CODEN: LUTFD2/(TFRT-7432)/1-19/(1989)

Hermite Forms in Algebraic System Theory

Michael Lundh

Department of Automatic Control
Lund Institute of Technology
October 1989

Department of Automatic Control **Lund Institute of Technology** P.O. Box 118 S-221 00 Lund Sweden	*Document name* Internal Report *Date of issue* October 1989 *Document Number* CODEN: LUTFD2/(TFRT–7432)/1-19/(1989)
Author(s) Michael Lundh	*Supervisor* Per Hagander *Sponsoring organisation*

Title and subtitle

Hermite Forms in Algebraic System Theory

Abstract

The use in algebraic system theory of the Hermite form of a stable rational transfer function matrix is demonstrated. A MACSYMA function that calculates the Hermite form of a matrix is given, and two examples show how it is used.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title		*ISBN*
Language English	*Number of pages* 19	*Recipient's notes*
Security classification		

1. Introduction

Design and synthesis of multi-input-multi-output systems using algebraic methods are presented in Pernebo (1978) and Vidyasagar (1985). The algebra is hard to work with. This report describes some MACSYMA functions that performs the calculations.

Pernebo uses Smith form to factorize matrices of stable rational transfer functions. This is a very powerful tool, in many cases unnecessarily complicated. In some cases it is sufficient to use Hermite forms instead. The Hermite form is much simpler to calculate than the Smith form. Here Hermite forms are used to determine coprime factorizations, to determine structure matrices and to solve Diophantine-Aryabhatta-Bezout identities for feedback design.

2. The Algebra

This section defines the algebra. A more thorough treatment is found in Pernebo (1978) and in Vidyasagar (1985).

Definition 1
Let $\mathbb{R}_\Lambda[\mathrm{s}]$ be the set of rational transfer functions $G(s) = b(s)/a(s)$ where $b(s)$ and $a(s)$ are polynomials with real coefficients. Further $a(s)$ has all roots outside the set $\Lambda \in \mathbb{C}$. □

For continuous systems the complement to the set Λ is a subset of the complex numbers with negative real part. If Λ is extended with the infinity point the set $\mathbb{R}_\Lambda[\mathrm{s}]$ is restricted to proper rational functions.

Definition 2
Here the set Λ is defined as

$$\Lambda = \{z \in \mathbb{C} : \mathrm{Re}\, z \geq 0\} \bigcup \{\infty\}$$

Now $\mathbb{R}_\Lambda[\mathrm{s}]$ consists of stable proper rational functions. □

Vidyasagar shows that $\mathbb{R}_\Lambda[\mathrm{s}]$ is an Euclidean domain with a degree function $\delta(G)$ that is the number of zeros in Λ of $G(s)$. For the definition of Λ above, δ equals the number of finite zeros of $b(s)$ in Λ plus the relative degree of $G(s)$. A unit is an element $U \in \mathbb{R}_\Lambda[\mathrm{s}]$ with $\delta(U) = 0$. Moreover U^{-1} is also a unit in $\mathbb{R}_\Lambda[\mathrm{s}]$.

Division

It is possible to use Euclid's algorithm for division since $\mathbb{R}_\Lambda[\mathrm{s}]$ is an Euclidean domain. Let $A(s)$ and $B(s)$ belong to $\mathbb{R}_\Lambda[\mathrm{s}]$. Division is defined through the identity

$$A(s) = B(s)Q(s) + R(s) \tag{1}$$

where $\delta(R) < \delta(B)$ and $Q(s)$ and $R(s)$ belong to $\mathbb{R}_\Lambda[\mathrm{s}]$. This division is however not unique and several Q and R may fulfill the degree conditions. Feasible quotient Q and remaider R are calculated as follows. If $\delta(B) = 0$ then B is a unit. It yields $Q = A/B$ and $R = 0$. If $\delta(A) < \delta(B)$ no division is possible. Therefore $Q = 0$ and $R = A$. In all other cases first define the polynomials a_{num}, a_{den}, b_{num} and b_{den} as the numerators and denominators of A and B respectively. Factorize $b_{num} = b^+b^-$ so that b^+ is monic and has all zeros outside Λ and all zeros of b^- belong to Λ. Define a monic polynomial $a_o(s)$ of degree $\delta(B) - 1$ with all zeros outside Λ, e.g. $(s+\alpha)^{\delta(B)-1}$, $\alpha > 0$. Solve the Diophantine-Aryabhatta-Bezout identity

$$a_{den}(s)x(s) + b^-(s)y(s) = a_{num}(s)a_o(s) \tag{2}$$

with $\deg y < \deg a_{den}$. The quotient and remainder in the division (1) are now given by

$$\begin{aligned} Q(s) &= \frac{x(s)b_{den}(s)}{a_{den}(s)a_o(s)b^+(s)} \\ R(s) &= \frac{y(s)}{a_o(s)} \end{aligned} \tag{3}$$

It must be remembered that this division algorithm is not guaranteed to yield a reminder $R(s)$ of minimal degree $\delta(R)$.

Matrices

DEFINITION 3
Let $\mathrm{IR}_\Lambda^{\mathrm{p}\times\mathrm{m}}[\mathrm{s}]$ define the set of matrices with entries in $\mathrm{IR}_\Lambda[\mathrm{s}]$. □

A matrix $M(s) \in \mathrm{IR}_\Lambda^{\mathrm{p}\times\mathrm{p}}[\mathrm{s}]$ is Λ-unimodular if $\det(M)$ is a unit. Elementary row operations may be obtained by multiplication from the left by a Λ-unimodular matrix. Using elementary row operations a Λ-Hermite form may be obtained.

LEMMA 1
Any matrix in $\mathrm{IR}_\Lambda^{\mathrm{p}\times\mathrm{m}}[\mathrm{s}]$ of rank r can be reduced by elementary row operations to a quasi triangular form in which

1. If $p > r$, the last $p - r$ rows are zero;
2. In column j, $1 \le j \le r$, the diagonal element is a product of primes and is of higher degree than any nonzero element above it;
3. If $m > r$, no particular statements can be made about the elements in the last $m - r$ columns and first r rows.

Proof: By row interchange bring the element of lowest degree in the first column to the 1,1-position. Call this m_{11}. Now use the division algorithm above to write the other elements in this column as a multiple m_{11} plus a reminder of lower degree than m_{11}. By elementary row operations subtract the appropriate multiple of m_{11} from every other element so that only remainders of lower degree than m_{11} are left. Repeat the whole procedure until all but the first element in the first column is zero.

Repeat this procedure on the second column while ignoring the first row until all elements below the diagonal element is zero. Then use an elementary row operation determined by the division algorithm to make the element above the diagonal in the second column of lower degree than the diagonal element. Continue the whole procedure for the remaining columns. □

Remark 1. The lemma describes the column Λ-Hermite form $R = UM$ of the matrix M. The row Λ-Hermite form $L = MV$ is achieved by transposing, since $L = (V^TM^T)^T$. □

Remark 2. The prime factors in $\mathbb{R}_\Lambda[s]$ are rational functions of three types

$$\frac{1}{s+c} \quad \text{or} \quad \frac{s-a}{s+c} \quad \text{or} \quad \frac{(s-a)^2+b^2}{(s+c)^2} \tag{4}$$

with $a \geq 0$, $b, c > 0$. They correspond to a zero at infinity, a real zero in Λ and a pair of complex conjugate zeros in Λ. □

Remark 3. The Hermite form is not unique since the elements above the diagonal are not unique, due to the nonunique remainder of the division in $\mathbb{R}_\Lambda[s]$. □

Remark 4. Compare with the Hermite form for polynomial matrices in Kailath (1980), theorem 6.3-2. □

3. Coprime Factorization

Let a multivariable system be described by

$$A(s)y(s) = B(s)u(s) \tag{5}$$

where $A(s) \in \mathbb{R}_\Lambda^{p\times p}[s]$ and $B(s) \in \mathbb{R}_\Lambda^{p\times m}[s]$. In some cases it is necessary to have a minimal realization of the system (5). For such a realization A and B are relatively left Λ-prime. They will then not have any common left divisors with zeros in Λ.

Assume that A and B have a common left Λ-divisor. A greatest common left Λ-divisor L is found for instance using the Λ-Hermite form.

$$[\,A(s) \quad B(s)\,]\,U(s) = [\,L(s) \quad 0\,] \quad \text{or} \quad \begin{bmatrix} L^T(s) \\ 0 \end{bmatrix} = U^T(s) \begin{bmatrix} A^T(s) \\ B^T(s) \end{bmatrix} \tag{6}$$

This gives

$$\begin{aligned} A(s) &= L(s)A_0(s) \\ B(s) &= L(s)B_0(s) \end{aligned} \tag{7}$$

Assuming $L(s)$ being invertible, a minimal realization of (5) is

$$A_0(s)y(s) = B_0(s)u(s) \tag{8}$$

If A and B are relatively left Λ-prime then $L = I$.

4. Structure Matrices

Pernebo defines structure matrices that in some sense represent the part of the system that is difficult to handle. The structure matrices contain the zeros in Λ of the system. These zeros will limit the servo and the regulator performance of the closed loop system. They are calculated from certain matrices in the system description, see Pernebo.

Any matrix $M(s) \in \mathbb{R}_\Lambda^{p\times m}[s]$ with rank r can be factorized as $M = \hat{M}\tilde{M}$ where $\hat{M} \in \mathbb{R}_\Lambda^{p\times r}[s]$ has rank r and $\tilde{M} \in \mathbb{R}_\Lambda^{r\times m}[s]$ is right Λ-invertible. The matrix $\hat{M}(s)$ is a left structure matrix of $M(s)$. The row Λ-Hermite factorization provides a way to calculate a left structure matrix.

$$M(s)V(s) = [\,L(s) \quad 0\,] \qquad \Longleftrightarrow \qquad M(s) = [\,L(s) \quad 0\,]V^{-1}(s) \tag{9}$$

Now $V^{-1}(s)$ is Λ-unimodular, i.e. its rows are linearly independent. The matrix $\tilde{M}(s)$ is the first r rows in $V^{-1}(s)$. These are linearly independent. The last $p-r$ rows have no influence since they correspond to columns with zeros in the matrix $[\,L(s) \quad 0\,]$. The matrix $L(s)$ is lower triangular with r rows. It follows that

$$M(s) = [\,L(s) \quad 0\,]V^{-1}(s) = [\,\hat{M}(s) \quad 0\,]\begin{bmatrix}\tilde{M}(s)\\ *\end{bmatrix} = \hat{M}(s)\tilde{M}(s) \tag{10}$$

The Λ-Hermite form of the matrix $M(s)$ is in fact a structure matrix $\hat{M}(s)$.

Right structure matrices are found from the column Λ-Hermite form.

5. Feedback Design

In this section it is shown how the Λ-Hermite form can be used for feedback design. Let the open system be described by a minimal stable factorization as in Pernebo.

$$\begin{bmatrix}A_1(s) & A_3(s)\\ 0 & A_2(s)\end{bmatrix}\begin{bmatrix}z(s)\\ y(s)\end{bmatrix} = \begin{bmatrix}B_1(s)\\ B_2(s)\end{bmatrix}u(s) + \begin{bmatrix}C_1(s)\\ C_2(s)\end{bmatrix}e(s) \tag{11}$$

It describes the relation between the output to be controlled z, the measured output y, the control input u and the disturbance input e. To stabilize the system it is required that $A_2(s)$ and $B_2(s)$ are relatively left Λ-prime and that $A_1(s)$ is Λ-invertible. Further the transfer function $G_{yu}(s) = A_2^{-1}(s)B_2(s)$ must be strictly proper to guarantee the existence of a proper controller. Assume therefore that $A_2(\infty) = I$ and $B_2(\infty) = 0$ to make $G_{yu}(s)$ strictly proper.

Determine a feedback controller

$$\begin{aligned}R(s)\xi(s) &= -y(s)\\ u(s) &= S(s)\xi(s)\end{aligned} \tag{12}$$

by solving the Diophantine-Aryabhatta-Bezout identity

$$A_2(s)R(s) + B_2(s)S(s) = I \tag{13}$$

using the Λ-Hermite factorization

$$[\,A_2 \quad B_2\,]\begin{bmatrix}U_{11} & U_{12}\\ U_{21} & U_{22}\end{bmatrix} = [\,L \quad 0\,] \tag{14}$$

A controller yielding a Λ-stable closed loop system is achieved if and only if $L = I$, i.e. if A_2 and B_2 have no common zeros in λ.

$$\begin{aligned}A_2U_{11} + B_2U_{21} &= I\\ A_2U_{12} + B_2U_{22} &= 0\end{aligned} \tag{15}$$

The second equation in (15) provides a right Λ-prime factorization of $A_2^{-1}B_2$

$$A_2^{-1}B_2 = -U_{12}U_{22}^{-1} = ND^{-1} \tag{16}$$

Multiply the second equation in (15) from the right with Q that is an arbitrary matrix of appropriate dimension with entries in $\mathbb{R}_\Lambda[s]$. Add this to the first equation in (15).

$$A_2(U_{11} + U_{12}Q) + B_2(U_{21} + U_{22}Q) = I \tag{17}$$

Define all stabilizing controllers by

$$\begin{aligned} R = U_{11} + U_{12}Q &\quad = R_0 + NQ \\ S = U_{21} + U_{22}Q &\quad = S_0 - DQ \end{aligned} \tag{18}$$

The requirements on G_{yu} now implies that

$$\begin{aligned} &A_2(\infty)(U_{11}(\infty) + U_{12}(\infty)Q(\infty)) \\ &\qquad + B_2(\infty)(U_{21}(\infty) + U_{22}(\infty)Q(\infty)) &= \\ &A_2(\infty)(U_{11}(\infty) + U_{12}(\infty)Q(\infty)) &= I \end{aligned} \tag{19}$$

Then $R(s) = U_{11}(s) + U_{12}(s)Q(s)$ is invertible for $s = \infty$. This means that the controller

$$S(s)R^{-1}(s) \tag{20}$$

is proper.

The all stable input-output relations between e and z and e and u are given by

$$\begin{aligned} A_1 z &= (C_1 - (A_3R + B_1S)C_2)\, e \\ u &= -SC_2 e \end{aligned} \tag{21}$$

The right structure matrix $\hat{C}$ of the system is given by the factorization $C_2 = \tilde{C}\hat{C}$. It is calculated from the column Λ-Hermite form. In (21) it is seen that the part $\hat{C}$ always will be present since only $\tilde{C}$ can be canceled by a matrix with entries in $\mathbb{R}_\Lambda[s]$.

The properties of the closed loop system are affected by the choice of denominator polynomials in (11) and of the choice of Q.

6. MACSYMA Programs

The calculations in this algebra are carried out using MACSYMA, see Macsyma (1983). The procedure for calculation of Λ-Hermite form is based on Holmberg (1986), where a MACSYMA function that calculates the Hermite form for polynomial matrices is given. The function here for matrices in $\mathbb{R}_\Lambda^{p\times m}[s]$ has been provided with another degree function and another division algorithm as defined in section 2. The main routines are

`INLAMBDA(c)`	A function that defines the set $\Lambda \in \mathbb{C}$. It returns true if the complex number c belongs to Λ.

`LAMBDAHERMITE(m)` A function that calculates the Λ-Hermite form of the matrix `m`. It returns a unimodular matrix `u` and the matrix `r` (= `u·m`) that is on Λ-Hermite form.

Other important functions are `DAB` that solves the Diophantine-Aryabhatta-Bezout identity $AX+BY=C$ for polynomials, `RATDEG` that returns the degree $\delta(\cdot)$ of an element in $\mathbb{R}_\Lambda[s]$ and `RATDIV` which performs Euclidean division in $\mathbb{R}_\Lambda[s]$. All functions are listed in appendix 1.

Since computations are performed numerically, it is necessary to define a limit stating that if the difference between two numbers are less that this limit the numbers are considered as equal. This limit is set using the variable `ZERO_LIMIT`.

7. Examples

Two examples will demonstrate how the Λ-Hermite form is calculated and what it is used for.

EXAMPLE 1—Calculation of Λ-Hermite Form
Consider Rosenbrock's system

$$G(s)=\begin{bmatrix}\frac{1}{s+1} & \frac{2}{s+3}\\ \frac{1}{s+1} & \frac{1}{s+1}\end{bmatrix} \tag{22}$$

This matrix has two zeros in Λ. They are revealed in the Λ-Hermite form

$$R(s)=\begin{bmatrix}\frac{1}{s+1} & \frac{2}{s+3}\\ 0 & \frac{1-s}{(s+1)^2}\end{bmatrix} \tag{23}$$

Appendix 2 shows a MACSYMA session where $R(s)$ is determined. All intermediate steps are are given. An identity matrix is initially placed beside $G(s)$ to record the different elementary row operations that together determine the unimodular matrix $U(s)$.

$$U(s)[G(s)\quad I]=[U(s)G(s)\quad U(s)]=[R(s)\quad U(s)] \tag{24}$$

The output from `LAMBDAHERMITE` is expression `(d39)` which is a list with $R(s)$ and $U(s)$. □

EXAMPLE 2
The other example deals with the feedback design of the system

$$\begin{bmatrix}z(s)\\ y_1(s)\\ y_2(s)\end{bmatrix}=\begin{bmatrix}0 & \frac{s-2}{s-1} & \frac{2}{s+1}\\ \frac{1}{s+1} & \frac{1}{s-1} & \frac{s-1}{s+1}\\ \frac{1}{s-1} & \frac{2}{s+2} & \frac{1}{s+2}\end{bmatrix}\begin{bmatrix}u_1(s)\\ u_2(s)\\ e(s)\end{bmatrix} \tag{25}$$

or shorter $x(s)=G(s)w(s)$. The numbers `(cij)` here refer to the equation numbers in appendix 3 where the MACSYMA session is found.

`c13-c14` Make a fractional representation. $A(s)x(s)=B(s)w(s)$.

`c16-c21` Make the fractional representation left Λ prime $A_0(s)x(s)=B_0(s)w(s)$.

c23-c36	Make the fractional representation on the standard form (11). Extract the matrices $A_2(s)$, $B_2(s)$ and $C_2(s)$.
c37-c40	Calculate the right structure matrix of the system.
c45-c52	Calculation of a stabilizing controller $S_0(s)R_0(s)^{-1}$, and a parametrization for all stabilizing controllers.

8. Conclusions

Some MACSYMA functions have been used for analysis and design of MIMO systems. A Hermite form for matrices with entries in $\mathbb{R}_\Lambda[s]$ is calculated. Some examples show how it may be used.

9. References

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

KAILATH, T. (1980): *Linear Systems*, Prentice-Hall, Inc., Englewood Cliffs, N.J.

MACSYMA (1983): *Reference Manual*, Symbolics, Inc., Cambridge, Mass.

PERNEBO, L. (1978): "Algebraic Control Theory for Linear Multivariable Systems," Doctorial Dissertation CODEN: LUTFD2/TFRT-1016, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

VIDYASAGAR, M. (1985): *Control System Synthesis: A factorization Approach*, MIT Press., Cambridge.

Appendix 1 – MACSYMA Programs

```
/*------------------------------------------------------------------
  LAMBDASTARTUP.MAC
  Startup file for using functions for algebriac approach.

  Michael Lundh
  LastEditDate : Thu Oct  5 09:17:34 1989
  ----------------------------------------------------------------*/

batchload("lambdahermite.mac");

floatformat(f,0,3);

/* ===== switches ===== */
keepfloat  : true$
print_true : false$
zero_limit : 0.0001$

/* ===== Definition of the set Lambda ===== */
lambdalim  : 0.0$

/* Function that returns true if the complex c belongs to Lambda */
/* [1]= */
inlambda(c):= (realpart(c)>=(lambdalim-zero_limit)) or abs(c)>1.0e30$

/*------------------------------------------------------------------
  LAMBDAHERMITE.MAC
  Functions for calculation of the lambda-hermite form.
  Skeletton taken from Holmberg (1986) TFRT-7333.

  Michael Lundh
  LastEditDate : Wed Oct 18 14:52:00 1989
  ----------------------------------------------------------------*/

/* Column-Lambda-Hermite form of x and Lambda-unimodular matrix
   performing the row operations. */
/* [Hermite,UL]= */
lambdahermite(x,[alpha]):=
   Block([p,n,m,i,np,UL],
         if length(alpha)=0 then alpha:-1 else alpha:part(alpha,1),

         n:length(transpose(x)),
         m:length(x),
         np:rank(x),
         x:addcol(x,ident(m)),

         p:1,
         for i:1 thru np do
           (p:findcol(x,i,p),
            x:reducebelow(x,i,p,alpha),
            if i#1 then x:reduceabove(x,i,p,alpha)),

         UL:x,
         for i:1 thru n do UL:submatrix(UL,1),
         for i:n+m step -1 thru n+1 do x:submatrix(x,i),
         return:[x,UL])$


/* Returns column where reduction should be applied */
findcol(x,i,p):=
   Block([n],
         n:length(transpose(x)),
         if p>n then n,
         if not zerobelow(x,i,p) then p else
         (if x[i,p]#0 then p else findcol(x,i,p+1)))$
```

```
/* Reduces x from x[i,p] and below */
reducebelow(x,i,p,[alpha]):=
   Block([down],
         if length(alpha)=0 then alpha:-1 else alpha:part(alpha,1),

         x:rowperm(x,i,p),
         print("rowperm below row",i),    if print_true then print(x),
         x:monic(x,i,p,alpha),
         print("monic leadcoef row",i),   if print_true then print(x),
         x:Euclid(x,i,p,down,alpha),
         print("Euclid div below row",i), if print_true then print(x),
         if not zerobelow(x,i,p) then reducebelow(x,i,p,alpha) else x)$


/* Reduces x above x[i,p] */
reduceabove(x,i,p,[alpha]):=
   Block([powi,pow,k,up],
         if length(alpha)=0 then alpha:-1 else alpha:part(alpha,1),

         x:Euclid(x,i,p,up,alpha),
         print("Euclid div above row",i), if print_true then print(x),
         powi:ratdeg(part(x,i,p)),
         for k:i-1 step -1 thru 1 do
           (pow:ratdeg(part(x,k,p)),
            if not pow<powi then x:reduceabove(x,i,p,alpha)),
         x)$


/* Returns true if column below x[i,p] is zero */
/* What is zero?  Possible modification of last statement. */
zerobelow(x,i,p):=
   Block([k,j],
         k:col(x,p),
         for j:1 thru i do k:if length(k)=1 then 0 else submatrix(1,k),
         if k.k=0 then true else false)$


/* Makes the element x[i,p] prime using row-operation */
monic(x,i,p,[alpha]):=
   Block([xip,bp,uii,k,zii],
         if length(alpha)=0 then alpha:-1 else alpha:part(alpha,1),

         xip:ratsimp(part(x,i,p)),
         bp:part(lambdafact(num(xip)),1),
         uii:1/bp,
         if inlambda(inf) then
           (k:hipow(expand(denom(xip)),s)-hipow(expand(bp),s),
            uii:uii*denom(xip)/(s-alpha)^k),
         zii:simpfact(uii*xip),
         if (denom(zii)=1) and (hipow(num(zii),s)=0) and
            (abs(zii)>zero_limit) then uii:uii/zii,
         x:setelmx(uii,i,i,ident(length(x))).x,
         return:simplify(x,i,row))$


/* Permutes rows giving lowest degree of column p first. */
rowperm(x,i,p):=
   Block([m,k,ei,e,powi,pow,L],
         m:length(x),
         ei:part(x,i,p),
         powi:if ei#0 then ratdeg(ei) else 10^6,
         for k:i+1 thru m do
           (e:part(x,k,p),
            pow:if e#0 then ratdeg(e) else 10^6,
            if pow<powi then
              (L:setelmx(0,i,i,ident(m)),
               L:setelmx(1,i,k,L),
               L:setelmx(0,k,k,L),
               L:setelmx(1,k,i,L),
               x:L.x)),
         x)$
```

```
/* Polynomial division for of elements in column p */
Euclid(x,i,p,upordown,[alpha]):=
   Block([m,k,qr,L,begin,end],
         if length(alpha)=0 then alpha:-1 else alpha:part(alpha,1),

         m:length(x),
         if upordown=down then (begin:i+1,end:m)
                          else (begin:1,end:i-1),
         for k:begin thru end do
           (qr:ratdiv(part(x,k,p),part(x,i,p),alpha),
            L:setelmx(-part(qr,1),k,i,ident(m)),
            x:simplify(L.x,k,row)),
         x)$


/*-----------------------------------------------------------------
  Auxiliary routines
  -----------------------------------------------------------------*/


/* Returns true if the polynomial p=0 */
/* [l]= */
zeropoly(p):=
   Block([l,k],
         p:expand(p),
         l:true,
         for k:0 thru hipow(p,s) do
           if abs(coeff(p,s,k))>zero_limit then
              (l:false,return(false)),
         return:l)$


/* Solution to diophantine equation aa*xx+bb*yy=cc with deg(yy)<deg(aa) */
/* [xx,yy]= */
dab(aa,bb,cc):=
   Block([dega,degb,x,y,u,v,g,h,i,hold,uold,vold,tmp,a,b,c],
         dega:hipow(expand(aa),s),
         degb:hipow(expand(bb),s),
         x:1, y:0, u:0, v:1,
         if degb>dega then (a:bb, b:aa, tmp:x, x:y, y:tmp, tmp:u, u:v, v:tmp)
         else              (a:aa, b:bb),
         g:a,   h:b,
         for i:1 while not zeropoly(h) do
           (hold:h, uold:u, vold:v,
            tmp:divide(g,h,s),
            g:part(tmp,1), h:part(tmp,2),
            u:x-g*u,
            v:y-g*v,
            g:hold, x:uold, y:vold),
         if degb>dega then (tmp:x, x:y, y:tmp, tmp:u, u:v, v:tmp),
         tmp:divide(1.0*cc,g,s),
         g:part(tmp,1), c:part(tmp,2),
         if not zeropoly(c) then
           error("Common factor of A and B is not in C"),
         b:g*y,
         tmp:divide(b,v,s),
         a:part(tmp,1),
         return:[ratsimp(g*x-a*u) , ratsimp(part(tmp,2))])$


/* Factorization of B=(B+)(B-). (B+) monic with zeros outside Lambda.
   (B-) with all zeros in Lambda */
/* [bp,bm]= */
lambdafact(b):=
   Block([be,degb,rb,i,rbi,bp,bm],
         be:expand(b),
         degb:hipow(be,s),
         bp:1,
         bm:coeff(be,s,degb),
          rb:allroots(b),
          for i:1 thru length(rb) do
            (rbi:part(rb,i,2),
             if inlambda(rbi) then bm:bm*(s-rbi)
                              else bp:bp*(s-rbi)),
          return:[realpart(expand(bp)),realpart(expand(bm))])$


/* Lambda-degree of the generalized polynomial gp */
/* [deg]= */
ratdeg(gp):=
   Block([rgp,bpbm,deg,k],
          rgp:ratsimp(gp),
          bpbm:lambdafact(num(rgp)),
          deg:hipow(expand(part(bpbm,2)),s),
          k:hipow(expand(denom(rgp)),s)-hipow(expand(num(rgp)),s),
          if (k>0) and inlambda(inf) then
            deg:deg+k,
          if num(rgp)=0 then
            deg:-1,
          return:deg)$


/* Quotient and Remainder from division of Lambda-generalized polynomials */
/* [q,r]= */
ratdiv(a,b,[alpha]):=
   Block([ar,br,anum,aden,bnum,bden,tmp,bp,bm,k,ao,x,y,q,r],
          if length(alpha)=0 then alpha:-1 else alpha:part(alpha,1),

          ar:ratsimp(a), anum:num(ar), aden:denom(ar),
          br:ratsimp(b), bnum:num(br), bden:denom(br),

          tmp:lambdafact(bnum),
          bp:part(tmp,1),
          bm:part(tmp,2),
          k:ratdeg(br),

          if k=0 then
            (q:ar/br, r:0)
          else if ratdeg(ar)<k then
            (q:0, r:ar)
          else
            (ao:(s-alpha)^(k-1),
             tmp:dab(aden,bm,anum*ao),
             y:part(tmp,1),
             x:part(tmp,2),
             q:x*bden/(aden*ao*bp),
             r:y/ao),
         return:[cutpoly(q), cutpoly(r)])$


/* Eliminates coefficients with magnitude less than zero_limit in
   Lambda-generalized polynomials. */
/* [gp]= */
cutpoly(gp):=
   Block([b,a,bo,ao,i,ci],
          gp:ratsimp(gp),
          b:expand(num(gp)),
          a:expand(denom(gp)),
          bo:0,
          for i:0 thru hipow(b,s) do
            (ci:coeff(b,s,i), if abs(ci)>zero_limit then bo:bo+ci*s^i),
          ao:0,
          for i:0 thru hipow(a,s) do
            (ci:coeff(a,s,i), if abs(ci)>zero_limit then ao:ao+ci*s^i),
          return:(bo/ao))$
```

```
/* Eliminates real common factors */
/* [gp]= */
simpfact(gp):=
   Block([b,a,i,rb,tmp],
         gp:cutpoly(gp),
         b:num(gp),
         a:denom(gp),

         rb:allroots(b),
         for i:1 thru length(rb) do
           (tmp:divide(a,s-part(rb,i,2),s),
            if zeropoly(part(tmp,2)) then
              (a:part(tmp,1),
               b:part(divide(b,s-part(rb,i,2),s),1))),
          return:realpart(expand(b))/realpart(expand(a)))$


/* Matrix version */
simplify(gpm,[p2]):=
   Block([im,ml,mu, in,nl,nu],
         ml:1,   mu:length(gpm),
         nl:1,   nu:length(transpose(gpm)),
         if length(p2)=1 then
           (ml:part(p2,1), nl:part(p2,1)),
         if length(p2)=2 then
           if part(p2,2)=row then
             (ml:part(p2,1), mu:part(p2,1))
           else
             (nl:part(p2,1), nu:part(p2,1)),

         for im:ml thru mu do
           for in:nl thru nu do
             gpm[im,in]:simpfact(gpm[im,in]),
         return:gpm)$
```

Appendix 2 – Example 1

```
(c37) g : matrix([1/(s+1),2/(s+3)],[1/(s+1),1/(s+1)]);

                                   [   1       2   ]
                                   [ -----   ----- ]
                                   [ s + 1   s + 3 ]
(d37)                              [               ]
                                   [   1       1   ]
                                   [ -----   ----- ]
                                   [ s + 1   s + 1 ]

(c39) ans : lambdahermite(g);

                          [   1       2          ]
                          [ -----   -----   1  0 ]
                          [ s + 1   s + 3        ]
rowperm below row 1       [                      ]
                          [   1       1          ]
                          [ -----   -----   0  1 ]
                          [ s + 1   s + 1        ]

                          [   1       2          ]
                          [ -----   -----   1  0 ]
                          [ s + 1   s + 3        ]
monic leadcoef row 1      [                      ]
                          [   1       1          ]
                          [ -----   -----   0  1 ]
                          [ s + 1   s + 1        ]

                          [   1          2                 ]
                          [ -----      -----       1    0  ]
                          [ s + 1      s + 3               ]
Euclid div below row 1    [                                ]
                          [            1 - s               ]
                          [   0    ------------   - 1   1  ]
                          [          2                     ]
                          [         s  + 4 s + 3           ]

                          [   1          2                 ]
                          [ -----      -----       1    0  ]
                          [ s + 1      s + 3               ]
rowperm below row 2       [                                ]
                          [            1 - s               ]
                          [   0    ------------   - 1   1  ]
                          [          2                     ]
                          [         s  + 4 s + 3           ]

                          [   1          2                           ]
                          [ -----      -----          1         0    ]
                          [ s + 1      s + 3                         ]
monic leadcoef row 2      [                                          ]
                          [            1 - s      - s - 3   s + 3    ]
                          [   0    ------------   -------   -----    ]
                          [          2             s + 1    s + 1    ]
                          [         s  + 2 s + 1                     ]

                          [   1          2                           ]
                          [ -----      -----          1         0    ]
                          [ s + 1      s + 3                         ]
Euclid div below row 2    [                                          ]
                          [            1 - s      - s - 3   s + 3    ]
                          [   0    ------------   -------   -----    ]
                          [          2             s + 1    s + 1    ]
                          [         s  + 2 s + 1                     ]
                          [   1         2                      ]
                          [ -----     -----        1       0   ]
                          [ s + 1     s + 3                    ]
Euclid div above row 2    [                                    ]
                          [             1 - s     - s - 3  s + 3 ]
                          [   0    ------------   -------  ----- ]
                          [            2           s + 1   s + 1 ]
                          [           s  + 2 s + 1               ]

              [   1         2      ]
              [ -----     -----    ]  [    1       0   ]
              [ s + 1     s + 3    ]  [                ]
(d39)        [[                    ], [ - s - 3  s + 3 ]]
              [           1 - s    ]  [ -------  ----- ]
              [   0    ------------]  [  s + 1   s + 1 ]
              [           2        ]
              [          s  + 2 s + 1 ]
```

Appendix 3 – Example 2

```
(c12) g:matrix([0,(s-2)/(s-1),2/(s+1)],
         [1/(s+1),1/(s-1),(s-1)/(s+1)],
         [1/(s-1),2/(s+2),1/(s+2)]);

                                    [          s - 2    2   ]
                                    [   0      -----  ----- ]
                                    [          s - 1  s + 1 ]
                                    [                       ]
                                    [   1        1    s - 1 ]
(d12)                               [ -----    -----  ----- ]
                                    [ s + 1    s - 1  s + 1 ]
                                    [                       ]
                                    [   1        2      1   ]
                                    [ -----    -----  ----- ]
                                    [ s - 1    s + 2  s + 2 ]

(c13) /* Fractional representation */
      a:ident(3)*(s-1)/(s+1);

                                    [ s - 1                 ]
                                    [ -----     0       0   ]
                                    [ s + 1                 ]
                                    [                       ]
                                    [         s - 1         ]
(d13)                               [   0     -----     0   ]
                                    [         s + 1         ]
                                    [                       ]
                                    [                 s - 1 ]
                                    [   0       0     ----- ]
                                    [                 s + 1 ]

(c14) b:a.g;

                        [                  s - 2              2 (s - 1)       ]
                        [    0             -----              ---------       ]
                        [                  s + 1                      2       ]
                        [                                      (s + 1)        ]
                        [                                                     ]
                        [                                            2        ]
(d14)                   [  s - 1             1                (s - 1)         ]
                        [ --------         -----              ---------       ]
                        [        2         s + 1                      2       ]
                        [ (s + 1)                              (s + 1)        ]
                        [                                                     ]
                        [    1         2 (s - 1)                s - 1         ]
                        [  -----   ---------------     --------------- ]
                        [  s + 1   (s + 1) (s + 2)     (s + 1) (s + 2) ]

(c16) /* Make left lambda prime factorization */
      m:addrow(transpose(a),transpose(b))$
(c17) ans:lambdahermite(m)$
(c18) l:transpose(part(ans,1))$
(c19) l:submatrix(l,4,5,6);

                                  [   1.000      0      0   ]
                                  [                         ]
                                  [            s - 1        ]
(d19)                             [ - 1.000    -----    0   ]
                                  [            s + 1        ]
                                  [                         ]
                                  [    0         0    1.000 ]
```

```
(c20) a0:invert(l).a;

                                   [ s - 1            ]
                                   [ -----   0    0   ]
                                   [ s + 1            ]
                                   [                  ]
(d20)                              [   1     1    0   ]
                                   [                  ]
                                   [             s - 1 ]
                                   [   0     0   ----- ]
                                   [             s + 1 ]

(c21) b0:ratsimp(invert(l).b);

                          [           s - 2        2 s - 2    ]
                          [   0       -----     ------------- ]
                          [           s + 1      2            ]
                          [                     s  + 2 s + 1  ]
                          [                                   ]
                          [   1                               ]
(d21)                     [ -----       1            1        ]
                          [ s + 1                             ]
                          [                                   ]
                          [   1     2 s - 2        s - 1      ]
                          [ -----  -------------  ------------- ]
                          [ s + 1   2              2            ]
                          [        s  + 3 s + 2   s  + 3 s + 2  ]

(c23) /* Represent system on Pernebos standard form */
      ans:lambdahermite(a0)$
(c24) a00:part(ans,1);

                                   [ 1    1      0   ]
                                   [                 ]
                                   [    1 - s        ]
                                   [ 0  -----    0   ]
(d24)                              [    s + 1        ]
                                   [                 ]
                                   [           s - 1 ]
                                   [ 0    0    ----- ]
                                   [           s + 1 ]

(c25) b00:ratsimp(part(ans,2).b0);

                   [       1                                          ]
                   [     -----            1              1            ]
                   [     s + 1                                        ]
                   [                                                  ]
                   [                                     2            ]
                   [     s - 1            1             s  - 2 s + 1  ]
(d25)              [ - -------------    - -----       - ------------- ]
                   [    2                 s + 1          2            ]
                   [   s  + 2 s + 1                     s  + 2 s + 1  ]
                   [                                                  ]
                   [       1          2 s - 2          s - 1          ]
                   [     -----     -------------    -------------     ]
                   [     s + 1      2                2                ]
                   [               s  + 3 s + 2     s  + 3 s + 2      ]

(c31) /* Extract submatrices */
       aa2:submatrix(1,a00,1);

                                   [ 1 - s        ]
                                   [ -----    0   ]
                                   [ s + 1        ]
(d31)                              [              ]
                                   [        s - 1 ]
                                   [   0    ----- ]
                                   [        s + 1 ]
```

```
(c34) bb2:submatrix(1,b00,3);

                               [      s - 1          1      ]
                               [ - ------------   - -----   ]
                               [    2               s + 1   ]
                               [   s  + 2 s + 1             ]
(d34)                          [                            ]
                               [       1          2 s - 2   ]
                               [     -----     ------------ ]
                               [     s + 1      2           ]
                               [               s  + 3 s + 2 ]

(c36) cc2:submatrix(1,b00,1,2);

                                  [    2          ]
                                  [   s  - 2 s + 1 ]
                                  [ - ------------ ]
                                  [    2          ]
(d36)                             [   s  + 2 s + 1 ]
                                  [               ]
                                  [     s - 1     ]
                                  [  ------------ ]
                                  [   2           ]
                                  [  s  + 3 s + 2  ]

(c37) /* Calculate right structure matrix */
      ans:lambdahermite(cc2)$
(c39) ctilde:simplify(invert(part(ans,2)));

                            [ 3.000 - 3.000 s                  ]
                            [ ---------------      1.000       ]
                            [ 1.000 s + 1.000                  ]
(d39)                       [                                  ]
                            [      3.000       0.333 s + 0.333 ]
                            [ ---------------  --------------- ]
                            [ 1.000 s + 2.000  1.000 s + 2.000 ]

(c40) chat:part(ans,1);

                                 [ 0.333 s - 0.333 ]
                                 [ --------------- ]
(d40)                            [      s + 1      ]
                                 [                 ]
                                 [        0        ]


(c45) /* Solve DAB identity for feedback design */
      mm2:addrow(transpose(aa2),transpose(bb2))$
(c46) ans:lambdahermite(mm2)$
(c47) part(ans,1);

                  [ 1.000    0   ]
                  [              ]
                  [   0    1.000 ]
(d47)             [              ]
                  [   0      0   ]
                  [              ]
                  [   0      0   ]

(c48) u:transpose(part(ans,2))$
```

```
(c49) u11:submatrix(3,4,u,3,4);

                [    3           2                                      ]
                [ - s  - 4.000 s  - 13.000 s + 6.000        2.000 ]
                [ --------------------------------------  - ----- ]
                [   3          2                                 s + 1 ]
(d49)           [  s  + 4.000 s  + 5.000 s + 2.000                  ]
                [                                                    ]
                [          4.000 s - 4.000                           ]
                [          ---------------                1.000  ]
                [                 2                                  ]
                [           s  + 3 s + 2                             ]

(c50) u12:submatrix(3,4,u,1,2);

                  [ 2.000 s - 2.000          2.000              ]
                  [ ---------------          -----              ]
                  [   2                      s + 1              ]
                  [  s  + 2 s + 1                               ]
(d50)             [                                             ]
                  [      2.000          4.000 s - 4.000         ]
                  [      -----       ---------------------      ]
                  [      s + 1        2                         ]
                  [                  s  + 3.000 s + 2.000       ]

(c51) u21:submatrix(1,2,u,3,4);

                        [ 8.000 s - 8.000          ]
                        [ ---------------  2.000   ]
(d51)                   [   2                      ]
                        [  s  + 3 s + 2            ]
                        [                          ]
                        [     - 2.000        0     ]

(c52) u22:submatrix(1,2,u,1,2);

                    [ 2.000 - 2.000 s                  ]
                    [ ---------------        0         ]
                    [      s + 1                       ]
(d52)               [                                  ]
                    [                2.000 - 2.000 s   ]
                    [       0        ---------------   ]
                    [                     s + 1        ]
```