$$\begin{bmatrix} \tilde{V}(s) & -\tilde{U}(s) \\ -\tilde{N}(s) & \tilde{M}(s) \end{bmatrix} \cdot \begin{bmatrix} M(s) & U(s) \\ N(s) & V(s) \end{bmatrix} = I^{(n)}$$

Algorithm:

(I) Find $M(s), N(s)$:

$$P(s) = \left[\begin{array}{c|c} A_p & B_p \\ \hline C_p & D_p \end{array}\right]$$

$$\left[P(s) = C_p \cdot (sI - A_p)^{-1} B_p + D_p \right]$$

Find a stabilizing ~~state~~ ~~feedback~~ F:

$$\left| \begin{array}{l} \dot{\underline{x}}_p = A_p \cdot \underline{x}_p + B_p \cdot \underline{e}_1 \\ \underline{y}_1 = C_p \cdot \underline{x}_p + D_p \cdot \underline{e}_1 \\ \underline{e}_1 = \underline{r} + F \cdot \underline{x}_p \end{array} \right|$$

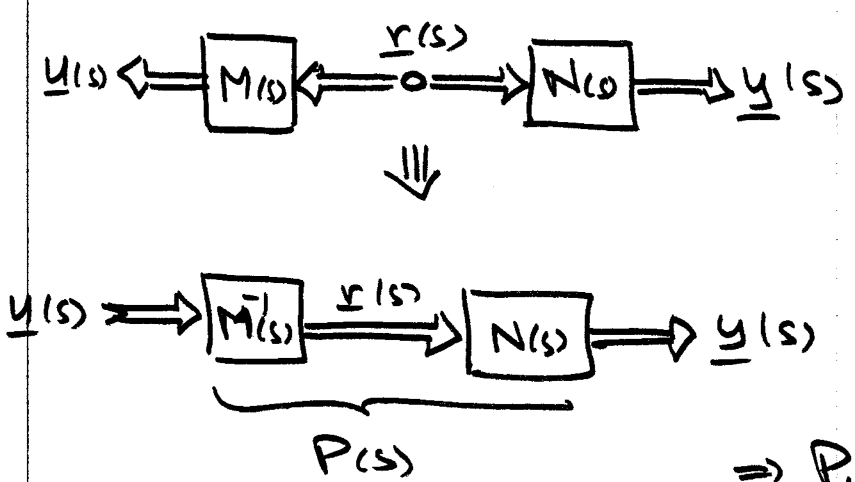
$$\Rightarrow \left| \begin{array}{l} \dot{\underline{x}}_p = (A_p + B_p \cdot F) \cdot \underline{x}_p + B_p \cdot \underline{r} \\ \underline{y}_1 = (C_p + D_p \cdot F) \cdot \underline{x}_p + D_p \cdot \underline{r} \\ \underline{e}_1 = F \cdot \underline{x}_p + I^{(m)} \cdot \underline{r} \end{array} \right|$$

Let: $\underline{u}(s) = M(s)\cdot\underline{r}(s)$

$$\Rightarrow \quad M(s) = \left[\begin{array}{c|c} A_p + B_p\cdot F & B_p \\ \hline F & I^{(m)} \end{array}\right]$$

Let: $\underline{y}(s) = N(s)\cdot\underline{r}(s)$

$$\Rightarrow \quad N(s) = \left[\begin{array}{c|c} A_p + B_p\cdot F & B_p \\ \hline C_p + D_p\cdot F & D_p \end{array}\right]$$



$$\Rightarrow \quad P(s) = N(s)\cdot M^{-1}(s)$$

- $N(s)$, $M(s)$ are both stable, since $\mathrm{eig}(A_p + B_p\cdot F) < \emptyset$ by design.

- $N(s)$, $M(s)$ are coprime (proof later).

(II) Find $U(s), V(s)$:

Design a stable observer L:

$$
\left|\begin{aligned}
\dot{\underline{x}}_p &= A_p \cdot \underline{x}_p + B_p \cdot \underline{e}_1 \\
\underline{y}_1 &= C_p \cdot \underline{x}_p + D_p \cdot \underline{e}_1 \\
\dot{\underline{\xi}} &= A_p \cdot \underline{\xi} + B_p \cdot \underline{e}_1 + L(\hat{\underline{y}} - \underline{y}_1) \\
\hat{\underline{y}} &= C_p \cdot \underline{\xi} + D_p \cdot \underline{e}_1 \\
\underline{e}_1 &= r + F \cdot \underline{\xi} \\
\underline{y}_2 &= F \cdot \underline{\xi}
\end{aligned}\right|
$$

$$
\Rightarrow \dot{\underline{\xi}} = A_p \cdot \underline{\xi} + B_p \cdot \underline{e}_1 + L \cdot C_p \underline{\xi} + L \cdot D_p \cdot \underline{e}_1 - L \cdot \underline{y}_1
$$

$$
= (A_p + L \cdot C_p)\underline{\xi} + (B_p + L \cdot D_p)\underline{e}_1 - L\underline{y}
$$

$$
\Rightarrow \dot{\underline{\xi}} = (A_p + B_p \cdot F + L \cdot C_p + L \cdot D_p \cdot F)\underline{\xi} + (B_p + L \cdot D_p)\underline{r} - L\underline{y}_1
$$

Superposition principle:

Set $w_1 = r = \phi$ for now:

$$\dot{\underline{\xi}} = (A_p + B_p \cdot F + L \cdot C_p + L \cdot D_p \cdot F)\underline{\xi} - L \cdot \underline{y}$$
$$\underline{y}_2 = F \cdot \underline{\xi}$$

$$\Rightarrow K(s) = \left[\begin{array}{c|c} A_p + B_p \cdot F + L \cdot C_p + L \cdot D_p \cdot F & -L \\ \hline F & \phi \end{array}\right]$$

$$K(s) \overset{!}{=} U(s) \cdot V^{-1}(s)$$

Introduce:

$$\underline{\eta} = (C_p + D_p \cdot F)\underline{\xi} - \underline{y}_1$$

$$\Rightarrow \dot{\underline{\xi}} = (A_p + B_p \cdot F)\underline{\xi} + L \cdot \underline{\eta}$$

Let: $\underline{y}_2(s) = (-U(s)) \cdot \underline{\eta}(s)$

$$\Rightarrow -U(s) = \left[\begin{array}{c|c} A_p + B_p \cdot F & L \\ \hline F & \phi \end{array}\right]$$

$$\Rightarrow U(s) = \left[\begin{array}{c|c} A_p + B_p \cdot F & -L \\ \hline F & \phi \end{array}\right]$$

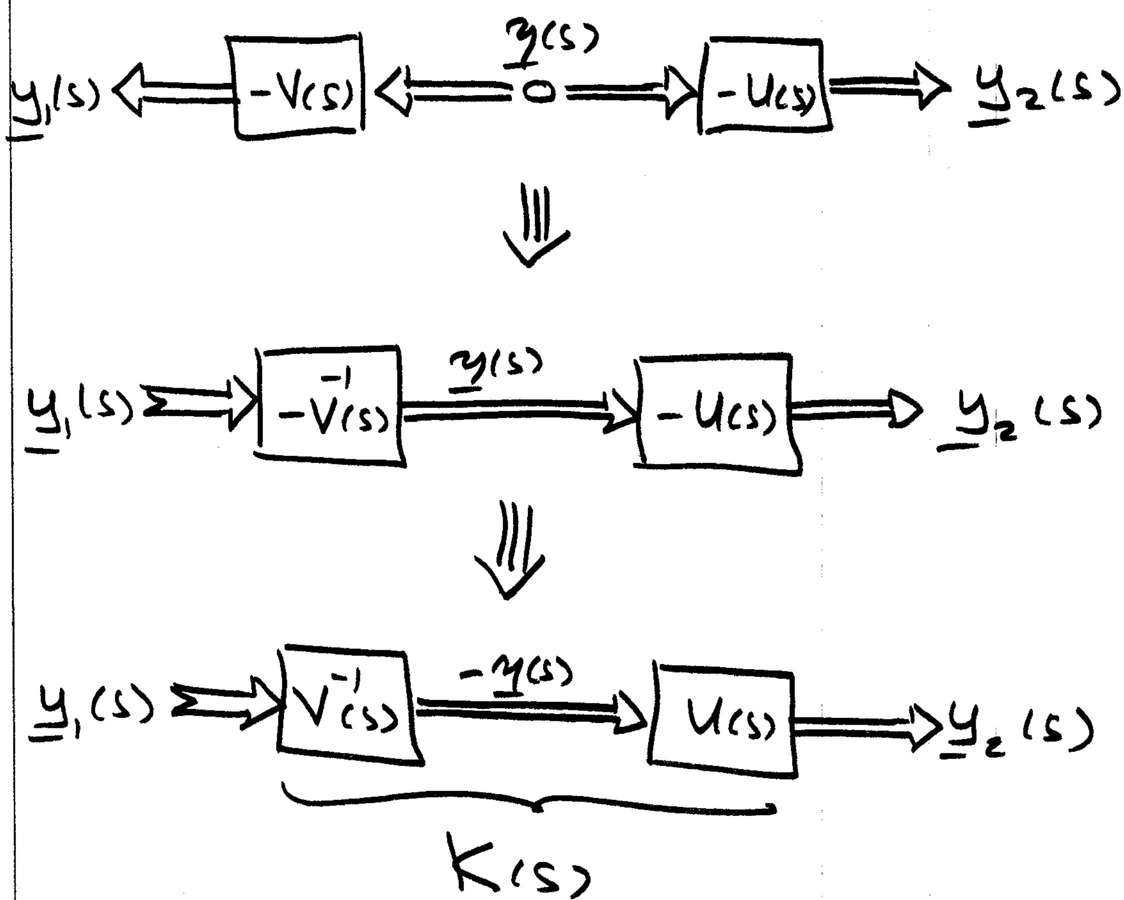
Let: $\underline{y}_1(s) = (-V(s)) \cdot \underline{\eta}(s)$

$$\underline{\eta} = (C_p + D_p \cdot F)\underline{\xi} - \underline{y}_1$$

$$\Rightarrow \underline{y}_1 = (C_p + D_p \cdot F)\underline{\xi} - \underline{\eta}$$

$$\Rightarrow -V(s) = \left[\begin{array}{c|c} A_p + B_p \cdot F & L \\ \hline C_p + D_p \cdot F & -I^{(p)} \end{array}\right]$$

$$\Rightarrow V(s) = \left[\begin{array}{c|c} A_p + B_p \cdot F & -L \\ \hline C_p + D_p \cdot F & I^{(p)} \end{array}\right]$$



$$\Rightarrow \underline{\underline{K(s) = U(s) \cdot V^{-1}(s)}}$$

✓

Left-Coprime Factorization:

$\Rightarrow$ Use duality principle:

$$\tilde{M}^*(s) = \left[\begin{array}{c|c} A_p^* + C_p^* \cdot L^* & C_p^* \\ \hline L^* & I^{(p)} \end{array}\right] \quad \text{etc.}$$

$$\Rightarrow \tilde{M}(s) = \left[\begin{array}{c|c} A_p + L \cdot C_p & L \\ \hline C_p & I^{(p)} \end{array}\right]$$

$$\tilde{N}(s) = \left[\begin{array}{c|c} A_p + L \cdot C_p & B_p + L \cdot D_p \\ \hline C_p & D_p \end{array}\right]$$

$$\tilde{U}(s) = \left[\begin{array}{c|c} A_p + L \cdot C_p & L \\ \hline -F & \phi \end{array}\right]$$

$$\tilde{V}(s) = \left[\begin{array}{c|c} A_p + L \cdot C_p & B_p + L \cdot D_p \\ \hline -F & I^{(m)} \end{array}\right]$$

```
function [N,M,U,V,K] = rcf(G,PcQ,PoRin,Rout)
%
%  This function operates on POLPAC system descriptions stored in modes 4..6.
%  It computes the right coprime factorization (RCF) of G.  It also computes
%  the optimal feedback matrix K.
%
%  Input Parameter:
%  ----------------
%
%    G       := Transfer function matrix (modes 4..6)
%    Pc/Q    := Pc is a column vector of controller pole locations
%               Q is a state weighting matrix for LQR
%    Po/Rin  := Po is a column vector of observer pole locations
%               Rin is an input weighting matrix for LQR
%    Rout    := Rout is an output weighting matrix for LQR
%
%  Output Parameter:
%  -----------------
%
%    N, M    := Right coprime factorization of G
%    U, V    := Right coprime factorization of K
%    K       := The optimal feedback system
%
%  Explanation:
%  ------------
%
%  Given a minimal transfer function matrix.  G is first converted to the
%  time domain.  Optimal feedback matrices F and L are computed using LQR
%  with Q = I, Rin = I, and Rout = I..  System representations are then found
%  for M, N, U, V.  These are converted back to the frequency domain (same
%  mode as G).  Then:  G(s) = N(s)*inv(M(s)), and
%                      K(s) = U(s)*inv(V(s)).
%
%  Defaults:
%  ---------
%
%  Only the first input parameter is mandated.  By default, the stabilizing
%  output feedback controller will be computed by LQR using identity matrices
%  for Q, Rin, and Rout.
%
%  If three input parameters are provided, the second should be the pole
%  locations of the controller poles, and the third should be the pole
%  locations of the observer poles.  The PLACE routine will be used to
%  compute the stabilizing output feedback controller.
%
%  If four input parameters are provided, the second should be the state
%  weighting matrix, the third the input wieghting matrix, and the fourth
%  the output weighting matrix.  LQR will be used to compute the stabilizing
%  output feedback controller.
%
push('RCF')
global debg
%
%  Start by unpacking the structural information
%
[mode,logcol,dtype,stype] = unpack(G(1,1));
[p,m] = logdim(G);
dom = domain(G);
%
%  Check for consistency
```

```
%
if debg == 3,
  if mode < 4,
    disp('RCF: Error - Operates on modes 4..6 only'),
    abort,
  end,
  if mode > 6,
    disp('RCF: Error - Operates on modes 4..6 only'),
    abort,
  end,
end
%
%  Convert to time domain
%
Sg = tfm2ss(G);
Sg = minreals(Sg);
%
%  Extract system matrices
%
[A,B,C,D] = oldsyst(Sg);
[n,n] = size(A);
%
%  Compute optimal control
%
if nargin == 3,
  F = -place(A,B,PcQ);
  L = -place(A',C',PoRin)';
  else
  if nargin == 1,
    PcQ = eye(n);
    PcRin = eye(m);
    Rout = eye(p);
  end,
  F = -lqr(A,B,PcQ,PcRin);
  L = -lqr(A',C',PcQ,Rout)';
end
%
%  Now compute the four systems.
%
Am = A + B*F;
Bm = B;
Cm = F;
Dm = eye(m);
%
An = Am;
Bn = B;
Cn = C + D*F;
Dn = D;
%
Au = Am;
Bu = -L;
Cu = F;
Du = zeros(m,p);
%
Av = Am;
Bv = Bu;
Cv = Cn;
Dv = eye(p);
%
%  Make into systems
```

```
%
Sm = newsyst(Am,Bm,Cm,Dm,stype);
Sn = newsyst(An,Bn,Cn,Dn,stype);
Su = newsyst(Au,Bu,Cu,Du,stype);
Sv = newsyst(Av,Bv,Cv,Dv,stype);
%
%  Convert back to the frequency domain
%
M = ss2tfm(Sm,mode,dom);
N = ss2tfm(Sn,mode,dom);
U = ss2tfm(Su,mode,dom);
V = ss2tfm(Sv,mode,dom);
M = reduceg(M);
N = reduceg(N);
U = reduceg(U);
V = reduceg(V);
%
%  Check whether we are done
%
if nargout == 5,
  %
  %  We still need to compute K
  %
  Ak = A + B*F + L*C + L*D*F;
  Bk = -L;
  Ck = F;
  Dk = zeros(m,p);
  Sk = newsyst(Ak,Bk,Ck,Dk,stype);
  K = ss2tfm(Sk,mode,dom);
  K = reduceg(K);
end
%
pull
return
```

```
function [N,M,U,V,K] = lcf(G,PcQ,PoRin,Rout)
%
%  This function operates on POLPAC system descriptions stored in modes 4..6.
%  It computes the left coprime factorization (RCF) of G.  It also computes
%  the optimal feedback matrix K.
%
%  Input Parameter:
%  ----------------
%
%    G       := Transfer function matrix (modes 4..6)
%    Pc/Q    := Pc is a column vector of controller pole locations
%               Q is a state weighting matrix for LQR
%    Po/Rin  := Po is a column vector of observer pole locations
%               Rin is an input weighting matrix for LQR
%    Rout    := Rout is an output weighting matrix for LQR
%
%  Output Parameter:
%  -----------------
%
%    N, M    := Left coprime factorization of G
%    U, V    := Left coprime factorization of K
%    K       := The optimal feedback system
%
%  Explanation:
%  ------------
%
%  Given a minimal transfer function matrix.  G is first converted to the
%  time domain.  Optimal feedback matrices F and L are computed using LQR
%  with Q = I, Rin = I, and Rout = I..  System representations are then found
%  for M, N, U, V.  These are converted back to the frequency domain (same
%  mode as G).  Then:  G(s) = inv(M(s))*N(s), and
%                      K(s) = inv(V(s))*U(s).
%
%  Defaults:
%  ---------
%
%  Only the first input parameter is mandated.  By default, the stabilizing
%  output feedback controller will be computed by LQR using identity matrices
%  for Q, Rin, and Rout.
%
%  If three input parameters are provided, the second should be the pole
%  locations of the controller poles, and the third should be the pole
%  locations of the observer poles.  The PLACE routine will be used to
%  compute the stabilizing output feedback controller.
%
%  If four input parameters are provided, the second should be the state
%  weighting matrix, the third the input wieghting matrix, and the fourth
%  the output weighting matrix.  LQR will be used to compute the stabilizing
%  output feedback controller.
%
push('LCF')
global debg
%
%  Start by unpacking the structural information
%
[mode,logcol,dtype,stype] = unpack(G(1,1));
[p,m] = logdim(G);
dom = domain(G);
%
%  Check for consistency
```

```
%
if debg == 3,
  if mode < 4,
    disp('LCF: Error - Operates on modes 4..6 only'),
    abort,
  end,
  if mode > 6,
    disp('LCF: Error - Operates on modes 4..6 only'),
    abort,
  end,
end
%
%  Convert to time domain
%
Sg = tfm2ss(G);
Sg = minreals(Sg);
%
%  Extract system matrices
%
[A,B,C,D] = oldsyst(Sg);
[n,n] = size(A);
%
%  Compute optimal control
%
if nargin == 3,
  F = -place(A,B,PcQ);
  L = -place(A',C',PoRin)';
  else
  if nargin == 1,
    PcQ = eye(n);
    PcRin = eye(m);
    Rout = eye(p);
  end,
  F = -lqr(A,B,PcQ,PcRin);
  L = -lqr(A',C',PcQ,Rout)';
end
%
%  Now compute the four systems.
%
Am = A + L*C;
Bm = L;
Cm = C;
Dm = eye(p);
%
An = Am;
Bn = B + L*D;
Cn = C;
Dn = D;
%
Au = Am;
Bu = L;
Cu = -F;
Du = zeros(m,p);
%
Av = Am;
Bv = Bn;
Cv = Cu;
Dv = eye(m);
%
%  Make into systems
```

```
%
Sm = newsyst(Am,Bm,Cm,Dm,stype);
Sn = newsyst(An,Bn,Cn,Dn,stype);
Su = newsyst(Au,Bu,Cu,Du,stype);
Sv = newsyst(Av,Bv,Cv,Dv,stype);
%
%  Convert back to the frequency domain
%
M = ss2tfm(Sm,mode,dom);
N = ss2tfm(Sn,mode,dom);
U = ss2tfm(Su,mode,dom);
V = ss2tfm(Sv,mode,dom);
M = reduceg(M);
N = reduceg(N);
U = reduceg(U);
V = reduceg(V);
%
%  Check whether we are done
%
if nargout == 5,
  %
  %  We still need to compute K
  %
  Ak = A + B*F + L*C + L*D*F;
  Bk = -L;
  Ck = F;
  Dk = zeros(m,p);
  Sk = newsyst(Ak,Bk,Ck,Dk,stype);
  K = ss2tfm(Sk,mode,dom);
  K = reduceg(K);
end
%
pull
return
```

```
%
%  Test [5.3]: Coprime factorizations of a POLPAC System
%  -------------------------------------------------------
%
polpac_init
global debg
debg = 3;
%
%  Enter the system
%
A = [ 0      1      0
      2      1      0
      0      0      1 ];
B = [ 0      0
      1      0
      2      1 ];
C = [ 1      0      1
      0      1      1 ];
D = zeros(2);
n = 3;
m = 2;
p = 2;
stype = 0;
mode = 4;
dom = 0;
S = newsyst(A,B,C,D,stype);
Gm = ss2tfm(S,mode,dom);
Gm = reduceg(Gm);
show(Gm,'Gm')
Gm(s) =
   (2*s^2 - s - 5)          (1)
---------------------     -------
(s^3 - 2*s^2 - s + 2)     (s - 1)

  (3*s^2 - 3*s - 4)         (1)
---------------------     -------
(s^3 - 2*s^2 - s + 2)     (s - 1)

%
[N,M,U,V,K] = rcf(Gm);
show(N,'N')
N(s) =
     (2*s^2 + 2.936*s + 0.3798)                (s^2 + 5.707*s + 5.625)
-----------------------------------     -----------------------------------
(s^3 + 4.278*s^2 + 5.599*s + 2.384)     (s^3 + 4.278*s^2 + 5.599*s + 2.384)

     (3*s^2 + 2.424*s - 0.1088)                (s^2 + 6.666*s + 4.667)
-----------------------------------     -----------------------------------
(s^3 + 4.278*s^2 + 5.599*s + 2.384)     (s^3 + 4.278*s^2 + 5.599*s + 2.384)

show(M,'M')
M(s) =
(s^3 - 0.5115*s^2 - 2.489*s - 0.9771)     (0.9588*s^2 - 0.9588*s - 1.918)
-------------------------------------   -----------------------------------
 (s^3 + 4.278*s^2 + 5.599*s + 2.384)    (s^3 + 4.278*s^2 + 5.599*s + 2.384)

   (0.9588*s^2 + 2.932*s + 2.063)       (s^3 + 2.79*s^2 + 0.877*s - 0.8316)
 -----------------------------------    -----------------------------------
 (s^3 + 4.278*s^2 + 5.599*s + 2.384)    (s^3 + 4.278*s^2 + 5.599*s + 2.384)
```

```
show(U,'U')
U(s) =
    (56.21*s^2 + 37.9*s - 18.27)          (-135.2*s^2 - 117.6*s + 17.71)
-----------------------------------   -----------------------------------
(s^3 + 4.278*s^2 + 5.599*s + 2.384)   (s^3 + 4.278*s^2 + 5.599*s + 2.384)

   (-40.04*s^2 - 126.4*s - 86.93)         (83.13*s^2 + 262.3*s + 180.9)
-----------------------------------   -----------------------------------
(s^3 + 4.278*s^2 + 5.599*s + 2.384)   (s^3 + 4.278*s^2 + 5.599*s + 2.384)

show(V,'V')
V(s) =
  (s^3 + 9.695*s^2 + 103.8*s + 12)        (2.629*s^2 - 167.6*s - 222.1)
-----------------------------------   -----------------------------------
(s^3 + 4.278*s^2 + 5.599*s + 2.384)   (s^3 + 4.278*s^2 + 5.599*s + 2.384)

   (2.629*s^2 + 137.9*s + 110.6)        (s^3 + 12.8*s^2 - 258.3*s - 220.2)
-----------------------------------   -----------------------------------
(s^3 + 4.278*s^2 + 5.599*s + 2.384)   (s^3 + 4.278*s^2 + 5.599*s + 2.384)

show(K,'K')
K(s) =
   (56.21*s^2 + 872.2*s + 866.6)           (-135.2*s^2 - 998*s - 810.9)
-----------------------------------   -----------------------------------
(s^3 + 18.21*s^2 - 120.9*s - 785.9)   (s^3 + 18.21*s^2 - 120.9*s - 785.9)

    (-40.04*s^2 - 686*s - 360.1)           (83.13*s^2 + 817.9*s + 1009)
-----------------------------------   -----------------------------------
(s^3 + 18.21*s^2 - 120.9*s - 785.9)   (s^3 + 18.21*s^2 - 120.9*s - 785.9)

%
[Nt,Mt,Ut,Vt,Kt] = lcf(Gm);
show(Nt,'Nt')
Nt(s) =
     (2*s^2 + 8.147*s + 13.28)              (s^2 + 4.888*s + 5.824)
----------------------------------   ----------------------------------
(s^3 + 11.93*s^2 + 35.22*s + 31.1)   (s^3 + 11.93*s^2 + 35.22*s + 31.1)

     (3*s^2 + 7.993*s - 5.975)             (s^2 + 1.788*s - 0.2789)
----------------------------------   ----------------------------------
(s^3 + 11.93*s^2 + 35.22*s + 31.1)   (s^3 + 11.93*s^2 + 35.22*s + 31.1)

show(Mt,'Mt')
Mt(s) =
(s^3 + 6.517*s^2 + 4.195*s - 3.259)      (-2.629*s^2 - 3.259*s - 2.566)
-----------------------------------    ----------------------------------
 (s^3 + 11.93*s^2 + 35.22*s + 31.1)    (s^3 + 11.93*s^2 + 35.22*s + 31.1)

   (-2.629*s^2 + 7.138*s + 10.83)     (s^3 + 3.417*s^2 - 9.205*s - 10.56)
 ----------------------------------   -----------------------------------
 (s^3 + 11.93*s^2 + 35.22*s + 31.1)    (s^3 + 11.93*s^2 + 35.22*s + 31.1)

show(Ut,'Ut')
Ut(s) =
   (56.21*s^2 + 826.9*s + 858.3)        (-135.2*s^2 - 876.4*s - 653.8)
----------------------------------   ----------------------------------
(s^3 + 11.93*s^2 + 35.22*s + 31.1)   (s^3 + 11.93*s^2 + 35.22*s + 31.1)

  (-40.04*s^2 - 548.2*s - 568.5)         (83.13*s^2 + 549.4*s + 406.2)
----------------------------------   ----------------------------------
```

```
(s^3 + 11.93*s^2 + 35.22*s + 31.1)    (s^3 + 11.93*s^2 + 35.22*s + 31.1)

show(Vt,'Vt')
Vt(s) =
 (s^3 + 16.72*s^2 - 189.5*s - 851)       (-0.9588*s^2 - 91.44*s - 174.6)
-----------------------------------    -----------------------------------
(s^3 + 11.93*s^2 + 35.22*s + 31.1)     (s^3 + 11.93*s^2 + 35.22*s + 31.1)

   (-0.9588*s^2 + 153*s + 576.7)      (s^3 + 13.42*s^2 + 97.56*s + 147.1)
-----------------------------------    -----------------------------------
(s^3 + 11.93*s^2 + 35.22*s + 31.1)     (s^3 + 11.93*s^2 + 35.22*s + 31.1)

show(Kt,'Kt')
Kt(s) =
   (56.21*s^2 + 872.2*s + 866.6)             (-135.2*s^2 - 998*s - 810.9)
------------------------------------    ------------------------------------
(s^3 + 18.21*s^2 - 120.9*s - 785.9)     (s^3 + 18.21*s^2 - 120.9*s - 785.9)

    (-40.04*s^2 - 686*s - 360.1)            (83.13*s^2 + 817.9*s + 1009)
------------------------------------    ------------------------------------
(s^3 + 18.21*s^2 - 120.9*s - 785.9)     (s^3 + 18.21*s^2 - 120.9*s - 785.9)

%
B11 = sub(multg(Vt,M),multg(Ut,N));
B11 = reduceg(B11);
show(B11,'B11')
B11(s) =
(1)    (0)
(0)    (1)
%
B12 = sub(multg(Vt,U),multg(Ut,V));
B12 = reduceg(B12);
show(B12,'B12')
B12(s) =
(0)    (0)
(0)    (0)
%
B21 = sub(multg(Mt,N),multg(Nt,M));
B21 = reduceg(B21);
show(B21,'B21')
B21(s) =
(0)    (0)
(0)    (0)
%
B22 = sub(multg(Mt,V),multg(Nt,U));
B22 = reduceg(B22);
show(B22,'B22')
B22(s) =
(1)    (0)
(0)    (1)
%
show(Gm,'Gm')
Gm(s) =
   (2*s^2 - s - 5)        (1)
---------------------    -------
(s^3 - 2*s^2 - s + 2)    (s - 1)

  (3*s^2 - 3*s - 4)       (1)
---------------------    -------
(s^3 - 2*s^2 - s + 2)    (s - 1)
```

```
Gm2 = multg(N,invg(M));
Gm2 = reduceg(Gm2);
show(Gm2,'Gm2')
Gm2(s) =
    (2*s^2 - s - 5)          (1)
---------------------     -------
(s^3 - 2*s^2 - s + 2)     (s - 1)

  (3*s^2 - 3*s - 4)          (1)
---------------------     -------
(s^3 - 2*s^2 - s + 2)     (s - 1)

Gm3 = multg(invg(Mt),Nt);
Gm3 = reduceg(Gm3);
show(Gm3,'Gm3')
Gm3(s) =
    (2*s^2 - s - 5)          (1)
---------------------     -------
(s^3 - 2*s^2 - s + 2)     (s - 1)

  (3*s^2 - 3*s - 4)          (1)
---------------------     -------
(s^3 - 2*s^2 - s + 2)     (s - 1)

%
show(K,'K')
K(s) =
   (56.21*s^2 + 872.2*s + 866.6)           (-135.2*s^2 - 998*s - 810.9)
-----------------------------------   -----------------------------------
(s^3 + 18.21*s^2 - 120.9*s - 785.9)   (s^3 + 18.21*s^2 - 120.9*s - 785.9)

    (-40.04*s^2 - 686*s - 360.1)           (83.13*s^2 + 817.9*s + 1009)
-----------------------------------   -----------------------------------
(s^3 + 18.21*s^2 - 120.9*s - 785.9)   (s^3 + 18.21*s^2 - 120.9*s - 785.9)

K2 = multg(U,invg(V));
K2 = reduceg(K2);
show(K2,'K2')
K2(s) =
SHOW1: Warning - Column too broad. Cannot properly display
              Col.  1

   (56.21*s^2 + 872.2*s + 866.6)
-----------------------------------
(s^3 + 18.21*s^2 - 120.9*s - 785.9)

    (-40.04*s^2 - 686*s - 360.1)
-----------------------------------
(s^3 + 18.21*s^2 - 120.9*s - 785.9)


                                                              Col.  2

(-135.2*s^8 - 2155*s^7 - 1.334e+04*s^6 - 4.35e+04*s^5 - 8.349e+04*s^4 - 9.797e
-------------------------------------------------------------------------------
  (s^9 + 26.77*s^8 + 64.42*s^7 - 1231*s^6 - 9281*s^5 - 2.859e+04*s^4 - 4.716e+
                                                    (83.13*s^2 + 817.9*s + 1009)
                                                ---------------------------------
```

```
                                  (s^3 + 18.21*s^2 - 120.9*s - 785.9)


K3 = multg(invg(Vt),Ut);
K3 = reduceg(K3);
show(K3,'K3')
K3(s) =
SHOW1: Warning - Column too broad. Cannot properly display
              Col.  1

   (56.21*s^2 + 872.2*s + 866.6)
-----------------------------------
(s^3 + 18.21*s^2 - 120.9*s - 785.9)

    (-40.04*s^2 - 686*s - 360.1)
-----------------------------------
(s^3 + 18.21*s^2 - 120.9*s - 785.9)


                                                              Col.  2

                                                     (-135.2*s^2 - 998*s - 810.9
                                                 -------------------------------
                                                 (s^3 + 18.21*s^2 - 120.9*s - 78

(83.13*s^8 + 2802*s^7 + 3.823e+04*s^6 + 2.732e+05*s^5 + 1.118e+06*s^4 + 2.714e
--------------------------------------------------------------------------------
(s^9 + 42.08*s^8 + 526.7*s^7 + 1108*s^6 - 2.607e+04*s^5 - 2.382e+05*s^4 - 9.08


%
diary off
```