APPENDIX A

This appendix contains function–descriptions of the MATLAB–functions gap and dirgap.

gap

Purpose :
Computation of the gap between two systems.

Synopsis :
[g1,g2,dist] = gap(A1,B1,C1,D1,A2,B2,C2,D2)

Description :
Given minimal realizations [A1,B1,C1,D1] and [A2,B2,C2,D2] of the plants P_1 and P_2 respectively, gap calculates g1, the directed gap between P_1 and P_2, and g2, the directed gap between P_2 and P_1, within an accuracy level, given by tol. The gap between P_1 and P_2, which equals the maximum of the two directed gaps, is returned in the variable dist. So in formulae we have:

$$g1 = \vec{\delta}(P_1,P_2),$$
$$g2 = \vec{\delta}(P_2,P_1),$$
$$dist = \delta(P_1,P_2) = \max \{ \vec{\delta}(P_1,P_2),\vec{\delta}(P_2,P_1) \},$$

where we didn't take the accuracy level tol into account.

Algorithm :
gap uses the method given in [6, ch.2] to calculate the gap between two systems. Let (D_i , N_i) and $(\tilde{N}_i , \tilde{D}_i)$ be normalized right– and left–Bezout factorizations of P_i respectively (i = 1,2). Then, according to (2.17), we have:

$$\vec{\delta}(P_1,P_2) = \inf_{Q \in M(\mathbb{R}H_\infty)} \left\| \begin{bmatrix} D_1 \\ N_1 \end{bmatrix} - \begin{bmatrix} D_2 \\ N_2 \end{bmatrix} Q \right\|_\infty. \qquad (A.1)$$

When we define

$$G := D_2^* D_1 + N_2^* N_1, \qquad (A.2)$$

$$J_1 := -\tilde{N}_2 D_1 + \tilde{D}_2 N_1, \tag{A.3}$$

we can rewrite (A.1) as (see [6, p.5])

$$\vec{\delta}(P_1,P_2) = \inf_{Q \in M(\mathbb{RH}_\infty)} \left\| \begin{bmatrix} G - Q \\ J_1 \end{bmatrix} \right\|. \tag{A.4}$$

Given state–space realizations of the plants P_1 and P_2 it is possible to construct state–space realizations of the matrices G and J_1, as defined in (A.2) and (A.3) (with help of the functions norcor, tfmult and tfadd). Given these realizations of G and J_1, the MATLAB–function dirgap is used to calculate the directed gap $\vec{\delta}(P_1,P_2)$ within the desired level of accuracy.
The calculation of the directed gap between P_2 and P_1 takes place in the same way, where G and J_1 are replaced by G^* and $J_2 := -\tilde{N}_1 D_2 + \tilde{D}_1 N_2$ respectively, i.e.

$$\vec{\delta}(P_2,P_1) = \inf_{Q \in M(\mathbb{RH}_\infty)} \left\| \begin{bmatrix} G^* - Q \\ J_2 \end{bmatrix} \right\|. \tag{A.5}$$

Finally the value of dist, the gap between P_1 and P_2, is calculated as the maximum of the two directed gaps g1 and g2.

WARNING: gap is not suitable to compute gaps between large order systems, certainly not in combination with a small value for tol, the desired accuracy level. The computation becomes then very expensive.

Nested functions :
norcor; see appendix B. dirgap; see appendix A.
tfadd, tfmult; see appendix E.

Program :

```
function [g1,g2,dist] = gap(A1,B1,C1,D1,A2,B2,C2,D2,tol)
```

Given minimal realizations of P1=[A1,B1,C1,D1] and P2=[A2,B2,C2,D2] gap calculates dist, the gap between P1 and P2, where the tolerance for the accuracy is equal to tol. g1 and g2 are the directed gaps between P1 and P2 respectively.

```
[DRA1,DRB1,DRC1,DRD1,NRA1,NRB1,NRC1,NRD1,..
 DLA1,DLB1,DLC1,DLD1,NLA1,NLB1,NLC1,NLD1]=norcor(A1,B1,C1,D1);
```

```
[DRA2,DRB2,DRC2,DRD2,NRA2,NRB2,NRC2,NRD2,..
 DLA2,DLB2,DLC2,DLD2,NLA2,NLB2,NLC2,NLD2]=norcor(A2,B2,C2,D2);

[H1,H2,H3,H4]=tfmult(-DRA2',DRC2',-DRB2',DRD2',DRA1,DRB1,DRC1,DRD1);
[H5,H6,H7,H8]=tfmult(-NRA2',NRC2',-NRB2',NRD2',NRA1,NRB1,NRC1,NRD1);
[GA,GB,GC,GD]=tfadd(H1,H2,H3,H4,H5,H6,H7,H8);
GSA=-GA';GSB=GC';GSC=-GB';GSD=GD';

[H1,H2,H3,H4]=tfmult(NLA2,NLB2,-NLC2,-NLD2,DRA1,DRB1,DRC1,DRD1);
[H5,H6,H7,H8]=tfmult(DLA2,DLB2,DLC2,DLD2,NRA1,NRB1,NRC1,NRD1);
[JA1,JB1,JC1,JD1]=tfadd(H1,H2,H3,H4,H5,H6,H7,H8);

[H1,H2,H3,H4]=tfmult(NLA1,NLB1,-NLC1,-NLD1,DRA2,DRB2,DRC2,DRD2);
[H5,H6,H7,H8]=tfmult(DLA1,DLB1,DLC1,DLD1,NRA2,NRB2,NRC2,NRD2);
[JA2,JB2,JC2,JD2]=tfadd(H1,H2,H3,H4,H5,H6,H7,H8);

g1=dirgap(GA,GB,GC,GD,JA1,JB1,JC1,JD1,tol);
g2=dirgap(GSA,GSB,GSC,GSD,JA2,JB2,JC2,JD2,tol);
dist=max([g1  g2]);
```

dirgap

Purpose :
Computation of the directed gap between two systems. dirgap is a special purpose algorithm, to be used in the MATLAB–function gap.

Synopsis :
alpha = dirgap(GA,GB,GC,GD,JA,JB,JC,JD,tol)

Description :
Given realizations [GA,GB,GC,GD] of a matrix G, and [JA,JB,JC,JD] of a matrix J, dirgap computes, under the assumption that

$$\inf_{Q \in M(\mathbb{R}H_\infty)} \left\| \begin{bmatrix} G - Q \\ J \end{bmatrix} \right\| \leq 1,$$

the value of this infimum (within an accuracy level of tol), and stores it in the variable alpha:

$$\text{alpha} = \inf_{Q \in M(\mathbb{R}H_\infty)} \left\| \begin{bmatrix} G - Q \\ J \end{bmatrix} \right\|. \qquad \text{(A.6)}$$

The relation of this formula with the directed gap is already explained in the function–description of the function gap. Note that in the case where G and J are defined as in (A.2) and (A.3), the value of alpha is equal to a directed gap, and so smaller or equal to one. So in this case (the only one we are interested in), the assumption

$$\inf_{Q \in M(\mathbb{R}H_\infty)} \left\| \begin{bmatrix} G - Q \\ J \end{bmatrix} \right\| \leq 1,$$

is always satisfied.

Algorithm :
dirgap is a computer–implementation of the algorithm for the calculation of formula (A.6) as given in [6, p.6], which is based upon the algorithm of Francis described in [4, sec.8.1.]. It proceeds as follows.
From formula (A.6) and our assumption it follows immediately that

$$\| J \| \leq \text{alpha} \leq 1.$$

Now define $\mu := \| J \|$ and suppose $\mu < \gamma < 1$. Let F_γ be a spectral factor of the matrix

$$\gamma^2 I - J^* J,$$

i.e.

$$F_\gamma^* F_\gamma = \gamma^2 I - J^* J, \qquad \text{(A.7)}$$

with F_γ and $F_\gamma^{-1} \in M(\mathbb{R}H_\infty)$. In [4, sec.7.3.] it is proven that such a factorization really exists. Now it can be shown that (see [4, sec.8.1.]) that

$$\inf_{Q \in M(\mathbb{R}H_\infty)} \left\| \begin{bmatrix} G - Q \\ J \end{bmatrix} \right\| \leq \gamma \quad \Leftrightarrow \quad \| \Gamma_{GF_\gamma^{-1}} \| \leq 1. \qquad \text{(A.8)}$$

where $\Gamma_{GF_\gamma^{-1}}$ denotes the Hankel–operator with symbol GF_γ^{-1}. Because we know how to calculate the norm of this Hankel–operator (see the MATLAB–function hankel), it is now possible, with help of formula (A.8), to approximate the value of alpha to any wished accuracy by performing an iterative search on the interval $[\mu,1]$. Bisection is used to carry out this search.

WARNING: Remark that in each step of the algorithm a quite expensive spectral factorization has to be carried out. Therefore this method is not very good applicable for matrices of a large order, and certainly not in combination with a small value for tol, the desired level of accuracy.

Nested functions :
hinfn; see [1]; hankel; see appendix C.
tfinv, tfmult; see appendix E.

Program :

```
function alpha=dirgap(GA,GB,GC,GD,JA,JB,JC,JD,tol)
```

Given realizations [GA,GB,GC,GD] of G and [JA,JB,JC,JD] of J, dirgap calculates:

$$alpha = \inf_{Q \in M(\mathbb{R}H_\infty)} \left\| \begin{bmatrix} G \\ J \end{bmatrix} - Q \right\|_\infty,$$

within an accuracy of tol. (This under the assumption alpha ≤ 1.)

```
[mu,freq,l]=hinfn(JA,JB,JC,JD,1e-8,0);
under=mu;upper=1;

while ((upper-under)>tol)
      gamma=(upper+under)/2;
      [FA,FB,FCh,FDh]=sfl(JA,JB,((1/gamma)*JC),((1/gamma)*JD));
      FC=gamma*FCh;FD=gamma*FDh;
      [FIA,FIB,FIC,FID]=tfinv(FA,FB,FC,FD);
      [RA,RB,RC,RD]=tfmult(GA,GB,GC,GD,FIA,FIB,FIC,FID);
      han=hankel(RA,RB,RC,RD);
      if han ≤ 1
            upper=gamma;
      else
            under=gamma;
      end
end

alpha=(upper+under)/2;
```

APPENDIX B

This appendix contains function–descriptions of the MATLAB–functions ncoprfac, norcor and speccopr.

ncoprfac, norcor

Purpose :
Normalized doubly–Bezout factorization.

Synopsis :

```
[DRA,DRB,DRC,DRD,NRA,NRB,NRC,NRD,..
 YRA,YRB,YRC,YRD,ZRA,ZRB,ZRC,ZRD,..
 DLA,DLB,DLC,DLD,NLA,NLB,NLC,NLD,..
 YLA,YLB,YLC,YLD,ZLA,ZLB,ZLC,ZLD] = ncoprfac(A,B,C,D)

[DRA,DRB,DRC,DRD,NRA,NRB,NRC,NRD,..
 DLA,DLB,DLC,DLD,NLA,NLB,NLC,NLD] = norcor(A,B,C,D)
```

Description :
Given a minimal realization [A,B,C,D] of a transfermatrix P(s), ncoprfac computes state–space realizations of a normalized right– and left–Bezout factorization ((D_0,N_0) and $(\tilde{N}_0,\tilde{D}_0)$ respectively) and of matrices $Y_0,Z_0,\tilde{Y}_0$ and $\tilde{Z}_0$ such that:

$$\begin{bmatrix} -Z_0 & Y_0 \\ \tilde{D}_0 & \tilde{N}_0 \end{bmatrix} \begin{bmatrix} -N_0 & \tilde{Y}_0 \\ D_0 & \tilde{Z}_0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \qquad (B.1)$$

These state–space realizations are given by

$D_0(s)$ = [DRA,DRB,DRC,DRD]; $\quad\quad$ $\tilde{D}_0(s)$ = [DLA,DLB,DLC,DLD];

$N_0(s)$ = [NRA,NRB,NRC,NRD]; $\quad\quad$ $\tilde{N}_0(s)$ = [NLA,NLB,NLC,NLD];

$Y_0(s)$ = [YRA,YRB,YRC,YRD]; $\quad\quad$ $\tilde{Y}_0(s)$ = [YLA,YLB,YLC,YLD];

$Z_0(s)$ = [ZRA,ZRB,ZRC,ZRD]; $\quad\quad$ $\tilde{Z}_0(s)$ = [ZLA,ZLB,ZLC,ZLD].

norcor is an abbreviated version of ncoprfac and produces only state–space realizations of the normalized right– and left–Bezout factorizations (D_0,N_0) and $(\tilde{N}_0,\tilde{D}_0)$ of P.

Algorithm :

ncoprfac and norcor use the construction method for state–space realizations of normalized Bezout factorizations as given in theorem 4.1.1.

Nested functions :

sqrtl; see appendix E.

See also :

speccopr (see appendix B); this is a special purpose algorithm to be used in other algorithms.

Programs :

```
function    [DRA,DRB,DRC,DRD,NRA,NRB,NRC,NRD,..
             YRA,YRB,YRC,YRD,ZRA,ZRB,ZRC,ZRD,..
             DLA,DLB,DLC,DLD,NLA,NLB,NLC,NLD,..
             YLA,YLB,YLC,YLD,ZLA,ZLB,ZLC,ZLD]=ncoprfac(A,B,C,D)
```

Given a minimal realization [A,B,C,D] of a transfermatrix P(s), ncoprfac computes realizations of a normalized right coprime factorization of P. $P=ND^{-1}$ with D=[DRA,DRB,DRC,DRD] and N=[NRA,NRB, NRC,NRD]; Y=[YRA,YRB,YRC,YRD] and Z=[ZRA,ZRB,ZRC,ZRD] are such that YD+ZN=I. The left coprime case is treated in a similar way.

```
[m,n]=size(D);
H=eye(n)+(D'*D);L=eye(m)+(D*D');
Hinv=inv(H);Linv=inv(L);
Ahulp1=A–(B*Hinv*D'*C);
Ahulp2=(A–(B*D'*Linv*C))';

[X,XPERR]=aresolv(Ahulp1,C'*Linv*C,B*Hinv*B','Schur');
[Y,YPERR]=aresolv(Ahulp2,B*Hinv*B',C'*Linv*C,'Schur');

F=Hinv*((D'*C)+(B'*X)); AC=A–(B*F);
K=((B*D')+(Y*C'))*Linv; AO=A–(K*C);

Hr=sqrtl(H);Hrinv=inv(Hr);
Lr=sqrtl(L);Lrinv=inv(Lr);

DRA=AC;DRB=–B*Hrinv;DRC=F;DRD=Hrinv;
NRA=AC;NRB=B*Hrinv;NRC=C–(D*F);NRD=D*Hrinv;
YRA=AO;YRB=B–(K*D);YRC=Hr*F;YRD=Hr;
```

```
ZRA=AO;ZRB=K;ZRC=Hr*F;ZRD=zeros(n,m);

DLA=AO;DLB=K;DLC=-Lrinv*C;DLD=Lrinv;
NLA=AO;NLB=B-(K*D);NLC=Lrinv*C;NLD=Lrinv*D;
YLA=AC;YLB=K*Lr;YLC=C-(D*F);YLD=Lr;
ZLA=AC;ZLB=K*Lr;ZLC=F;ZLD=zeros(n,m);
```

```
function    [DRA,DRB,DRC,DRD,NRA,NRB,NRC,NRD,..
              DLA,DLB,DLC,DLD,NLA,NLB,NLC,NLD]=norcor(A,B,C,D)
```

Given a minimal realization [A,B,C,D] of a transfermatrix P(s), norcor computes realizations of a normalized right coprime factorization of P, ($P=ND^{-1}$ with D=[DRA,DRB,DRC,DRD] and N=[NRA,NRB,NRC,NRD]) and a normalized left coprime factorization of P ($P=D^{-1}N$ with D=[DLA,DLB,DLC,DLD] and N=[NLA,NLB,NLC,NLD]).

```
[m,n]=size(D);
H=eye(n)+(D'*D);L=eye(m)+(D*D');
Hinv=inv(H);Linv=inv(L);
Ahulp1=A-(B*Hinv*D'*C);
Ahulp2=(A-(B*D'*Linv*C))';

[X,XPERR]=aresolv(Ahulp1,C'*Linv*C,B*Hinv*B','Schur');
[Y,YPERR]=aresolv(Ahulp2,B*Hinv*B',C'*Linv*C,'Schur');

F=Hinv*((D'*C)+(B'*X));  AC=A-(B*F);
K=((B*D')+(Y*C'))*Linv;  AO=A-(K*C);

Hr=sqrtl(H);Hrinv=inv(Hr);
Lr=sqrtl(L);Lrinv=inv(Lr);

DRA=AC;DRB=-B*Hrinv;DRC=F;DRD=Hrinv;
NRA=AC;NRB=B*Hrinv;NRC=C-(D*F);NRD=D*Hrinv;

DLA=AO;DLB=K;DLC=-Lrinv*C;DLD=Lrinv;
NLA=AO;NLB=B-(K*D);NLC=Lrinv*C;NLD=Lrinv*D;
```

speccopr

Purpose :
Normalized Bezout factorization of a plant P, and calculation of the matrix V, associated with P. speccopr is a special purpose algorithm, to be used in the functions robstab and robstgl.

Synopsis :
[DRA,DRB,DRC,DRD,NRA,NRB,NRC,NRD,..
YLA,YLB,YLC,YLD,ZLA,ZLB,ZLC,ZLD,..
VA,VB,VC,VD] = speccopr(A,B,C,D)

Description :
Given a minimal realization [A,B,C,D] of a transfermatrix P(s), speccopr computes state–space realizations of a normalized right–Bezout factorization (D_0,N_0) of P and of matrices $\tilde{Y}_0$ and $\tilde{Z}_0$ such that (B.1) holds. (So speccopr delivers only a part of the output of ncoprfac.) Also a minimal state–space realization of the matrix V, associated with P as defined in (4.57), is given. These state–space realizations are returned as follows:

$D_0(s)$ = [DRA,DRB,DRC,DRD]; $\tilde{Y}_0(s)$ = [YLA,YLB,YLC,YLD];
$N_0(s)$ = [NRA,NRB,NRC,NRD]; $\tilde{Z}_0(s)$ = [ZLA,ZLB,ZLC,ZLD];
V(s) = [VA,VB,VC,VD].

Algorithm :
speccopr uses the construction method of theorem 4.1.1. for the state–space realizations of the matrices $D_0,N_0,\tilde{Y}_0$ and $\tilde{Z}_0$, and the method of theorem 5.2.1. for the state–space realization of V (formula (5.4)).

Nested functions :
sqrtl; see appendix E.

See also :
ncoprfac (see appendix B).

Program :
See next page.

```
function    [DRA,DRB,DRC,DRD,NRA,NRB,NRC,NRD,..
              YLA,YLB,YLC,YLD,ZLA,ZLB,ZLC,ZLD,..
              VA,VB,VC,VD]=speccopr(A,B,C,D)
```

Given a minimal realization [A,B,C,D] of a transfermatrix P, there are an n.c.r.f. (D,N) of P and an n.l.c.f. $(\bar{D},\bar{N})$ of P and stable matrices Y, Z and $\bar{Y}$, $\bar{Z}$, such that

$$\begin{bmatrix} -Z & Y \\ \bar{D} & \bar{N} \end{bmatrix} \begin{bmatrix} -N & \bar{Y} \\ D & \bar{Z} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}.$$

speccopr computes state space realizations of D=[DRA,DRB,DRC,DRD], N=[NRA,NRB,NRC,NRD], $\bar{Y}$=[YLA,YLB,YLC,YLD] and $\bar{Z}$=[ZLA,ZLB,ZLC,ZLD]. Moreover, a state space realization of the matrix $V = D^*\bar{Z} - N^*\bar{Y}$ (V=[VA,VB,VC,VD]) is computed.

```
[m,n]=size(D);[p,q]=size(A);
H=eye(n)+(D'*D);L=eye(m)+(D*D');
Hinv=inv(H);Linv=inv(L);
Ahulp1=A-(B*Hinv*D'*C);
Ahulp2=(A-(B*D'*Linv*C))';

[X,XPERR]=aresolv(Ahulp1,C'*Linv*C,B*Hinv*B','Schur');
[Y,YPERR]=aresolv(Ahulp2,B*Hinv*B',C'*Linv*C,'Schur');

F=Hinv*((D'*C)+(B'*X)); AC=A-(B*F);
K=((B*D')+(Y*C'))*Linv; AO=A-(K*C);

Hr=sqrtl(H);Hrinv=inv(Hr);
Lr=sqrtl(L);Lrinv=inv(Lr);

DRA=AC;DRB=-B*Hrinv;DRC=F;DRD=Hrinv;
NRA=AC;NRB=B*Hrinv;NRC=C-(D*F);NRD=D*Hrinv;
YLA=AC;YLB=K*Lr;YLC=C-(D*F);YLD=Lr;
ZLA=AC;ZLB=K*Lr;ZLC=F;ZLD=zeros(n,m);

VA=-AC';VB=(eye(p)+(X*Y))*C'*Lrinv;
VC=Hrinv*B';VD=-Hrinv*D'*Lr;
```

APPENDIX C

This appendix contains function–descriptions of the MATLAB–functions hankel, hanneh and robstab.

hankel

Purpose :
Computation of the norm of a Hankel–operator.

Synopsis :
norm = hankel(A,B,C,D)

Description :
Given a (not necessary minimal) realization [A,B,C,D] of a real–rational matrix V(s), hankel computes the norm of the Hankel–operator with symbol V.

Algorithm :
hankel uses the algorithm for the computation of the norm of a Hankel–operator as given in section 5.3.

See also :
hanneh (see appendix C); this function not only computes the norm of the Hankel–operator with symbol V, but also solves the related Nehari–problem.

Program :

```
function norm=hankel(A,B,C,D)

% Given a realization [A,B,C,D] of a real rational matrix V(s) hankel computes the norm of the
% hankel-operator with symbol V.

[A1,B1,C1,D1,A4,B2,C2,D2,m]=stabproj(A,B,C,D);
BBT2=B2*(B2');CTC2=(C2')*C2;
LC=lyap(A4,A4',-BBT2);LO=lyap(A4',A4,-CTC2);
LCLO=LC*LO;eigenval=eig(LCLO);a=max(eigenval);norm=sqrt(a);
```

hanneh

Purpose :
Computation of the norm of a Hankel–operator and of a sub–optimal solution to the related Nehari–problem.

Synopsis :
[XA,XB,XC,XD,hannorm] = hanneh(A,B,C,D,tol)

Description :
Given a minimal realization [A,B,C,D] of a real–rational matrix V(s), hanneh computes:

i) hannorm; the norm of the Hankel–operator with symbol V.

ii) a realization [XA,XB,XC,XD] of a matrix X(s) $\in$ M($\mathbb{RH}_\infty$) which is a sub–optimal solution to the related Nehari–problem, within the desired accuracy–level, given by 'tol'. I.e. a matrix X(s) $\in$ M($\mathbb{RH}_\infty$) is calculated such that:

$$\| V - X \| < \| \Gamma_V \| (1 + tol) \qquad (C.1)$$

Algorithm :
hanneh uses the second algorithm stated in section 5.4. for the solution of the Nehari–problem. This algorithm is based upon the method for solving the Nehari–problem described in the same paragraph (this is the same method, Francis gives in [4, ch.5]). In this method the norm of the Hankel–operator is calculated automatically, actually it is an intermediate result.

See also :
hannehgl; this function gives for an anti–stable matrix V(s) an exact solution to the Nehari–problem (see appendix D).

Program :

```
function[XA,XB,XC,XD,hannorm]=hanneh(A,B,C,D,tol)

Given a minimal realization [A,B,C,D] of a real rational matrix V(s) hanneh computes:
i)  hannorm: the norm of the hankel-operator with symbol V
ii) a realization [XA,XB,XC,XD] of a matrix X(s) such that the Hinf-norm of V-X smaller is than
    hannorm*(1+tol)

[A1,B1,C1,D1,A4,B2,C2,D2,m]=stabproj(A,B,C,D);
BBT2=B2*(B2');CTC2=(C2')*C2;
```

```
LC=lyap(A4,A4',–BBT2);
LO=lyap(A4',A4,–CTC2);
LCLO=LC*LO;eigenval=eig(LCLO);a=max(eigenval);

a1=1/(a*(1+tol)*(1+tol));
[p,q]=size(LCLO);
Ninv=eye(p)–(a1*LCLO);N=inv(Ninv);

[r,s]=size(A);[k,l]=size(C);
hulp1=a1*N*LO*B2;hulp2=(–N')*BBT2;

XA=[A zeros(r,(2*p));zeros(p,r) A4 hulp2;zeros(p,(r+p)) (–A4'–(hulp1*B2'))];
XB=[B;N'*B2;hulp1];
XC=[C –C2 zeros(k,p)];
XD=D;

hannorm=sqrt(a);
```

robstab

Purpose :
Computation of the maximal stability radius belonging to a plant P and a sub–optimally robust controller C.

Synopsis :
[CA,CB,CC,CD,bound] = robstab(PA,PB,PC,PD,tol)

Description :
Given a minimal realization [PA,PB,PC,PD] of a plant P, robstab computes the maximal stability radius belonging to P (this value is stored in the variable bound). Also a state–space realization [CA,CB,CC,CD] of a sub–optimally robust controller C is calculated, which realizes the value of bound within the desired accuracy–level, given by tol. So not only P is stabilized by C, but also the systems in the ball around P, with radius (bound)*(1–tol), i.e. the whole set

$$\{ P_\lambda \in M(\mathbb{R}(s)) \mid \delta(P,P_\lambda) < (\text{bound}) * (1 - \text{tol}) \} \qquad (C.2)$$

is stabilized by C.
When the order of the original plant P is n, the order of the calculated compensator is 10n.

Algorithm :

robstab is the computer–implementation of the algorithm for the solution of the problem of (sub)–optimally robust control as given in section 5.5. This means that the internal Nehari–problem is solved by the method of section 5.4., which is implemented in the function hanneh.

See also :

robstgl, rbstglfar (see appendix D); these are other algorithms for the solution of the problem of optimally robust control.

Nested functions :

speccopr; see appendix B. hanneh; see appendix C.
tfadd, tfinv, tfmult; see appendix E.

Program :

```
function[CA,CB,CC,CD,bound]=robstab(PA,PB,PC,PD,tol)
```

Given plant P=[PA,PB,PC,PD], robstab calculates a compensator C=[CA,CB,CC,CD] that stabilizes P, and for each plant Q, such that d(P,Q) < bound(1-tol), C is also a stabilizing compensator. The variable bound represents the maximal stability radius belonging to P, i.e. the radius of the largest ball around P that can be stabilized by one compensator. In this way C is a sub-optimal solution to the problem of optimally robust control.*
In the computation a Nehari-problem has to be solved. tol determines how accurate this problem has to be solved (tolerance in accuracy).

```
[DRA,DRB,DRC,DRD,NRA,NRB,NRC,NRD,YLA,YLB,..
  YLC,YLD,ZLA,ZLB,ZLC,ZLD,VA,VB,VC,VD]=speccopr(PA,PB,PC,PD);
[RA,RB,RC,RD,Ahan]=hanneh(VA,VB,VC,VD,tol);
wg=sqrt(1+(Ahan*Ahan));  bound=1/wg;

[H1,H2,H3,H4]=tfmult(DRA,DRB,DRC,DRD,RA,RB,–RC,–RD);
[H5,H6,H7,H8]=tfadd(ZLA,ZLB,ZLC,ZLD,H1,H2,H3,H4);

[H1,H2,H3,H4]=tfmult(NRA,NRB,NRC,NRD,RA,RB,RC,RD);
[H9,H10,H11,H12]=tfadd(YLA,YLB,YLC,YLD,H1,H2,H3,H4);
[H1,H2,H3,H4]=tfinv(H9,H10,H11,H12);

[CA,CB,CC,CD]=tfmult(H5,H6,H7,H8,H1,H2,H3,H4);
```

APPENDIX D

This appendix contains function–descriptions of the MATLAB–functions hannehgl, robstgl and rbstglfar.

hannehgl

Purpose :
Computation of the norm of a Hankel–operator, and of an exact solution to the related Nehari–problem.

Synopsis :
[XA,XB,XC,XD,hannorm] = hannehgl(A,B,C,D)

Description :
Given a minimal state–space realization [A,B,C,D] of a real–rational *anti–stable* matrix V(s), hannehgl computes:

i) hannorm, the norm of the Hankel–operator with symbol V.

ii) a realization [XA,XB,XC,XD] of a matrix X(s) $\in$ M($\mathbb{RH}_\infty$), which is an exact solution to the Nehari–problem for the matrix V(s). I.e. the matrix X(s) $\in$ M($\mathbb{RH}_\infty$) has the property that

$$\| V - X \| = \| \Gamma_V \|. \qquad (D.1)$$

When the order of the matrix V is n, the state–space realization of the matrix X is of the order n–1.

Algorithm :
hannehgl uses the method of Glover as described in section 6.2. to solve the Nehari–problem exactly. The realization of X is constructed without the use of balanced realizations, but with the method given in [14] and implemented in the MATLAB–function ohkapp. This last function is a standard function in the Robust Control Toolbox.

See also :
hanneh (see appendix B); this algorithm computes a sub–optimal solution to the Nehari–problem.

Program :
See next page.

function[XA,XB,XC,XD,hannorm]=hannehgl(A,B,C,D)

Given a minimal realization [A,B,C,D] of a real anti-stable matrix V(s), hannehgl computes:

i) hannorm: the norm of the Hankel-operator with symbol V

ii) a realization [XA,XB,XC,XD] of a stable matrix X(s), such that the Hinf-norm of V-X is equal to hannorm, and the order of X is smaller than the order of V

```
[AX,BX,CX,DX,AY,BY,CY,DY,aug]=ohkapp(-A,B,-C,D,1,0);
XA=-AY;XB=BY;XC=-CY;XD=DY;
hannorm=aug(1,1);
```

robstgl

Purpose :
Computation of the maximal stability radius belonging to a plant P and an optimally robust controller C.

Synopsis :
[CA,CB,CC,CD,bound] = robstgl(PA,PB,PC,PD)

Description :
Given a minimal realization [PA,PB,PC,PD] of a plant P, robstgl computes the maximal stability radius belonging to P (stored in the variable bound), and a state–space realization [CA,CB,CC,CD] of an optimally robust controller C that realizes this bound. This controller C not only stabilizes P, but also systems in the ball around P, with radius 'bound', i.e. the set

$$\{ P_\lambda \in M(\mathbb{R}(s)) \mid \delta(P,P_\lambda) < \text{bound} \}. \qquad (D.2)$$

When the order of the plant is n, the order of the calculated compensator C is 6n–2.

Algorithm :
robstgl uses the method of Glover for the solution of the problem of optimally robust control, as explained in section 6.2. So the internal Nehari–problem is exactly solved by the method of Glover, which is implemented in the function hannehgl.

See also :
robstab (see appendix C), rbstglfar (see appendix D); these are other algorithms for the solution of the problem of optimally robust control.

Nested functions :
speccopr; see appendix B. hannehgl; see appendix D.
tfadd, tfinv, tfmult; see appendix E.

Program :

```
function[CA,CB,CC,CD,bound]=robstgl(PA,PB,PC,PD)
```

Given plant P=[PA,PB,PC,PD], robstgl calculates a compensator C=[CA,CB,CC,CD] that stabilizes P, and for each plant Q, such that d(P,Q)<bound, C is also a stabilizing compensator. Moreover, C is chosen in such a way that bound is maximal. I.e. for each compensator C' that stabilizes P, the radius of the ball around P of plants also stabilized by C', is not greater than bound.

```
[DRA,DRB,DRC,DRD,NRA,NRB,NRC,NRD,YLA,YLB,..
 YLC,YLD,ZLA,ZLB,ZLC,ZLD,VA,VB,VC,VD]=speccopr(PA,PB,PC,PD);
[RA,RB,RC,RD,Ahan]=hannehgl(VA,VB,VC,VD);
wg=sqrt(1+(Ahan*Ahan)); bound=1/wg;

[H1,H2,H3,H4]=tfmult(DRA,DRB,DRC,DRD,RA,RB,-RC,-RD);
[H5,H6,H7,H8]=tfadd(ZLA,ZLB,ZLC,ZLD,H1,H2,H3,H4);
[H1,H2,H3,H4]=tfmult(NRA,NRB,NRC,NRD,RA,RB,RC,RD);
[H9,H10,H11,H12]=tfadd(YLA,YLB,YLC,YLD,H1,H2,H3,H4);
[H1,H2,H3,H4]=tfinv(H9,H10,H11,H12);

[CA,CB,CC,CD]=tfmult(H5,H6,H7,H8,H1,H2,H3,H4);
```

rbstglfar

Purpose :
Computation of the maximal stability radius belonging to a plant P and a sub–optimally robust controller C.

Synopsis :
[CA,CB,CC,CD,bound] = rbstglfar(PA,PB,PC,PD,tol)

Description :
Given a minimal realization [PA,PB,PC,PD] of a plant P, rbstglfar computes the maximal stability radius belonging to P (which is returned in the variable bound) and a state–space

realization [CA,CB,CC,CD] of a sub–optimally robust controller C, which realizes this bound within an accuracy–level of tol (tol is one of the input variables). This means that C stabilizes all the systems in the ball around P with radius (bound)*(1–tol), i.e. C stabilizes the set

$$\{ P_\lambda \in M(\mathbb{R}(s)) \mid \delta(P,P_\lambda) < (\text{bound}) * (1-\text{tol}) \}. \qquad (D.3)$$

The order of this compensator C is equal to the order of the original plant P.

Algorithm :
To compute the maximal stability radius belonging to P, rbstglfar uses the method of theorem 6.3.2. So the value of bound is calculated with formula (6.22). The state–space realization of C is constructed by the method of Glover and McFarlane as given in theorem 6.3.3. Formula (6.35) is used for the computation of the realization of C.

See also :
robstab (see appendix C), robstgl (see appendix D); these are other algorithms for the solution of the problem of optimally robust control.

Program :

```
function[CA,CB,CC,CD,bound]=rbstglfar(PA,PB,PC,PD,tol)
```

Given a minimal realization [PA,PB,PC,PD] of a plant P, rbstglfar calculates the maximal stability radius 'bound' and a realization [CA,CB,CC,CD] of a compensator C, which realizes this bound within an accuracy-level of tol.

```
[m,n]=size(PD);[r,k]=size(PA);
H=eye(n)+(PD'*PD);L=eye(m)+(PD*PD');
Hinv=inv(H);Linv=inv(L);
Ahulp1=PA–(PB*Hinv*PD'*PC);
Ahulp2=(PA–(PB*PD'*Linv*PC))';

[X,XPERR]=aresolv(Ahulp1,PC'*Linv*PC,PB*Hinv*PB','Schur');
[Y,YPERR]=aresolv(Ahulp2,PB*Hinv*PB',PC'*Linv*PC,'Schur');

l=max(eig(Y*X));bound=1/sqrt(1+l);
gamma=(1/bound)*(1+tol);g2=gamma*gamma;

F=Hinv*((PD'*PC)+(PB'*X));
```

```
AC=PA-(PB*F);
W1=((1-g2)*eye(r))+(X*Y);

CB=g2*(inv(W1'))*Y*PC';
CA=AC+CB*(PC-(PD*F));
CC=PB'*X;
CD=-PD';
```

APPENDIX E

This appendix contains function–descriptions of the MATLAB–functions tfadd, tfinv, tfmult and sqrtl.

tfadd

Purpose :
Addition of two transfermatrices in state–space form.

Synopsis :
[P,Q,R,S] = tfadd(A1,B1,C1,D1,A2,B2,C2,D2)

Description :
Given state–space realizations of two transfermatrices of the same size, tfadd computes a state–space realization of the sum of these transfermatrices, i.e.

[P,Q,R,S] = [A1,B1,C1,D1] + [A2,B2,C2,D2].

Algorithm :
tfadd uses the state–space formula of page ix for the addition of two transfermatrices, to construct a state–space realization of the sum.

Program :

```
function[P,Q,R,S]=tfadd(A1,B1,C1,D1,A2,B2,C2,D2)

% Given state space realizations of two transfermatrices of the same size, tfadd calculates a state space
% realization of the sum of these transfermatrices, i.e.
%                    [P,Q,R,S]=[A1,B1,C1,D1]+[A2,B2,C2,D2].

[na1,ma1]=size(A1); [na2,ma2]=size(A2);
P=[A1 zeros(na1,ma2);zeros(na2,ma1) A2];
Q=[B1;B2];
R=[C1 C2];
S=D1+D2;
```

tfinv

Purpose :
Inversion of a transfermatrix in state–space form.

Synopsis :
[P,Q,R,S] = tfinv(A,B,C,D)

Description :
Given a state–space realization of a square invertible transfermatrix, tfinv computes a state–space realization of the inverse of this transfermatrix, i.e.

$$[P,Q,R,S] = [A,B,C,D]^{-1}.$$

Algorithm :
tfinv uses the state–space formula of page ix for the inversion of a transfermatrix, to construct a state–space realization of the inverse.

Program :

```
function[P,Q,R,S]=tfinv(A,B,C,D)

Given a state space realization of a square transfermatrix (with D invertible), tfinv calculates a state space
realization of the inverse of this transfermatrix. So
                    [P,Q,R,S]=inv([A,B,C,D]).

hulp=inv(D);
P=A–(B*hulp*C);
Q=B*hulp;
R=–hulp*C;
S=hulp;
```

tfmult

Purpose :
Multiplication of two transfermatrices in state–space form.

Synopsis :
[P,Q,R,S] = tfmult(A1,B1,C1,D1,A2,B2,C2,D2)

Description :
Given state–space realizations of two transfermatrices of compatible size (the number of columns of the first one equals the number of rows of the second one), tfmult calculates a state–space realization of the product of these transfermatrices, i.e.

[P,Q,R,S] = [A1,B1,C1,D1] * [A2,B2,C2,D2].

Algorithm :
tfmult uses the state–space formula of page ix for the multiplication of two transfermatrices, to construct a state–space realization of the product.

Program :

```
function[P,Q,R,S]=tfmult(A1,B1,C1,D1,A2,B2,C2,D2)

%Given state space realizations of two transfermatrices (number of columns of the first equals the number of
%rows of the second), tfmult computes a state space realization of the product of the two transfermatrices,
%i.e.
%                [P,Q,R,S]=[A1,B1,C1,D1]*[A2,B2,C2,D2].

[na1,ma1]=size(A1); [na2,ma2]=size(A2);
P=[A1 (B1*C2);zeros(na2,ma1) A2];
Q=[(B1*D2);B2];
R=[C1 (D1*C2)];
S=D1*D2;
```

sqrtl

Purpose :
Computation of the square root of a positive definite matrix.

Synopsis :
root = sqrtl(A)

Description :

Given a positive definite matrix A, sqrtl computes the matrix $A^{1/2}$ with help of the singular value decomposition. So when A is given by

$$A = U\ S\ V^T,$$

with S diagonal, then sqrtl(A) is

$$\text{sqrtl}(A) = U\ S^{1/2}\ V^T.$$

Program :

```
function wortel=sqrtl(A)

sqrtl computes the square root of a PD matrix  with help of the SV-Decomposition.

[U,S,V]=svd(A);
D=S.^(1/2);
wortel=U*D*V';
```

APPENDIX F

This appendix contains solutions to the problem of (sub)–optimally robust stabilization for a plant with transferfunction $P(s) = \frac{1}{s-1}$, calculated in five different ways:

i) with the MATLAB–function robstab; tolerance level tol = 10^{-5};
ii) with the MATLAB–function robstab; tolerance level tol = 10^{-8};
iii) with the MATLAB–function robstgl;
iv) with the MATLAB–function rbstglfar; tolerance level tol = 10^{-5};
v) with the MATLAB–function rbstglfar; tolerance level tol = 10^{-8}.

In each case the calculated maximal stability radius is given and a state–space realization of a compensator that realizes this bound within (for the cases i), ii), iv) and v)) the desired accuracy level.

i) MATLAB–function robstab, tolerance level tol = 10^{-5}

PA = 1

PB = 1

PC = 1

PD = 0

bound = 0.3827

tol = 1.0000e–005

CA = 1.0e+006 *

Columns 1 through 7

−0.0000	0	0	0	0	−0.0000	−0.0000
0	−0.0000	0.0000	−0.0000	0	0	0
0	0	0.0000	0	0	−0.0000	−0.0000
0	0	0	0.0000	−2.3314	−0.3414	−0.3414
0	0	0	0	−0.1414	−0.0207	−0.0207
0	0	0	0	0	−0.0000	−0.0000
0	0	0	0	0	0	−0.0000
0	0	0	0	0	−0.0000	−0.0000
0	0	0	0	0	−0.3414	−0.3414
0	0	0	0	0	−0.0207	−0.0207

Columns 8 through 10

0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0.0000	−0.0000	0
0.0000	0	0
0	0.0000	−2.3314
0	0	−0.1414

CB = 1.0e+005 *

0.0000
0
0.0001
3.4143
0.2071
0.0000
0
0.0001
3.4143
0.2071

CC =

Columns 1 through 7

2.4142	2.4142	−1.0000	1.0000	0	0	0

Columns 8 through 10

0	0	0

CD = 0

ii) MATLAB–function robstab, tolerance level tol = 10^{-8}

PA = 1

PB = 1

PC = 1

PD = 0

bound = 0.3827

tol = 1.0000e–008

CA = 1.0e+009 *

Columns 1 through 7

–0.0000	0	0	0	0	–0.0000	–0.0000
0	–0.0000	0.0000	–0.0000	0	0	0
0	0	0.0000	0	0	–0.0000	–0.0000
0	0	0	0.0000	–2.3314	–0.3414	–0.3414
0	0	0	0	–0.1414	–0.0207	–0.0207
0	0	0	0	0	–0.0000	–0.0000
0	0	0	0	0	0	–0.0000
0	0	0	0	0	–0.0000	–0.0000
0	0	0	0	0	–0.3414	–0.3414
0	0	0	0	0	–0.0207	–0.0207

Columns 8 through 10

0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0.0000	–0.0000	0
0.0000	0	0
0	0.0000	–2.3314
0	0	–0.1414

CB = 1.0e+008 *

0.0000
0
0.0000
3.4142
0.2071
0.0000
0
0.0000
3.4142
0.2071

CC =

Columns 1 through 7

2.4142	2.4142	−1.0000	1.0000	0	0	0

Columns 8 through 10

0	0	0

CD = 0

iii) MATLAB–function robstgl

PA = 1

PB = 1

PC = 1

PD = 0

bound = 0.3827

CA =

−1.4142	0	−2.4142	−2.4142
0	−1.4142	2.4142	2.4142
0	0	−3.8284	−2.4142
0	0	2.4142	1.0000

CB =

2.4142
−2.4142
2.4142
−2.4142

CC =

2.4142	2.4142	−2.4142	−2.4142

CD = 2.4142

iv) MATLAB–function rbstglfar, tolerance level tol = 10^{-5}

PA = 1

PB = 1

PC = 1

PD = 0

bound = 0.3827

tol = 1.0000e–005

CA = –1.2071e+005

CB = –1.2071e+005

CC = 2.4142

CD = 0

v) MATLAB–function rbstglfar, tolerance level tol = 10^{-8}

PA = 1

PB = 1

PC = 1

PD = 0

bound = 0.3827

tol = 1.0000e–008

CA = –1.2071e+008

CB = –1.2071e+008

CC = 2.4142

CD = 0

REFERENCES

[1] N.A. Bruinsma and M. Steinbuch, *A fast algorithm to compute the H_∞-norm of a transferfunction.* Philips Research Laboratories Eindhoven. Eindhoven, 1989.

[2] R. Chiang and M. Safonov, *Robust-Control Toolbox, User's Guide.* The MathWorks, Inc. South Natick, 1988.

[3] H.O. Cordes and J.P. Labrousse, The invariance of the index in the metric space of closed operators. *Journal of Mathematics and Mechanics*, vol. 12, pp. 693–719, 1963.

[4] B.A. Francis, *A Course in H_∞ Control Theory.* Berlin–Heidelberg–New York, Springer Verlag, 1987. Lecture Notes in Control and Information Sciences, vol. 88.

[5] T.T. Georgiou, On the computation of the gap metric. *Systems and Control Letters*, vol. 11, pp. 253–257, 1988.

[6] T.T. Georgiou and M.C. Smith, *Optimal Robustness in the Gap Metric.* Preprint, 1989.

[7] K. Glover, All optimal Hankel–norm approximations of linear multivariable systems and their L_∞–error bounds. *Int. J. Control*, vol. 39, pp. 1115–1193, 1984.

[8] K. Glover and D. McFarlane, Robust stabilization of normalized coprime factor plant descriptions with H_∞–bounded uncertainty. *IEEE Trans. on Automat. Control*, vol. 34, pp. 821–830, 1989.

[9] T. Kato, *Perturbation Theory for Linear Operators.* Berlin–New York, Springer Verlag, 1966.

[10] M.A. Krasnosel'skii, G.M. Vainikko and P.P. Zabreiko, *Approximate Solution to Operator Equations.* Groningen, Wolters–Noordhoff, 1972.

[11] E. Kreyszig, *Introductory Functional Analysis with Applications.* New York, Wiley, 1978.

[12] D. Meyer and G. Franklin, A connection between normalized coprime factorizations and linear quadratic regulator theory. *IEEE Trans. on Automat. Control*, vol. AC–32, pp. 227–228, 1987.

[13] Z. Nehari, On bounded bilinear forms. *Annals of Mathematics*, vol. 65, pp. 153–162, 1957.

[14] M. Safonov, R. Chiang and D. Limebeer, Hankel model reduction without balancing – A descriptor approach. *Proc. of the 26th Conf. on Decision and Control*, Los Angeles, 1987.

[15] M. Vidyasagar, *Control System Synthesis: A Factorization Approach.* Cambridge, MA, M.I.T. Press, 1985.

[16] M. Vidyasagar, Normalized coprime factorizations for nonstrictly proper systems. *IEEE Trans. on Automat. Control*, vol. 33, pp. 300–301, 1988.

[17] M. Vidyasagar and H. Kimura, Robust controllers for uncertain linear multivariable systems. *Automatica*, vol. 22, pp. 85–94, 1986.

[18] S.Q. Zhu, Graph topology and gap topology for unstable plants. *IEEE Trans. on Automat. Control*, vol. 34, pp. 848–855, 1989.

[19] S.Q. Zhu, *Robustness of Feedback Stabilization: A topological approach.* Ph. D. dissertation, Eindhoven University of Technology, Eindhoven, 1989.

[20] S.Q. Zhu, M.L.J. Hautus and C. Praagman, Sufficient conditions for robust BIBO stabilization: Given by the gap metric. *Systems and Control Letters*, vol. 11, pp. 53–59, 1988.