IMPLEMENTATION OF SOME LINEAR MULTIVARIABLE SYSTEMS ALGORITHMS VIA MAPLE V

Y. Vlioras*, S.N. Antoniou*, A.I.G. Vardulakis*,
**Aristotle University of Thessaloniki*
Dept. of MAthematics - Faculty of Sciences
54 006 - Thessaloniki - Greece
Fax : +30 31 997 951
e-mail : vlioras@ccf.auth.gr - antoniou@ccf.auth.gr - avardula@ccf.auth.gr

Keywords : Education, Linear Systems, Symbolic Computation.

Abstract

The algorithms required for the solution of several problems in the study of linear multivariable systems, can be proved to be a very tedious task and usually only a few trivial examples can be entirely worked out manually in a reasonable time. A package containing some basic algorithms for the study of linear systems, implemented via Maple V, is presented here as a very useful tool mainly for educational purposes.

1 Introduction

The study of linear multivariable systems is one of the most important issues in control theory. For the study of state space systems or even generalized state space systems, there exist several algorithms which can be used to solve analysis or design problems. This is because a state space description is nothing more than a quadruple of real constant matrices A, B, C, D and thus numerical algorithms, having as input these matrices, can be used. For the study of polynomial matrix models we cannot use directly numerical algorithms, but there are two alternatives. The first alternative is to use polynomial matrix theory to implement directly algorithms using some symbolic computation package like Maple V, Macsyma or Mathematica. The second one, is to reduce the polynomial matrix description to a (generalized) state space form using some known realization method and then to apply numerical algorithms (using for example Matlab) for the study of the equivalent systems. However, even in this second case we need a reliable symbolic algorithm to convert the original system to its reduced equivalent.

Numerical algorithms are very efficient and fast comparing to the symbolic ones and thus they are much more attractive for real time applications or industrial use. On the other hand symbolic algorithms are more accurate and able to manipulate directly polynomial matrices, which is the case here. This second feature is the one that makes them very attractive for educational purposes.

In this paper we present a package implemented in Maple V, which contains some basic algorithms mainly for the analysis of polynomial models such as extraction of greatest common divisors, determination of matrix fraction descriptions of rational matrices, state space realizations, etc. Similar works using Maple V can be found in [6], [7], [5]. The aim of the paper is to show the power of symbolic packages for the manipulation of polynomial matrices, which are the basic tools for the study of linear systems in the frequency domain.

2 The package

This package was implemented in Maple V - release 3 and contains some procedures mainly for the analysis of polynomial models. Here we shall present only the basic ones, while the rest of them are simply auxiliary functions used internally by the algorithms. The complete listing of the procedures can be found in the appendix at the end of the paper. All the examples were worked out on a Pentium 75MHz machine with 16MB of main memory. The package should be loaded using the command

```
>read 'linsys.m';
```

In all the algorithms s is considered to be the indeterminate of the polynomials and thus the symbol s should remain unassigned. In order to unassign the symbol s we can use the following

```
>s:='s';
```

$$s := s$$

or equivalently

```
>unassign('s');
```

It is also important to notice here that all these procedures require the Maple's package for Linear Algebra. This can be loaded as follows

>with (linalg);

The first two algorithms compute the right or left greatest common divisor of two polynomial matrices.

gcrd (A,B) - gcld (A,B): compute the right (resp. left) greatest common divisor of the polynomial matrices A,B. The method used here (see for example [1], [2]) performs row (resp. column) elementary operations to reduce the block matrix

$$\begin{bmatrix} A \\ B \end{bmatrix} \text{ (resp. } [A, B]\text{)}$$

to the form

$$\begin{bmatrix} Gr \\ 0 \end{bmatrix} \text{ (resp. } [Gl, 0]\text{)}$$

where Gr (resp. Gl) are square triangular matrices. Then Gr and Gl are respectively the right and left greatest common divisor of A, B. The number of columns or rows respectively of the matrices A,B must equal otherwise we get the error message "incompatible matrix dimensions".

Example 1 *Consider the following two matrices*

>A:=matrix([[s^2+1,s+2],[s-1,0]]);

$$A := \begin{bmatrix} s^2+1 & s+2 \\ s-1 & 0 \end{bmatrix}$$

>B:=matrix([[s+1],[s-1]]);

$$B := \begin{bmatrix} s+1 \\ s-1 \end{bmatrix}$$

>Gl:=gcld(A,B);

$$Gl := \begin{bmatrix} 5 & 0 \\ s-1 & \frac{6}{5}s - \frac{6}{5} \end{bmatrix}$$

> ■

The next two procedures compute a coprime matrix fraction description, i.e. a numerator and a denominator matrix, of a rational transfer function.

mfrd(R) - mfld(R): result a pair of polynomial matrices which are the numerator and the denominator (respectively right or left) of the rational matrix R, i.e.

$$R = N_R D_R^{-1} \text{ or } R = D_L^{-1} N_L$$

and furthermore N_R, D_R (resp. N_L, D_L) are right (resp. left) coprime. These two algorithms use the gcrd and gcld routines to derive a coprime numerator and denominator (for more details on the method used here see [4]).

Example 2 *Consider the following*

>R:=matrix([[(s^2-s)/(s+2),s+1/s], [0,s/(s+1)+3]]);

$$R := \begin{bmatrix} \frac{s^2-s}{s+2} & s+\frac{1}{s} \\ 0 & \frac{s}{s+1}+3 \end{bmatrix}$$

>T:=mfrd(R);

$$T := \begin{bmatrix} s^2-s & s^3+s^2+s+1 \\ 0 & 4s^2+3s \end{bmatrix}, \begin{bmatrix} s+2 & 0 \\ 0 & s^2+s \end{bmatrix}$$

>Nr:=T[1];

$$Nr := \begin{bmatrix} s^2-s & s^3+s^2+s+1 \\ 0 & 4s^2+3s \end{bmatrix}$$

>Dr:=T[2];

$$Dr := \begin{bmatrix} s+2 & 0 \\ 0 & s^2+s \end{bmatrix}$$

>evalm(Nr&*inverse(Dr)); ← {verify that $R = NrDr^{-1}$}

$$\begin{bmatrix} \frac{s(s-1)}{s+2} & \frac{s^2+1}{s} \\ 0 & \frac{4s+3}{s+1} \end{bmatrix}$$

> ■

The following procedures results a column or row reduced form of a polynomial matrix

colreduce(P) - rowreduce(P): return as result two matrices. The first matrix is the column (resp. row) reduced form of the polynomial matrix P, while the second one is a unimodular matrix which post- (resp. pre-) multiplies P to obtain its column (resp. row) reduced form. The method used here (see [3] or [1]) applies elementary column (resp. row) operations to reduce the column (resp. row) complexity of P until it reaches its lower bound which is the dertminental degree of the matrix.

Example 3 *Consider the following*

>P:=matrix([[s^2+1,s^10+s+1],[0,s-1]]);⟵ {obviously the column complexity is $2 + 10 = 12 > \deg|P| = 3$}

$$P := \begin{bmatrix} s^2+1 & s^{10}+s+1 \\ 0 & s-1 \end{bmatrix}$$

>T:=colreduce(P);

$$T := \begin{bmatrix} -s & s^2+s+1 \\ -s+1 & s-1 \end{bmatrix}, \begin{bmatrix} -s^2+s^4-s^6+s^8+1 & s^2-s^4+s^6-s^8 \\ -1 & 1 \end{bmatrix}$$

>PR:=T[1];⟵ { Now it is obvious that the complexity is $1 + 2 = 3$}

$$PR := \begin{bmatrix} -s & s^2+s+1 \\ -s+1 & s-1 \end{bmatrix}$$

>U:=T[2];

$$U := \begin{bmatrix} -s^2+s^4-s^6+s^8+1 & s^2-s^4+s^6-s^8 \\ -1 & 1 \end{bmatrix}$$

>det(U);⟵ {U is unimodular}

$$1$$

>evalm(P&*U);⟵ { verify that $PU = P_r$}

$$\begin{bmatrix} -s & s^2+s+1 \\ -s+1 & s-1 \end{bmatrix}$$

> ■

The following algorithms perform several types of realization of a given rational matrix transfer function.

spreal (P) : This function returns three matrices, namely A, B, C, as a minimal state space realization of the strictly proper matrix transfer function P, i.e. A, B, C satisfy

$$P = C(sI - A)^{-1}B$$

and the pairs (A, B) and (A, C) are respectively controllable and observable. The method is similar to these proposed in [3] and [1] and it is based on the derivation of coprime matrix fraction description of P, whose denominator is in a column reduced form. This procedure internally uses all the above functions to obtain the realization.

Example 4 *Consider the following*

>P:=matrix([[0,1/(s+1)],[-1/(s+1),1/s]]);

$$P := \begin{bmatrix} 0 & \frac{1}{s+1} \\ -\frac{1}{s+1} & \frac{1}{s} \end{bmatrix}$$

>T:=spreal(P);

$$T := \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ -1 & 1 & 1 \end{bmatrix}$$

>A:=T[1]:
>B:=T[2]:
>C:=T[2]:
>evalm (C&*inverse(s*Id(3)-A)&*B); ← {verify that $P = C(sI - A)^{-1}B$}

$$\begin{bmatrix} 0 & \frac{1}{s+1} \\ -\frac{1}{s+1} & \frac{1}{s} \end{bmatrix}$$

> ■

polreal (P): Returns a 'pure' generalized state space realization of the polynomial matrix transfer function P. The procedure results three matrices namely E, B, C, such that

$$P = C(sE - I)^{-1}B$$

Furthermore E is nilpotent and the block $[E, B]$, $[E^T, C^T]^T$ have full rank, which means that the realization is controllable and observable at $s = \infty$. The method used for this type of realization can be found in [1] and is based on the state space realization of the strictly proper transfer function

$$\tilde{P}(s) = \frac{1}{s}P(s)$$

Example 5 *Consider the polynomial matrix*

>P:=matrix([[s^2+1,s-2],[s+1,s-1]]);

$$P := \begin{bmatrix} s^2+1 & s-2 \\ s+1 & s-1 \end{bmatrix}$$

>T:=polreal (P);

$$T \quad : \quad = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 1 & 1 & -2 \\ 0 & 1 & 1 & 1 & -1 \end{bmatrix}$$

>E:=T[1]:
>B:=T[2]:
>C:=T[3]:
>evalm(C&*inverse(s*E-Id(5))&*B);⟵{verify that $P = C(sE - I)^{-1}B$}

$$\begin{bmatrix} s^2+1 & s-2 \\ s+1 & s-1 \end{bmatrix}$$

> ■

The last procedure performs an irreducible generalized state space realization of a general rational transfer function.

realize(P) :returns a quadruple of matrices E, A, B, C such that

$$P = C(sE - A)^{-1}B$$

where P is the given rational matrix transfer function. The procedure decomposes P into two parts as $P = P_{pol} + P_{sp}$ where P_{pol} is the polynomial part of P and P_{sp} is the strictly proper part of P. The result follows simply (see [1]) from

$$P = [C, C_\infty] \begin{bmatrix} sI - A & 0 \\ 0 & sA_\infty - I \end{bmatrix} \begin{bmatrix} B \\ B_\infty \end{bmatrix}$$

Example 6 *Consider the rational transfer function*

>P:=matrix([[(1-s)/(s+2),s^2+1/s],[0,s+3]]);

$$P := \begin{bmatrix} \frac{1-s}{s+2} & s^2+\frac{1}{s} \\ 0 & s+3 \end{bmatrix}$$

>T:=realize(P);

$$T := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix}, \begin{bmatrix} 3 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 3 \end{bmatrix}$$

>Ep:=T[1]:
>A:=T[2]:
>B:=T[3]:
>C:=T[4]:
>evalm(C&*inverse(s*Ep-A)&*B);⟵ {verify that $C(sE-A)^{-1}B = P$}

$$\begin{bmatrix} -\frac{s-1}{s+2} & \frac{s^3+1}{s} \\ 0 & s+3 \end{bmatrix}$$

> ■

3 Conclusions

In the past symbolic computations was a very difficult task, which was carried out exclusively by mainframes or supercomputers. Today's progress of computer technology allows the usage of symbolic packages even in a cheap personal computer. Several software packages, like Maple, Mathematica and Macsysma, provide a user-friendly environment and a high-level programming language, for the development of specialized routines. These features make them very attractive for the implementation of algorithms which can be used either for analysis, design or educational purposes, in the field of control theory.

The aim of this note is to present some first steps towards this direction. We have constructed a small but very useful Maple package which contains algorithms for basic polynomial matrix calculations such as GCDs, MFDs etc. Using these basic procedures and the existing theory we implemented a generalized state space realization algorithm. The algorithms used in all these procedures can be very complicated and only trivial examples can be entirely worked out manually. Thus the help of such a tool, in the study of linear multivariable systems can be proved to be invaluable.

4 Appendix

```
findmin:=proc(A:array,j:nonnegint)
local f,i,p,mn,wmn;
p := rowdim(A);
wmn := j;
mn := infinity;
for i from j to p do
f := degree(A[i,j],s); if f < mn and A[i,j] <> 0 then mn
:= f; wmn := i fi
od;
wmn
end
elim:=proc(A:array,j:nonnegint)
local t,i,r,p,wmn;
p := rowdim(A);
t := A;
while not iszero(col(delrows(t,1 .. j),j)) do
wmn := findmin(t,j);
t := swaprow(t,j,wmn);
for i from j+1 to p do t := addrow(t,j,i,-quo(t[i,j],t[j,j],s))
od;
t := map(expand,t)
od;
evalm(t)
end
gcrd:=proc(A:array,B:array)
local T,p,m,j,G;
if coldim(A) <> coldim(B) then ERROR(`Incompatible
matrix dimensions`) fi;
T := blockmatrix(2,1,A,B);
p := rowdim(T);
m := coldim(T);
G := T;
for j to m do G := elim(G,j) od;
G := delrows(G,m+1 .. p)
end
gcld:=proc(A:array,B:array)
transpose(gcrd(transpose(A),transpose(B))) end
mfrd:=proc(R:array)
local Rn,De,Den,Num,G,i,j;
Rn := map(simplify,R);
De := map(denom,Rn);
Den := diag(seq(lcm(seq(col(De,j)[i],
i = 1 .. rowdim(De))),j = 1 .. coldim(De)));
Num := map(expand,evalm(Rn &* Den));
G := gcrd(Num,Den);
if 0 < degree(det(G),s) then
Num := map(expand,evalm(Num &* inverse(G)));
Den := map(expand,evalm(Den &* inverse(G)))
fi;
evalm(Num),evalm(Den)
end
mfld:=proc(R:array)
local t;
t := mfrd(transpose(R));
evalm(transpose(t[1])),evalm(transpose(t[2]))
end
coldeg:=proc(A:matrix,c:nonnegint)
local n,d,co,i,de;
co := col(A,c);
```

```
n := rowdim(A);
d := -infinity;
for i to n do
de := degree(expand(co[i]),s); if d < de then d := de fi
od;
d
end
highcol:=proc(A:matrix)
local p,m,d,i,Ah,j,de;
m := coldim(A);
p := rowdim(A);
Ah := matrix(p,m);
d := matrix(m,m,0);
for i to m do
de := coldeg(A,i);
d[i,i] := s^de;
for j to p do Ah[j,i] := coeff(expand(A[j,i]),s,de) od
od;
evalm(Ah),evalm(d)
end
colreduce:=proc(A:matrix)
local tA,Ah,U,p,m,fr,N,md,wmd,d,i,No,t;
tA := A;
p := rowdim(A);
m := coldim(A);
U := array(identity,1 .. m,1 .. m);
fr := min(p,m);
t := highcol(tA);
Ah := evalm(t[1] &* t[2]);
while rank(Ah) < fr do
N := eval(kernel(Ah)[1]);
md := infinity;
for i to m do
d := degree(N[i],s);
if d < md and N[i] <> 0 then md := d; wmd := i fi
od;
No := N[wmd];
N := scalarmul(N,1/No);
for i to m do
if i <> wmd then
tA := evala(addcol(tA,i,wmd,N[i]));
U := evala(addcol(U,i,wmd,N[i]))
fi
od;
t := highcol(tA);
Ah := evalm(t[1] &* t[2])
od;
evalm(tA),evalm(U)
end
rowreduce:=proc(A:matrix)
local tA;
tA := colreduce(transpose(A));
transpose(tA[1]),transpose(tA[2])
end
makeS:=proc(A:matrix,Nu:matrix,De:matrix)
local v,Ss,Ac,Bh,Ch,tA,k,j,i,t,p,m,n,l,tNu;
tA := map(expand,A);
tNu := map(expand,Nu);
p := rowdim(Nu);
m := coldim(A);
n := 0;
for i to m do v[i] := coldeg(De,i)-1; n := n+v[i]+1 od;
Ac := matrix(m,n,0);
Ss := matrix(n,m,0);
Bh := matrix(n,m,0);
Ch := matrix(p,n);
l := 1;
for j to m do
Bh[l+v[j],j] := 1;
for k from 0 to v[j] do
Ss[l+k,j] := s^k;
for i to m do Ac[i,l+k] := coeff(tA[i,j],s,k) od;
for i to p do Ch[i,l+k] := coeff(tNu[i,j],s,k) od
od;
l := l+v[j]+1
od;
evalm(Ac),evalm(Ss),evalm(Bh),evalm(Ch)
end
spreal:=proc(R:matrix)
local
temp,p,m,i,Ao,Bm,Ss,Lp,Am,Bh,Dbc,Aht,Bht,Cht,De,Nu;
if iszero(R) then ERROR(`Cannot realize identicaly zero
matrix`)
fi;
temp := mfrd(R);
De := map(expand,temp[2]);
Nu := map(expand,temp[1]);
temp := colreduce(De);
Nu := map(expand,evalm(Nu &* temp[2]));
De := map(expand,temp[1]);
p := rowdim(Nu);
m := coldim(De);
temp := highcol(De);
Bm := inverse(temp[1]);
Lp := map(expand,evalm(De-(temp[1] &* temp[2])));
Ao := JordanBlock(0,coldeg(De,1));
for i from 2 to m do
Ao := diag(Ao,JordanBlock(0,coldeg(De,i))) od;
temp := makeS(Lp,Nu,De);
Dbc := temp[1];
Am := evalm(-(Bm &* Dbc));
Bh := temp[3];
Ss := temp[2];
Cht := temp[4];
Aht := evalm(Ao+(Bh &* Am));
Bht := evalm(Bh &* Bm);
evalm(Aht),evalm(Bht),evalm(Cht)
end
polreal:=proc(P:matrix)
local R;
if iszero(P) then ERROR(`Cannot realize identicaly zero
matrix`) fi;
R := scalarmul(map(x -> subs(s = 1/s,x),P),1/s);
R := map(simplify,R);
```

```
R := spreal(R);
evalm(R[1]),evalm(-R[2]),evalm(R[3])
end
Id:=j->array(identity,1..j,1..j);
separate:=proc(A:matrix)
local p,m,Hpol,Hsp,tA,d,n,i,j,Af,Ai;
tA := convert(A,rational);
p := rowdim(A);
m := coldim(A);
Hpol := matrix(p,m);
Hsp := matrix(p,m);
for i to p do
for j to m do
d := denom(tA[i,j]);
n := numer(tA[i,j]);
Hpol[i,j] := quo(n,d,s);
Hsp[i,j] := rem(n,d,s)/d
od
od;
evalm(Hpol),evalm(Hsp)
end
realize:=proc(R:matrix)
local T,Ai,Af,Ep,A,B,C,n,mi;
T := separate(R);
n := 0;
mi := 0;
if not iszero(T[1]) then Ai := polreal(T[1]);
mi := rowdim(Ai[1]) fi;
if not iszero(T[2]) then Af := spreal(T[2]);
n := rowdim(Af[1]) fi;
if 0 < n and 0 < mi then
Ep := diag(Id(n),Ai[1]);
A := diag(Af[1],Id(mi));
C := concat(Af[3],Ai[3]);
B := stack(Af[2],Ai[2])
elif n = 0 and 0 < mi then Ep := Ai[1];
A := Id(mi); C := Ai[3]; B := Ai[2]
elif mi = 0 and 0 < n then Ep := Id(n);
A := Af[1]; C := Af[3]; B := Af[2]
else ERROR(`Cannot realize identicaly zero matrix`)
fi;
evalm(Ep),evalm(A),evalm(B),evalm(C)
end
```

References

[1] Vardulakis A.I.G., "Linear Multivariable Control : Algebraic Analysis & Synthesis Methods", J. Willey & Sons, UK (1991).

[2] Kucera V., "Discrete Linear Control, The Polynomial Equation Approach", J. Willey & Sons, Chichester (1979).

[3] Wolovich W.A., "The determination of state space representations for linear multivariable systems", Automatica, 9, pp. 97-106, (1973).

[4] Kailath T., "Linear Systems", Prentice Hall, Enclewood Cliffs, N.J. (1980).

[5] Tzekis P., Karampetakis N., Vardulakis A.I.G., "Solution of diophantine equations via Maple", Proc. 4th IEEE Mediterranean Symposium on Control, Chania, Crete, (1996).

[6] Ogunye A.B., Penlidis A., " State Space computations using Maple V", IEEE Control Systems Mag., vol. 16, No 1, pp. 70-77, (1996).

[7] Ogunye A.B., A. Penlidis A., "Computation of Systems Gramians and balanced realizations using Maple V", Int. Journal of Systems Science, vol. 26, No 4, pp.899-926, (1995).