H^∞ control.

C. An infinite Gröbner basis for preNF

The set of relations considered in this section is named (preNF) because it is *pre*liminary to a set of relations which is named NF for *N*agy-*F*oias. [14] The indeterminates which are used in (preNF) and the ordering which we use are as follows. Using the guidelines for ordering atomic expressions mentioned at the beginning of Section II, the orders

[13]That is, they come from the definition of "inverse".

[14]The NF relations add $(1-xy)^{\frac{1}{2}}$ and $(1-yx)^{\frac{1}{2}}$ to preNF. They are important to those working with 2×2 block unitary matrices or with discrete time lossless balanced systems (called the Nagy-Foias operator model by mathematicians). Further details are found in [**HW**].

which we consider for the expressions of $(preNF)$ (expressions in x, y, x^{-1}, y^{-1}, $(1-x)^{-1}$, $(1-y)^{-1}$, $(1-xy)^{-1}$ and $(1-yx)^{-1}$) all have the form

$$\begin{matrix} x \\ y \end{matrix} < \begin{matrix} x^{-1} \\ y^{-1} \end{matrix} < \begin{matrix} (1-x)^{-1} \\ (1-y)^{-1} \end{matrix} < \begin{matrix} (1-xy)^{-1} \\ (1-yx)^{-1} \end{matrix} .$$

By specifying in addition that $x < y$ we have the order

$$\text{preNF} \quad \begin{matrix} x < y < x^{-1} < y^{-1} < (1-x)^{-1} < \\ (1-y)^{-1} < (1-xy)^{-1} < (1-yx)^{-1} . \end{matrix}$$

The set of relations of (preNF) is the set of defining relations of x^{-1}, y^{-1}, $(1-x)^{-1}$, $(1-y)^{-1}$, $(1-xy)^{-1}$ and $(1-yx)^{-1}$. Notice that the variables and relations for (preNF) are those for (EB) together with those for (RESOL) with $\lambda = 0$ and $\mu = 1$. The Gröbner basis obtained for (preNF) is infinite. It consists of a small collection of special relations followed by several sequences of parameterized relations. This is the content of the next theorem.

The theorem is proved using the S-Polynomial criterion discussed in Section I-D. The details of a similar proof are found in [HW]. Since the proof involves detailed checking of a large number of cases, we omit it here. The authors are studying ways to automate and simplify proofs of this sort.

Theorem 16: The following relations form a Gröbner Basis for $(preNF)$.
There are 22 special relations:

(A) The relations for (EB)
(B) The $(RESOL)$ relations for both x and y with $\lambda = 0, \mu = 1$
(C) Two additional relations

$$Prenf_1 = (1-y)^{-1}x(1-yx)^{-1} - (1-y)^{-1}(1-yx)^{-1} - x(1-yx)^{-1} + (1-y)^{-1}$$

$$Prenf_2 = (1-x)^{-1}y(1-xy)^{-1} - (1-x)^{-1}(1-xy)^{-1} - y(1-xy)^{-1} + (1-x)^{-1}$$

There are 8 (infinite) classes of general relations each of which are parameterized by a positive integer n:

$$I[n] = x(1-yx)^{-n}(1-x)^{-1} - (1-xy)^{-n}(1-x)^{-1} + (1-xy)^{-n}$$

$$II[n] = x(1-yx)^{-n}(1-y)^{-1} - (1-xy)^{-n}(1-y)^{-1} - x(1-yx)^{-n} + (1-xy)^{-(n-1)}(1-y)^{-1}$$

$$III[n] = y(1-xy)^{-n}(1-x)^{-1} - (1-yx)^{-n}(1-x)^{-1} - y(1-xy)^{-n} + (1-yx)^{-(n-1)}(1-x)^{-1}$$

$$IV[n] = y(1-xy)^{-n}(1-y)^{-1} - (1-yx)^{-n}(1-y)^{-1} + (1-yx)^{-n}$$

$$V[n] = (1-x)^{-1}(1-yx)^{-n}(1-x)^{-1} - (1-x)^{-1}(1-xy)^{-n}(1-x)^{-1} - (1-yx)^{-n}(1-x)^{-1} + (1-x)^{-1}(1-xy)^{-n}$$

$$VI[n] = (1-x)^{-1}(1-yx)^{-n}(1-y)^{-1} - (1-x)^{-1}(1-xy)^{-n}(1-y)^{-1} - (1-yx)^{-n}(1-y)^{-1} - (1-x)^{-1}(1-yx)^{-n} + (1-x)^{-1}(1-xy)^{-(n-1)}(1-y)^{-1} + (1-yx)^{-n}$$

$$VII[n] = (1-y)^{-1}(1-yx)^{-n}(1-x)^{-1} - (1-y)^{-1}\Sigma(1-x)^{-1} + \Sigma(1-x)^{-1} + (1-y)^{-1}\Sigma - \Sigma - (1-y)^{-1}(1-x)^{-1}$$

$$VIII[n] = (1-y)^{-1}(1-yx)^{-n}(1-y)^{-1} - (1-y)^{-1}(1-xy)^{-n}(1-y)^{-1} + (1-xy)^{-n}(1-y)^{-1} - (1-y)^{-1}(1-yx)^{-n}$$

where $\sum = \sum_{k=0}^{n}(1-xy)^{-k}$

Observations: Class IV is obtained from class I by interchanging x and y. Class II and III are similarly related. Class $VIII$ is obtained from class V by interchanging x and y and reordering terms. Some of the other classes (classes I and V) are obtained from very general rules ($(Gr2)$ respectively $(Gr4)$) in the forthcoming Section III.

III. General Rules

Some of the infinite families of rules which you have just seen are special cases of the simple rules which are given in this section. These rules are a bit sophisticated in that they are stated directly in terms of the functional calculus of a matrix. The functional calculus is an important construction in matrix and operator theory which associates the matrix $h(M)$ to a matrix M and a polynomial h in one complex variable. More generally, one can use a function h which is analytic on the spectrum of M. The mapping $h \to h(M)$ of analytic functions to matrices is what is called the **functional calculus** of M. For example, if $h(s) = \frac{1}{1-s}$, then $h(M)$ is $(1-M)^{-1}$. Similarly, one can obtain expressions like $h(xy) = (1-xy)^{1/2}$ and $h(M) = e^M$ provided the eigenvalues of xy and M are in the right location.

This section concentrates on a particular list of rules which are described in terms of the functional calculus. As you will see, a brief list of functional calculus based rules contains a great deal of information.

A. Statement of the (GENR) Rules

The following is a set of rules which hold for all operators x and y on a Hilbert space $\mathcal{H}$ with x, y, $\lambda - x$ and $\lambda - y$ invertible, functions h analytic on the spectrum of xy and yx and all $\lambda \neq 0$. (Technically, the following are not necessarily replacement rules for certain h since the left hand side may not be a monomial. We will discuss this shortly.)

GENR Rules

(Gr0)	$h(xy)x \to xh(yx)$
(Gr1)	$h(yx)x^{-1} \to x^{-1}h(xy)$
(Gr2)	$xh(yx)(\lambda - x)^{-1} \to \lambda h(xy)(\lambda - x)^{-1} - h(xy)$
(Gr3)	$x^{-1}h(xy)(\lambda - x)^{-1} \to \lambda^{-1}x^{-1}h(xy) + \lambda^{-1}h(yx)(\lambda - x)^{-1}$
(Gr4)	$(\lambda - x)^{-1}h(yx)(\lambda - x)^{-1} - (\lambda - x)^{-1}h(xy)(\lambda - x)^{-1} \to \lambda^{-1}(h(yx)(\lambda - x)^{-1} - (\lambda - x)^{-1}h(xy))$
(Gr5-9)	The rules (Gr0) through (Gr4) with x and y swapped.

In the above list of rules, the expression on the left hand side of the rule may not be a monomial. For example, see (Gr4) or set $h(z) = z + z^2$. It is easier to automate replacement rules if the LHS of a rule is a monomial. When these rules are used for machine computation, they are rearranged, using a term ordering, so that LHS is a monomial.

It should be noted that it is easy to verify the GENR Rules by hand so that they can be introduced independent

of the Gröbner Basis machinery. For example, (Gr1) follows from (Gr0) by multiplying (Gr0) on both sides by x^{-1} and the following calculation verifies (Gr3) using (RESOL) and (Gr1).

$$\begin{aligned} x^{-1}h(xy)(\lambda-x)^{-1} &= h(yx)x^{-1}(\lambda-x)^{-1} \\ &= (yx)(\lambda^{-1}x^{-1} + \lambda^{-1}(\lambda-x)^{-1}) \\ &= \lambda^{-1}h(yx)x^{-1} + \lambda^{-1}h(yx)(\lambda-x)^{-1} \end{aligned}$$

B. Properties of GENR

The rules $(RESOL)$ plus $(GENR)$ are a "complete" set of rules in a sense. A major point is that these rules are valid for *every* analytic function h. These rules are a "complete" [15] set of rules in that they are complete for the ideal generated by the key relations on

$$x, y, x^{-1}, y^{-1}, (\lambda-x)^{-1}, (\lambda-y)^{-1}, h(xy), h(yx)$$

for *any* h (which is analytic on the spectrum of xy and yx). These key relations are

$$\begin{array}{l} \text{the defining relations for the inverses } x^{-1}, \\ y^{-1}, (\lambda-x)^{-1} \text{ and } (\lambda-y)^{-1} \text{ and} \\ \text{the relations } h(xy)x - xh(yx) \text{ and } h(yx)y - yh(xy). \end{array} \tag{17}$$

This is discussed more thoroughly in [**HW**].

C. Using GENR with a particular function h

Now suppose we specialize to a particular h. If, for example, $h(z) = (1-z)^{-1}$, then we will have the same atomic expressions as we used in the preNF situation. A major difference, however, is that in the preNF case we added additional relations which come from the definition of $(1-xy)^{-1}$ and $(1-yx)^{-1}$. We do not expect, nor do we find, that GENR embodies all the extra relations that may hold for a particular h. What we do expect is that GENR will provide a useful set of easily implemented rules that at least provides simplification in a general sense – without using any special properties of a particular h.

Of the 8 infinite families listed for preNF, we find that I, IV, V, and VIII can be obtained from the GENR Rules together with the RESOL Rules while the other 4 families cannot. Here one makes the substitution $h(s) = (1-s)^{-n}$, so that $h(xy) = (1-xy)^{-n}$ and $h(yx) = (1-yx)^{-n}$. The remaining 4 families require the use of more particular properties of $(1-xy)^{-1}$ and $(1-yx)^{-1}$.

While this paper has not listed the NF rules, the NF Rules contain 16 infinite families. Eight of these infinite families follow from the GENR Rules together with the RESOL Rules.

Also, we mention that some special relations in $(preNF)$ and (NF) can be obtained from the GENR Rules together with the RESOL Rules.

[15] This is formalized by a computation of a Gröbner basis. in a related setting. A Gröbner basis can be found for polynomials in $x < y < x^{-1} < y^{-1} < (1-x)^{-1} < (1-y)^{-1} < S < T$ where S and T are variables and we take as starting relations the defining relations for the inverses and the relations $Sx = xT$ and $yS = Ty$ (see [**HW**]) which extract the algebraic essence of (17). For the case of $\lambda = 1$, the (GENR) rules are obtained by substituting $h(xy)$ for S and $h(yx)$ for T in the Gröbner basis from [**HW**].

Even in situation besides (preNF) and (NF) a natural thing to try is to supplement $(GENR)$ plus $(RESOL)$ with some of the obvious rules for whatever particular h you are using in your computations. We have found, in practice, that extremely effective simplification can be done this way.

PART THREE: USES OF GRÖBNER BASES

This part treats several different topics.

In Section IV, we will see that GB's are useful for the computational task of simplifying expressions. We will also provide an example of their use in making deductions.

The Gröbner Theory starts with a set of relations and produces new relations. The primary purpose of the Mora algorithm is to produce a "complete" set of relations. While it does not generate all possible relations, the new relations it does generate are often of intrinsic interest. We show how the famous formulas of Youla and Tissi for system similarity emerge directly from a GB as does half of the State Space Isomorphism Theorem. This is the subject of Section VI.

Section V concerns efficient computation of GB for Lyapunov equations.

IV. Simplification of formulas: an illustration involving H^∞ control.

In this section we will give an application of the simplification machinery discussed in Part One. We will also provide an example of the use of GB in making deductions.

A. An application of a Gröbner bases to H^∞ Control

A.1 Simplification

In H^∞ control, c.f. [**DGKF**], one deals with a Hamiltonian H on the state space (x, z) of the closed loop system. Here x is a state of the plant and z is a state of the compensator. The unknowns in H are a quadratic form ε which is to be a storage function of the closed loop system and the a, b, c, d which define the unknown compensator. As usual, take $d = 0$. Thus a, b, c and ε are unknowns. If a solution ε exists, then one can derive that for some controller, called the central controller, a, b and c must be given by certain formulas. These formulas do not imply that a solution to the H^∞ control problem exists. To see if it does, one must plug the central controller formulas back into H and see if $H \le 0$ for all states (x, z) of the closed loop system.

We apply the usual normalization to H:

$$\varepsilon(x,z) = \begin{pmatrix} x^T z^T \end{pmatrix} \begin{pmatrix} Y^{-1} & -(Y^{-1}-X) \\ -(Y^{-1}-X) & Y^{-1}-X \end{pmatrix} \begin{pmatrix} x \\ z \end{pmatrix} \tag{18}$$

The Doyle Glover Khargonekar Francis simplifying assumptions (c.f., [**DGKF**]) are then made. These greatly reduce the complexity of the formula for H. We still obtain an expression which is very complicated:

```
H =
  tp[x] ** X ** A ** z + tp[x] ** inv[Y] ** A ** x -
  tp[x] ** inv[Y] ** A ** z + tp[x] ** tp[A] ** X ** z +
  tp[x] ** tp[A] ** inv[Y] ** x - tp[x] ** tp[A] ** inv[Y] ** z +
  tp[x] ** tp[C1] ** C1 ** x + tp[z] ** X ** A ** x -
  tp[z] ** X ** A ** z - tp[z] ** inv[Y] ** A ** x +
  tp[z] ** inv[Y] ** A ** z + tp[z] ** tp[A] ** X ** x -
  tp[z] ** tp[A] ** X ** z - tp[z] ** tp[A] ** inv[Y] ** x +
```

```
tp[z] ** tp[A] ** inv[Y] ** z + tp[x] ** X ** B1 ** tp[B1] ** X ** z -
tp[x] ** X ** B2 ** tp[B2] ** X ** z +
tp[x] ** inv[Y] ** B1 ** tp[B1] ** inv[Y] ** x -
tp[x] ** inv[Y] ** B1 ** tp[B1] ** inv[Y] ** z +
tp[z] ** X ** B1 ** tp[B1] ** X ** x -
tp[z] ** X ** B1 ** tp[B1] ** X ** z -
tp[z] ** X ** B2 ** tp[B2] ** X ** x +
tp[z] ** X ** B2 ** tp[B2] ** X ** z -
tp[z] ** inv[Y] ** B1 ** tp[B1] ** inv[Y] ** x +
tp[z] ** inv[Y] ** B1 ** tp[B1] ** inv[Y] ** z -
tp[x] ** X ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** x +
tp[x] ** X ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** z +
tp[x] ** inv[Y] ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** x -
tp[x] ** inv[Y] ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** z -
tp[x] ** tp[C2] ** C2 ** Y ** inv[-1 + X ** Y] ** X ** x +
tp[x] ** tp[C2] ** C2 ** Y ** inv[-1 + X ** Y] ** X ** z +
tp[x] ** tp[C2] ** C2 ** Y ** inv[-1 + X ** Y] ** inv[Y] ** x -
tp[x] ** tp[C2] ** C2 ** Y ** inv[-1 + X ** Y] ** inv[Y] ** z +
tp[z] ** X ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** x -
tp[z] ** X ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** z -
tp[z] ** inv[Y] ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** x +
tp[z] ** inv[Y] ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** z +
tp[z] ** tp[C2] ** C2 ** Y ** inv[-1 + X ** Y] ** X ** x -
tp[z] ** tp[C2] ** C2 ** Y ** inv[-1 + X ** Y] ** X ** z -
tp[z] ** tp[C2] ** C2 ** Y ** inv[-1 + X ** Y] ** inv[Y] ** x +
tp[z] ** tp[C2] ** C2 ** Y ** inv[-1 + X ** Y] ** inv[Y] ** z +
tp[x] ** X ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** Y **
 inv[-1 + X ** Y] ** X ** x -
tp[x] ** X ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** Y **
 inv[-1 + X ** Y] ** X ** z -
tp[x] ** X ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** Y **
 inv[-1 + X ** Y] ** inv[Y] ** x +
tp[x] ** X ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** Y **
 inv[-1 + X ** Y] ** inv[Y] ** z -
tp[x] ** inv[Y] ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** Y **
 inv[-1 + X ** Y] ** X ** x +
tp[x] ** inv[Y] ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** Y **
 inv[-1 + X ** Y] ** X ** z +
tp[x] ** inv[Y] ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** Y **
 inv[-1 + X ** Y] ** inv[Y] ** x -
tp[x] ** inv[Y] ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** Y **
 inv[-1 + X ** Y] ** inv[Y] ** z -
tp[z] ** X ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** Y **
 inv[-1 + X ** Y] ** X ** x +
tp[z] ** X ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** Y **
 inv[-1 + X ** Y] ** X ** z +
tp[z] ** X ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** Y **
 inv[-1 + X ** Y] ** inv[Y] ** x -
tp[z] ** X ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** Y **
 inv[-1 + X ** Y] ** inv[Y] ** z +
tp[z] ** inv[Y] ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** Y **
 inv[-1 + X ** Y] ** X ** x -
tp[z] ** inv[Y] ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** Y **
 inv[-1 + X ** Y] ** X ** z -
tp[z] ** inv[Y] ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** Y **
 inv[-1 + X ** Y] ** inv[Y] ** x +
tp[z] ** inv[Y] ** inv[-1 + Y ** X] ** Y ** tp[C2] ** C2 ** Y **
 inv[-1 + X ** Y] ** inv[Y] ** z
```

Figure IV.1

Here we have used the same notation that one finds in our NCAlgebra program to give a feel for this type of computation. *tp* stands for transpose while *inv* stands for inverse and '**' for multiply. This expression has 57 terms. The leading term has 10 factors. Many of the factors in Figure IV.1 contain inverses of the type discussed in Part One. The rules (RESOL) together with the rules (EB) from Part One are stored in a function NCSimplifyRational (NCSR) in NCAlgebra which applies them repeatedly to an expression until no change occurs. When we apply NCSimplifyRational to H, we get the following considerably simpler expression. [16]

```
HYI := NCSimplifyRational[H]=
  tp[x] ** X ** A ** z + tp[x] ** inv[Y] ** A ** x -
   tp[x] ** inv[Y] ** A ** z + tp[x] ** tp[A] ** X ** z +
   tp[x] ** tp[A] ** inv[Y] ** x - tp[x] ** tp[A] ** inv[Y] ** z +
   tp[x] ** tp[C1] ** C1 ** x - tp[x] ** tp[C2] ** C2 ** x +
   tp[x] ** tp[C2] ** C2 ** z + tp[z] ** X ** A ** x -
   tp[z] ** X ** A ** z - tp[z] ** inv[Y] ** A ** x +
   tp[z] ** inv[Y] ** A ** z + tp[z] ** tp[A] ** X ** x -
   tp[z] ** tp[A] ** X ** z - tp[z] ** tp[A] ** inv[Y] ** x +
   tp[z] ** tp[A] ** inv[Y] ** z + tp[z] ** tp[C2] ** C2 ** x -
   tp[z] ** tp[C2] ** C2 ** z + tp[x] ** X ** B1 ** tp[B1] ** X ** z -
   tp[x] ** X ** B2 ** tp[B2] ** X ** z +
   tp[x] ** inv[Y] ** B1 ** tp[B1] ** inv[Y] ** x -
   tp[x] ** inv[Y] ** B1 ** tp[B1] ** inv[Y] ** z +
   tp[z] ** X ** B1 ** tp[B1] ** X ** x -
   tp[z] ** X ** B1 ** tp[B1] ** X ** z -
   tp[z] ** X ** B2 ** tp[B2] ** X ** x +
   tp[z] ** X ** B2 ** tp[B2] ** X ** z -
   tp[z] ** inv[Y] ** B1 ** tp[B1] ** inv[Y] ** x +
   tp[z] ** inv[Y] ** B1 ** tp[B1] ** inv[Y] ** z
```

Figure IV.2

[16]Reduction of H by just the starting rules does not produce a change.

This expression has 29 terms and the highest order term has only 6 factors. Notice that everything of the form inv[1-Y**X] and inv[1-X**Y] has been eliminated from H. This took 27 seconds on a SPARC II using NCAlgebra. [17]

We expect our simplifier to replace high order terms by lower order terms. The decrease in the number of factors in each term and the elimination of complicated factors is the expected behavior. The simplifier can also, as in this case, reduce the total number of terms. This is a consequence of the fact that terms are reduced to a standard form. This can produce cancellation of like terms. In our experience, the Gröbner technology has been very effective for simplifying expressions built from the type of subexpressions discussed in Part One.

Notice that the transition from Figure IV.1 to Figure IV.2 involves the use of general purpose simplification tools. It uses information about the way H is constructed as an algebraic expression, not on specialized information from H^∞ Control Theory.

A.2 Proving a Theorem

A major theorem in H^∞ Control Theory is that $H = 0$ if the Doyle-Glover-Khargonekar-Francis [**DGKF**] Riccati relations hold. We have simplified H to obtain the expression HYI which is still quite complicated. We now introduce the assumption that the [**DGKF**] Riccati equations $R_X = 0$ and $R_Y = 0$ hold where:

$$R_X = -XB_2B_2^TX + XB_1B_1^TX + C_1^TC_1 + A^TX + XA \quad \text{and}$$

$$R_Y = Y^{-1}B_1B_1^TY^{-1} - C_2^TC_2 + C_1^TC_1 + A^TY^{-1} + Y^{-1}A.$$

An ordering for the variables was chosen essentially at random by our computer program:

$A < B_1 < B_2 < C_1 < C_2 < x < X < Y^{-1} < z < A^T < B_1^T < B_2^T < C_1^T < C_2^T < x^T < z^T$.

This is done in NCAlgebra using the command

$$SetMonomialOrder[\{A, B1, B2, C1, C2, x, X, inv[Y], z, tp[A], tp[B1], tp[B2], tp[C1], tp[C2], tp[x], tp[z]\}].$$

The NCAlgebra command

$$GroebnerSimplify[HYI, \{R_X, R_Y\}]$$

computes the reduction of HYI with respect to the Gröbner Basis generated by R_X and R_Y. The result of the simplification of HYI using the above command was zero. The computation took 116 seconds using NCAlgebra.[18] The GB generated was finite.

We have just obtained [19] the classic result:

[17]The same computation took 1.8 secs using the special purpose system we have used for research (running on a 486/33MHz PC).

[18]Using the special-purpose research software it took 5.1 secs to calculate the Gröbner Basis and 0.4 secs to perform the reduction. NCAlgebra is integrated with Mathematica and is therefore slower.

[19]The idea of the proof is this: The fact that HYI reduces to zero using the GB obtained from the DGKF Riccati equations shows that HYI is in the ideal generated by R_X and R_Y. That is, HYI is a sum of terms of the form $c\mathcal{L}f\mathcal{R}$ where c is a number, $\mathcal{L}$ and $\mathcal{R}$ are monomials, and f is either R_X or R_Y. Thus, if matrices are substituted for the variables, a substitution which solves the DGKF equations will also make $HYI = 0$.

Theorem *The Hamiltonian of the closed loop system based on the central controller is identically zero if X and Y satisfy the two Doyle Glover Khargonekar Francis Riccati equations $R_X = 0$ and $R_Y = 0$.*

B. Comments

In the proof in Section IV-A.2 we produce a GB from relations special to the problem. The goal is to examine consequences of these relations. Gröbner Bases are applicable in other areas which involve matrix expressions. We emphasize that when working in H^∞ control, one often knows the DGKF Riccati equations hold. It is, therefore, natural to introduce these relations as hypotheses and seek to draw conclusions from them. We have used a well known theorem to illustrate the process. The ideas which we have presented can be just as useful when the answer is not known in advance. They can be a valuable tool for exploration. They can provide a quick way to check the correctness of a tentative set of assumptions. They can disclose additional conditions needed for a theorem to hold. They can provide a proof for a general theorem whose truth is suggested by examples or special cases.

V. The Effects of Ordering: An Illustration with Lyapunov Equations

In this section we shall examine GBs which arise in the study of Lyapunov equations. We shall see that, in some cases, the same starting relations will produce either a finite or an infinite GB, depending on the term ordering. Finally, in section V-B we make some comments on the description of infinite Gröbner Bases by generating functions.

A. Lyapunov Equations

If the Mora Algorithm terminates with a finite basis, this basis is automatically a Gröbner Basis. A finite basis is, therefore, advantageous. In contrast, if the Mora Algorithm does not terminate and is interrupted, a truncated list of rules produced may be useful for simplifying expressions. As we saw in Section II-C it is often possible to describe an infinite basis as collections of parameterized polynomials. However, this type of analysis is not automated and can be time consuming. In this section we examine a situation in which the same set of starting relations produce both finite and infinite bases depending on the choice of term ordering. We will give some guidelines for obtaining a finite GB in the case of the Lyapunov Equation and an application to Lossless systems. The finite Gröbner Bases which we find in this section are very small (e.g., around 30 relations) and can be generated with a computer in less than a minute.

The Lyapunov Equation is one of the most common equations in engineering:

$$am - mf - q = 0\,. \tag{19}$$

Here a, f and q are typically given and m is unknown, eventually to be determined numerically. At the algebraic stages of the research, one often is manipulating expressions in: a, f, q, m, m^{-1}, a^{-1}, f^{-1} and the resolvents of a and f. (We will not treat e^{at} and e^{ft} in this paper, although they do commonly arise.) We take (19), together with some invertibility assumptions as indicated, to be the starting relations for the GB process.

The following chart summarizes the results of some experiments with term ordering:

$am - mf =$	Invertible	Behavior	Examples
q	a and f	finite for all tested orders	$m < a < a^{-1} < f < f^{-1} < q$ $a < a^{-1} < f < f^{-1} < q < m$
q	a, f, m	infinite for all tested orders	$q < a < a^{-1} < f < f^{-1}$ $< m < m^{-1}$
$q_1 q_2$	a, f, m	finite for most tested orders	all orders in which $q_1 >$ both a and f
$q_1 q_2$	a, f, m	infinite for at least one order	$q_1 < q_2 < a < a^{-1} < f$ $< f^{-1} < m < m^{-1}$

Table V.1

Notice that the finitude of the GB depends on whether or not m is invertible, whether we use a 'q' (with total degree 1) or 'q_1q_2' (with total degree 2) as the affine term, and the choice of ordering. In many engineering applications the affine term 'q' in (19) is a quadratic of the form q_1q_2. For example, $Am - mA^T = C^TC$ is a familiar expression which has this property. The results above show that a Gröbner Basis obtained from this relation (together with the defining relations for the inverses) will be finite for suitably chosen orders.

It is common to manipulate expressions which contain the resolvents $(r-a)^{-1}$ and $(s-f)^{-1}$ where r and s are scalars. When we add these resolvents, similar conclusions are reached. In particular, it seems that if m and m^{-1} are high in the order, we obtain an infinite Gröbner Basis while if m and m^{-1} are low in the order, the basis is finite.

B. Infinite Families and Generating Functions

In §II-C, we gave a GB for ($preNF$) which was infinite. This GB consisted of a finite set of polynomials together with 8 infinite collections of polynomials. Each of these collections of polynomials was parameterizable using a single integer. There are other interesting ways to describe the members of an infinite family. Here we will explore the use of generating functions. A generating function can often be found which has the members of the family appearing as coefficients in its expansion.

Here is an example which is related to the Lyapunov Equations discussed in this section. We found that we obtain an infinite basis in some situations (Table V.1). For example, if one uses the starting relation $am - mf - q$ with the order $a < f < q < m < m^{-1}$, the Mora Algorithm produces general rules:

$$L_1(n) = m^{-1}a^n m - \sum_{k=0}^{n-1} m^{-1}a^k q f^{n-1-k} - f^n$$
$$L_2(n) = \sum_{k=0}^{n-1} m^{-1}a^k q f^{n-1-k} m^{-1} - m^{-1}a^n + f^n m^{-1}$$

for $n \geq 0$. If we set $GenFun(\lambda) = m^{-1}(1-\lambda a)^{-1}m - m^{-1}(1-\lambda a)^{-1}\lambda q(1-\lambda f)^{-1} - (1-\lambda f)^{-1}$, then

$$GenFun(\lambda) = \sum_{k=0}^{\infty} L_1(k)\lambda^k\,.$$

Generating functions like this occur in classical studies of Lyapunov equations. For example, if the spectrum of a

and f are disjoint, then the integral equation

$$\int_Q \frac{m}{\xi} GenFunc(\frac{1}{\xi}) d\xi = 0$$

yields the commonplace formula

$$m = \int_Q (\xi - a)^{-1} q (\xi - f)^{-1} d\xi \tag{20}$$

for m. Here the contour Q is chosen so that the spectrum of a lies inside of Q and the spectrum of f lies outside of Q.

VI. Gröbner Bases Spawn Interesting Formulas

Rather than viewing the Gröbner Basis Algorithm as a means towards the end of simplification, we view it in this section as a means for obtaining algebraic consequences of a set of equations. We will provide a simple illustration of how this occurs in a familiar system engineering context. Our example shows how the famous formulas of Youla and Tissi [**YT**] for system similarity emerge directly from a GB as does half of the State Space Isomorphism Theorem. One thing we shall see is that the occurrence of an infinite GB in this case is quite natural. For example, the elements of the infinite GB appear as coefficients of the power series expansions of frequency response function.

A. State Space Isomorphism Theorem

A basic theorem of system theory, the *State Space Isomorphism Theorem*, says that two controllable and observable systems with identical frequency response function T are "similar". Systems (a, b, c, d) and (A, B, C, d) are said to be similar if there is a map m satisfying

$$m^{-1} A m = a \tag{21}$$

$$c m^{-1} = C \tag{22}$$

$$m b = B\,. \tag{23}$$

It is natural to generate a GB for these relations in order to discover consequences. The defining relations for m^{-1} together with (21), (22) and (23) were used as starting relations for the Mora Algorithm:

$$\begin{array}{l} Rel[0] = m^{-1}m - 1 \\ Rel[1] = m m^{-1} - 1 \\ Rel[2] = m^{-1} A m - a \\ Rel[3] = c m^{-1} - C \\ Rel[4] = m b - B \end{array} \quad .$$

The ordering used was the graded lexicographical order induced by $A < B < C < a < b < c < m < m^{-1}$.

When we run Mora's Algorithm on $Rel[0]$-$Rel[4]$, the algorithm does not terminate in a short time and so we interrupted it and viewed the set which had been produced up to that point (see Section I-D). It was apparent that the Gröbner Basis was not finite. Computations by hand showed that the reduced Gröbner basis is the starting relations $Rel[0] - Rel[4]$ together with the special relations

$$\begin{array}{l} Rel[5] = ma - Am \\ Rel[6] = m^{-1}A - am^{-1} \\ Rel[7] = Cm - c \\ Rel[8] = m^{-1}B - b \end{array}$$

and general rules

$$\begin{array}{l} Gen[1] = CA^k m - ca^k \\ Gen[2] = ca^k b - CA^k B \\ Gen[3] = ca^k m^{-1} - CA^k \end{array}$$

for all $k \geq 0$.

We now recall that two system have the same frequency response function if

$$1 + c(s-a)^{-1} b = 1 + C(s-A)^{-1} B \tag{24}$$

for all $s \in \mathbf{C}$ such that both $s - a$ and $s - A$ are invertible. If (24) is expanded in powers of s, we find that the coefficients of the various powers (called Markov parameters) are precisely the relations $Gen[2]$. Thus we have shown that similar systems have the same Markov parameters; indeed this is the content of $Gen[2]$.

Also note that if the defining relations for $(s-a)^{-1}$ and $(s-A)^{-1}$ are added to the starting relations using the order

$$A < B < C < a < b < c < m < m^{-1} < (s-a)^{-1} < (s-A)^{-1}$$

then (24) itself is in the GB. Thus we see several ways in which the Gröbner process could be interpreted as producing formulas which prove one half the State Space Isomorphism Theorem.

A.1 The Youla-Tissi Formulas

Now we turn to the interpretation of $Gen[1]$ and $Gen[3]$.

Recall the famous formulas (due to Youla and Tissi [**YT**]) for the state space isomorphism m which say that it intertwines the controllability operators and observability operators of the system. In our notation these say

$$\text{(Y-T)} \qquad m a^k b - A^k B = 0 \;\text{ and }\; CA^k m - ca^k = 0$$

for all $k \geq 0$.

We see that $Gen[1]$ is exactly the second of the (Y-T) formulas while $Gen[3]$ is a simple variant of it. The first of the (Y-T) formulas reduces to zero (using Rel[5] and Rel[4]). So both the controllability and observability formulas have been shown to be a consequence of the relations which define similarity.

It is interesting to note that the second (Y-T) formula appears explicitly in the GB while the first does not (although they do reduce to zero). We note that there is a change in the ordering of variables [20] for which Mora's algorithm produces a GB in which the first (Y-T) formula occurs, but the second does not.

APPENDIX

[20] this is true for the ordering $m < m-1 < A < B < C < a < b < c$.

VII. Remarks on special situations

A. Exploiting finite dimensions

What we have done in this paper is purely algebraic and a formula derived by these methods takes no account of whether one is working with matrices on an n dimensional space, operators on a Hilbert space or on a Banach space, be they bounded or unbounded. Many of us wish to work with finite dimensional matrices and are fully willing to use the fact that they are finite dimensional. An interesting question is: can one formulate the finite dimensionality in an algebraic way which fits well with the techniques of this paper?

A common suggestion is that we use the Cayley-Hamilton Theorem if we work in an n dimensional space for a fixed n. The fact that every matrix M satisfies a polynomial equation of degree n would, at first glance, seem to provide a way for reducing higher powers of matrices to lower powers and eliminate infinite families of simplification rules. The main problem with this is that the characteristic polynomial, $p(M)$, of the matrix M has coefficients which depend on M. Thus even if we knew the characteristic polynomial for each of the atomic variables $x_1, \ldots, x_k$ in the problem, we would not necessarily know the characteristic polynomial for any sum, product, etc. of the atomic variables. Consequently, using the Cayley-Hamilton Theorem to impose finite dimensionality does not give rules which apply in general. Indeed, when a specialist uses the Cayley-Hamilton Theorem in derivations, he typically applies it to one or two matrices which he has carefully constructed. In this context, one might adjoin a characteristic polynomial equation to a computer algebra session to obtain results for a particular matrix.

B. Square vs nonsquare matrices

Another question which arises is how do these techniques handle nonsquare matrices. At first glance it appears that there is a problem because the setting for this paper is an algebra, to wit we may multiply any two matrices, while if, for example, B is not a square matrix, then BB is not meaningful. In other words, when the matrices involved are all square matrices of the same size, the translation between polynomials and matrix expressions is clear and simple. Any product of variables makes sense. On the other hand, when the matrices involved are not all square matrices, then some products and some sums of matrices are allowed while others are not.

We begin by considering a collection of polynomials which we call **allowable**. This is done in a purely algebraic way by attaching to each variable, x, a pair of numbers $r(x)$ and $c(x)$ (which will correspond to the number of rows and columns for the matrix which will be substituted for x in the problem). We allow only products of elements with compatible dimensions. We can attach dimensions to any allowable product[21]. A polynomial is allowable if each term is allowable and all the terms have the same dimensions. Intuitively, these are polynomials which produce meaningful matrix expressions when we substitute matrices of the proper dimensions for the variables.

Note that, in all examples of this paper, the starting relations correspond to allowable matrix expressions and all the relations in the GB's we obtained correspond to allowable matrix expressions. The following theorem shows that this phenomenon holds in general.

Theorem 25: If the starting relations are allowable, then Mora's algorithm produces only relations which are allowable.

The proof requires an analysis of the details of the Mora Algorithm at a level beyond the scope of this paper and is omitted.

[21] The empty product 1 is considered to be allowable, but it is not assigned dimensions. It acts as if $r(1) = c(1) = n$ for arbitrary n.

REFERENCES

[BL] Buchberger, B. & Loos, R. "Algebraic Simplification" Computer Algebra - Symbolic and Algebraic Computation, Springer-Verlag(1982),pp 11-43.

[CLS] D. Cox, J. Little, D. O'Shea, *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*, Springer-Verlag, Undergraduate Texts in Mathematics, 1992.

[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.

[HW] J. W. Helton and J. J. Wavrik "Rules for Computer Simplification of the formulas in operator model theory and linear systems", Operator Theory: Advances and Applications 73 (1994), pp. 325—354.

[FMora] F. Mora, "Groebner Bases for Non-commutative Polynomial Rings" Lecture Notes in Computer Science, number 229 (1986) pp 353-362.

[TMora] T. Mora, "An introduction to commutative and non-commutative 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

[NF] B. Sz-Nagy and C. Foias, Harmonic Analysis of Operators on Hilbert Space North Holland 1970

[YT] D.C. Youla and P. Tissi "n-port synthesis via lossless extraction- part I" 1966 Intern. Conv. Record vol 14 pt 7, pp 183-208