SYStems

Version 0.1

J. William Helton [1]
Math Dept., UCSD

Michael L. Walker[1]
General Atomic Corp.
La Jolla, California 92093

This document provides an overview of the NCAlgebra tools which are provided for studying engineering systems. At present what is well developed are files for doing H^∞ control and with a slight modification of language differential games. This includes expressions and rules useful when using the Hamiltonian point of view. We hope others will contribute packages for doing basic systems theory or for doing research on more specialized subjects. Many topics in linear systems could EASILY be programmed into this setting.

[1]This work was partially supported by the AFOSR and the NSF.

Contents

Chapter 1

PREFACE

To what extent can we make our A, B, C, D linear systems and operator calculations easier with computer assistance? The standard symbol manipulators Mathematica, Maple, and Macsyma do not do noncommutative algebra intelligently, so it is hard to find out. Our group wrote a package NCAlgebra which runs under Mathematica and which is a reasonable beginning.

We see this package as a competitor to the yellow pad. Once you get used to it this might be considerably more effective for hand calculations of modest size. Like Mathematica the emphasis is on interaction with the program and flexibility.

I now prefer it to a yellow pad for many types of calculation and can recommend it to anyone who is good with Mathematica or alternatively is familiar with computers to the extent of doing a substantial amount of word processing. Putting a graduate student with computer talent on NCAlgebra also might work well. Our experience at UCSD is that students find it easy to learn. Indeed anyone can play with our package effortlessly, but one would have to be cautious about committing to a long term research project using NCAlgebra without being able to make additional commands on his own.

SYStems is based on NCAlgebra, a collection of "functions" for Mathematica designed to facilitate manipulation and reduction of noncommutative algebraic expressions. Specifically, it allows computer calculation in an algebra with involution. Such computations are common in many areas but our background is operator theory and engineering systems so we are aiming at research in these areas rather than at the complete treatments of the basics in these subjects.

SYSTEMS is a package which runs under NCAlgebra. It is in a primitive form with its main contents being a file for doing H^∞ control for linear and certain nonlinear systems. This originated with a paper (Aug91; by Ball, Helton and Walker [BHW]) which solves a collection of nonlinear H^∞ control problems. The formulas in that paper are executable inside NCAlgebra which greatly facilitates further research along these lines.

Chapter 2

HOW TO RUN THE SYSTEMS PACKAGE

The beginner should first print out the file NCDOCUMENT.doc which explains how to use NCAlgebra. A little practice with this would help.

To run SYSTEMS, enter

```
math
```

at the UNIX prompt to execute Mathematica (or whatever command is appropriate for your non-UNIX system), followed by

```
<<SYStems.m
```

at the Mathematica

```
In[1]:=
```

prompt.

That is all there is to it. Here SYStems.m loads NCAlgbra.m, one of the SYSTEMS DEFINITION files, plus two of the SYSTEMS UTILITY files described below.

These files are currently limited primarily to the study of H^∞ control of a certain class of nonlinear systems. There are additional files listed within comment delimiters in the SYSTEMS file which can be loaded to study slightly (but not much) different problems. This includes the option of studying so-called WIA systems in place of IA systems (see SYSHinf1 for definitions) or specializing to certain linear problems. A summary of files is given below. See the SYSTEMS file for further description.

The first time you run SYStem.s you should probably test to see that everything is in order. To do this you type:

```
math
<<SYStems.m
<<SYSTEST.m
```

while ignoreing lots of stuff which scrolls by on the screen. What counts comes at the end. You should see a list which says Test1 is True, Test2 is True, etc.

SYSTEMS DEFINITIONS

File	Description
SYSDef1portlin.m	
SYSDefIA.m	Loads the system definitions and hamiltonian for an Input-Affine system (see EXAMPLE 5.2)
SYSDefWIA.m	Definition of a WIA (W-Input-Affine) system SYSTEMS file could be edited to replace loading of SYSDef with loading of this file.
SYSDefWAccess.m	
SYSDef2&2.m	Definition of two 2 ports and the coupling laws for coupling them as well as hamiltonians.

UTILITY FILES

File	Description
SYSSpecialize.m	Rules for converting systems from nonlinear to linear, notation changes , etc.
SYSLinearize.m	Linearize system around 0 or around the diagonal x=z. (We do not support this one)
SYSHinfFormulas.m	Formulas used in study of H-infinity control.

FILES CORRESPONDING TO RESEARCH ARTICLES

File	Description
SYSHinfTAC.m	An executable paper on nonlinear H-infinity control together with rules to allow conversion to linear case. Primarily for study of IA systems.

Chapter 3

HOW TO CONTRIBUTE

Send files.m preferably as packages to

`ncalg@osiris.ucsd.edu`

Also we would like a file.doc which tells how to use it. Also we would like a SYSfileTest.m which automatically tests your program. This is not essential. See Diff.NCTest for an example of the format for these files. Usually you can just modify a copy of this file.

Chapter 4

WHAT SYStems DOES

The tools in the SYSTEMS package are designed to assist in studying various systems using the following commonplace idea (see [BHW] for more detail):

Define a finite-gain dissipative system to be a system for which

$$\int_{t0}^{t1} |out|^2 dt \leq K \int_{t0}^{t1} |W|^2 dt$$

where K is a constant, and $x(t0) = 0$. (See Figure 1.) If $K = 1$ the system will be called dissipative. In circuit theory these would be called passive. This agrees with the notion of dissipative in [W],[HM] with respect to the supply rate $|W|^2 - |out|^2$. Our notation here is like that one puts in the computer.

Define a storage or energy function on the state space to be a nonnegative function e satisfying

$$\int_{t0}^{t1} (|out|^2 - |W|^2) dt \leq e(x(t0)) - e(x(t1))$$

and $e(0) = 0$. Hill-Moylan ([HM]) showed that a system is dissipative iff an energy (storage) function (possibly extended real valued) exists. Under controllability assumptions, there exists an energy function with finite values.

We find it convenient to say that a system of the above form is e-dissipative provided that the energy Hamiltonian H defined by

$$H = out^T\, out - W^T\, W + (p\, F[x, W] + F[x, W]^T\, p^T)/2$$

is nonpositive where $p = \nabla(e(x))$. That is, $0 \geq H$ for all W and all x in the set of states reachable from 0 by the system.

Theorem 1 *(see [W],[HM]) Let e be a given differentiable function. Then a system is e-dissipative if and only if e is a storage function for the system. In this case, the system is dissipative.*

Using this background, we may study dissipativeness of systems by examining the non-positiveness of Hamiltonians. Replacing the dual variables p by

$$p = \nabla_x\, e$$

This substitution converts H to

$$sHW = out^T\, out - W^T\, W + (\nabla_x\, e\, F[x, W] + F[x, W]^T\, \nabla_x^T\, e)/2$$

Often one finds that

$$sHWo := \max_W sHW(W, x)$$

(the Hamiltonian in state space variables which has been optimized in W) is well behaved and is the first max taken in many approaches to solving the problem of checking if $sHW \leq 0$ for all x. $sHWo$ can be computed concretely for systems (see e.g. files SYSDefIA.m and SYSDefWIA.m) by taking the gradient of sHW in W and setting it to 0 to find the critical point CritW. Substitute this back into sHW to get sHWo. In our language CritW can be computed by

```
ruCritW=Crit[H,W];      (Calculate critical value for H w.r.t. W)
HWo=Sub[H,ruCritW];     (Substitute critical W back in H)
sHWo=Sub[HWo,ruxz];     (Convert to state-space coordinates)
```

since p is independent of W. Some examples are given later to illustrate.

What we have done so far is explain the very simplest case. Whenever one has a Hamiltonian which is quadratic in a variable our symbol manipulator optimizes the variable automatically. In H^∞ control one gets Hamiltonians which are very complicated expressions in many variables (e.g. compensator parameters). One must take many maxima and minima in analysing these problems and we have found SYStems very effective at such computations.

Game theory is another area which produces Hamiltonians and calls for such computations.

Rather than try to list complicated classifications of Hamiltonian types that this addresses we proceed by presenting some examples. One is the classical bounded real lemma while the other derives the famous DGKF two Riccati equation formulas for nonlinear systems (which are affine linear in the input variables).

Also there is substantial systems capability in a completely different directions. NCAlgebra contains commands for manipulating block matricies. Since such computations are the substance of many results in systems theory it is already easy to do computations in these areas with NCAlgbra. A few formulas for 2×2 block matricies are stored: Schur complements, block LU decomposition, inverses, and Chain to scattering formalism conversions.

Chapter 5

SAMPLE APPLICATIONS

Each of the following examples uses one of the SYSDef files as input. That SYSDef file is reproduced here for your reference. Text between comment symbols (* and *) is non-executable. In this document we have modified the actual SYSDef files by a few deletions and by transcribing many of the comments to TeX. Everything else in the SYSDef file is loaded into your session when you type:

```
<<SYSDef####.m
```

where #### represents the particular SYSDef file chosen.

5.1 Bounded Real Lemma

This example derives the bounded real lemma for a linear system. This is a special case of this lemma for a more general system described in [BHW]. This example uses the file SYSDef1portlin.m as input. We next give a listing of that file (with a few non-executable passages transcribed to TeX.)

```
SYSDef1portlin.m
(*************************************************************************
The system definition is:

                   dx/dt = F[x,W];
                    out  = G1[x,W];


                     --------------
                     |            |
            W  ---->--| F     G1   |---->--- out
                     |____________|

                        Figure 1.

For linear systems this is:
*)
                   SNC[W,U,Y,DW,DU,DY];
```

```
                    SNC[A,x,B1,C1,G1];
                    SNC[D11];

                    F[x_,W_,U_]:=A**x+B1**W;
                    G1[x_,W_,U_]:=C1**x+D11**W;
                             out=G1[x,W,U];

(*
Energy Hamiltonian H is defined by
*)
             SNC[F,G1,G2,p]
   H = tp[out]**out - tp[W]**W + (p**F[x,W,U]+tp[F[x,W,U]]**tp[p])/2

                    SNC[e,XX]
              e[x_]:=tp[x]**XX**x
                        tp[XX]=XX
(*
Thus  p=Grad[e[x]] equals
*)

       p=2 tp[XX**x]

END OF FILE ##############################################END OF FILE
```

Now we prove the classical bounded real lemma using NCAlgebra.

Lemma 1 *(BOUNDED REAL) Assume* $(-K + D11^T D11)^{-1}$ *exists. Then the linear system above is finite-gain dissipative with gain* $\leq K$ *and energy function* $x^T XX\, x$ *if and only if* XX *is a positive semidefinite solution to the Riccati inequality* $0 \geq ricd(XX)$. *Here the Riccati operator is*

$$ricd = XX\, A + A^T XX + C1^T C1 + (XX\, B1 + C1^T D11)(-K + D11^T D11)^{-1}(-B1^T XX - D11^T C1)$$

Proof. See [AV].

```
PROOF OF BR LEMMA (BY COMPUTATION):

In[2]:= <<NCAlgebra

In[3]:= <<SYSDef1portlin.m---- loads the definitions above.

In[4]:= H

Out[4]= (tp[W] ** tp[D11] + tp[x] ** tp[C1]) ** (C1 ** x + D11 ** W) -
   tp[W] ** W + (2*(tp[W] ** tp[B1] + tp[x] ** tp[A]) ** XX ** x +
      2*tp[x] ** XX ** (A ** x + B1 ** W))/2

----- The strategy is to maximize H and see if this is negative----
```

```
In[5]:= Grad[H,W]      ------- differentiate to find the "worst" input W -----

Out[5]= 2*(tp[W] ** tp[D11] ** D11 + tp[x] ** XX ** B1 +
     tp[x] ** tp[C1] ** D11 - tp[W])

In[6]:= tp[%]/2        ------- Start solving for the critical W  ---------

Out[6]= -W + tp[B1] ** XX ** x + tp[D11] ** C1 ** x + tp[D11] ** D11 ** W

In[7]:= NCCollect[%,W] ------- Collect together terms containing W -------

Out[7]= (-1 + tp[D11] ** D11) ** W + tp[B1] ** XX ** x + tp[D11] ** C1 ** x

In[8]:= inv[-1+tp[D11]**D11]**%  ------ Apply inverse to clear coefficient of W-

Out[8]= inv[-1 + tp[D11] ** D11] **
   ((-1 + tp[D11] ** D11) ** W + tp[B1] ** XX ** x + tp[D11] ** C1 ** x)

In[9]:= NCE[%] ---- NonCommutativeExpand expands expression to get cancellation-

Out[9]= W + inv[-1 + tp[D11] ** D11] ** tp[B1] ** XX ** x +
   inv[-1 + tp[D11] ** D11] ** tp[D11] ** C1 ** x

In[10]:= Solve[%==0,W]  ------ Solve "expression=0" for W to get critical W ---

Out[10]= {{W ->
     -inv[-1 + tp[D11] ** D11] ** tp[B1] ** XX ** x -
      inv[-1 + tp[D11] ** D11] ** tp[D11] ** C1 ** x}}

In[11]:= CRW = W/.%[[1]]

Out[11]= -inv[-1 + tp[D11] ** D11] ** tp[B1] ** XX ** x -
   inv[-1 + tp[D11] ** D11] ** tp[D11] ** C1 ** x

In[12]:= H/.W->CRW  ----- Substitute critical W into the Hamiltonian -------

Out[12]= ((-tp[x] ** XX ** B1 ** inv[-1 + tp[D11] ** D11] -
         tp[x] ** tp[C1] ** D11 ** inv[-1 + tp[D11] ** D11]) ** tp[D11] +
      tp[x] ** tp[C1]) ** (C1 ** x +
      D11 ** (-inv[-1 + tp[D11] ** D11] ** tp[B1] ** XX ** x -
         inv[-1 + tp[D11] ** D11] ** tp[D11] ** C1 ** x)) -
   (-tp[x] ** XX ** B1 ** inv[-1 + tp[D11] ** D11] -
      tp[x] ** tp[C1] ** D11 ** inv[-1 + tp[D11] ** D11]) **
    (-inv[-1 + tp[D11] ** D11] ** tp[B1] ** XX ** x -
      inv[-1 + tp[D11] ** D11] ** tp[D11] ** C1 ** x) +
   (2*((-tp[x] ** XX ** B1 ** inv[-1 + tp[D11] ** D11] -
           tp[x] ** tp[C1] ** D11 ** inv[-1 + tp[D11] ** D11]) ** tp[B1] +
        tp[x] ** tp[A]) ** XX ** x +
      2*tp[x] ** XX ** (A ** x +
        B1 ** (-inv[-1 + tp[D11] ** D11] ** tp[B1] ** XX ** x -
```

```
            inv[-1 + tp[D11] ** D11] ** tp[D11] ** C1 ** x)))/2

In[13]:= NCE[%]  ---- expand again looking for cancellation ------

Out[13]= tp[x] ** XX ** A ** x + tp[x] ** tp[A] ** XX ** x +
   tp[x] ** tp[C1] ** C1 ** x -
   2*tp[x] ** XX ** B1 ** inv[-1 + tp[D11] ** D11] ** tp[B1] ** XX ** x -
   2*tp[x] ** XX ** B1 ** inv[-1 + tp[D11] ** D11] ** tp[D11] ** C1 ** x -
   2*tp[x] ** tp[C1] ** D11 ** inv[-1 + tp[D11] ** D11] ** tp[B1] ** XX **
     x - 2*tp[x] ** tp[C1] ** D11 ** inv[-1 + tp[D11] ** D11] ** tp[D11] **
     C1 ** x - tp[x] ** XX ** B1 ** inv[-1 + tp[D11] ** D11] **
    inv[-1 + tp[D11] ** D11] ** tp[B1] ** XX ** x -
   tp[x] ** XX ** B1 ** inv[-1 + tp[D11] ** D11] **
    inv[-1 + tp[D11] ** D11] ** tp[D11] ** C1 ** x -
   tp[x] ** tp[C1] ** D11 ** inv[-1 + tp[D11] ** D11] **
    inv[-1 + tp[D11] ** D11] ** tp[B1] ** XX ** x -
   tp[x] ** tp[C1] ** D11 ** inv[-1 + tp[D11] ** D11] **
    inv[-1 + tp[D11] ** D11] ** tp[D11] ** C1 ** x +
   tp[x] ** XX ** B1 ** inv[-1 + tp[D11] ** D11] ** tp[D11] ** D11 **
    inv[-1 + tp[D11] ** D11] ** tp[B1] ** XX ** x +
   tp[x] ** XX ** B1 ** inv[-1 + tp[D11] ** D11] ** tp[D11] ** D11 **
    inv[-1 + tp[D11] ** D11] ** tp[D11] ** C1 ** x +
   tp[x] ** tp[C1] ** D11 ** inv[-1 + tp[D11] ** D11] ** tp[D11] ** D11 **
    inv[-1 + tp[D11] ** D11] ** tp[B1] ** XX ** x +
   tp[x] ** tp[C1] ** D11 ** inv[-1 + tp[D11] ** D11] ** tp[D11] ** D11 **
    inv[-1 + tp[D11] ** D11] ** tp[D11] ** C1 ** x

In[14]:= NCC[%,inv[-1+tp[D11]**D11]]  ---- collect together terms with inverses-

Out[14]= (tp[x] ** XX ** B1 + tp[x] ** tp[C1] ** D11) **
    inv[-1 + tp[D11] ** D11] ** (-tp[B1] ** XX ** x - tp[D11] ** C1 ** x) +
   tp[x] ** XX ** A ** x + tp[x] ** tp[A] ** XX ** x +
   tp[x] ** tp[C1] ** C1 ** x

In[15]:= NCC[%,{x,tp[x]}]  ----- then collect x, tp[x] on ends (quadratic form)-

Out[15]= tp[x] ** (XX ** A + tp[A] ** XX + tp[C1] ** C1) ** x +
   tp[x] ** (XX ** B1 + tp[C1] ** D11) ** inv[-1 + tp[D11] ** D11] **
    (-tp[B1] ** XX - tp[D11] ** C1) ** x

-----  Passivity condition is that this is =<0 for all x.  This is equivalent to
negative semidefiniteness of

In[16]:= ricd = %/.x->1

Out[16]= XX ** A + tp[A] ** XX + tp[C1] ** C1 +
   (XX ** B1 + tp[C1] ** D11) ** inv[-1 + tp[D11] ** D11] **
    (-tp[B1] ** XX - tp[D11] ** C1)

In[17]:= ric0=%/.D11->0  ---- When D11=0, you get well-known Riccati -----
```

```
Out[17]= XX ** A + tp[A] ** XX + tp[C1] ** C1 + XX ** B1 ** tp[B1] ** XX

-----This proves the theorem -------------------

In[18]:= Quit


                  OTHER ISSUES

  Another issue is that of finding a stabilizing solution to the
Algebraic Riccati Equation ricc0 ==0. Namely, find an XX so that
(fb)           A-B1**tp[B1]**XX
is stable. Note that CRW =-tp[B1]**XX so (fb) is
          A+B1**CRW


END DEMO ############################################ END DEMO
```

5.2 MEASUREMENT FEEDBACK H^{∞} CONTROL

We wish to analyse the dissipativity condition on 2 port systems with a one port system in feedback. The basic question is when does feedback exist which makes the full system dissipative and internally stable? This example is based on the paper [BHW]. Here is a listing of the definitions for a nonlinear system which is affine in the input. It is part of our SYS file named SYSDefIA.m - again there are TeX transcriptions.

INPUT AFFINE SYSTEM DEFINITIONS

```
(*
  This file loads in basic definitions of two systems and of
the energy balance relations various connections of the
systems satisfy.

                  __________________
                 |                 |
           W  ---->--|              G1 |---->--- out1
                 |  F              |
           U  ---->--|              G2 |---->--- Y
                 |_________________|
                  ______________
                 |              |
            U  ----<--| g     f   |----<--- Y
                 |______________|

                        Figure 2.
```

```
                 dx/dt  =  F[x,W,U]
                  out1  =  G1[x,W,U]
                    Y   =  G2[x,W,U]

                 dz/dt  =  f[z,Y]
                    U   =  g[z,Y]

 An Input Affine (IA) one port system is
---------------------------------------- *)
                   SetNonCommutative[f,g,a,b,c,dd,z]

                       f[z_,Y_]:=a[z]+b[z]**Y
                       g[z_,Y_]:=c[z]+dd[z]**Y

(* An Input Affine (IA) two port system is *)

                   SetNonCommutative[W,U,Y,DW,DU,DY]
                   SetNonCommutative[A,x,B1,B2,C1,C2,G1,G2]
                   SetNonCommutative[D11,D22,D12,D21]
                     D11[x_]:=0
                     D22[x_]:=0

                   F[x_,W_,U_]:=A[x]+B1[x]**W+B2[x]**U
                   G1[x_,W_,U_]:=C1[x]+D11[x]**W+D12[x]**U;
                            out1=G1[x,W,U];
                   G2[x_,W_,U_]:=C2[x]+D21[x]**W+D22[x]**U;
                            out2=G2[x,W,U];
                            G2I[x_,Y_,U_]:=inv[D21[x]]**Y-inv[D21[x]]**C2[x];
```

ENERGY BALANCE EQUATIONS

We begin with notation for analyzing the DISSIPATIVITY of the systems obtained by connecting f,g to F,G in several different ways. The energy function on the statespace is denoted by e. HWUY below is the Hamiltonian of the **two decoupled systems** where inputs are W,U, and Y.

$$HWUY = |out|^2 - |W|^2 + PP \cdot f(z,Y) + p \cdot F(x,W,U)$$

```
SetNonCommutative[F,G1,G2,f,g,p,PP];
HWUY = tp[out1]**out1-tp[W]**W+
                        (p**F[x,W,U]+tp[F[x,W,U]]**tp[p])/2+
                                      (PP**f[z,Y]+tp[f[z,Y]]**tp[PP])/2;
```

CONNECTING INPUT AND OUTPUT

If we connect the Y output of the 2 port to the f,g input then the resulting system has Hamiltonian function HWU.

$$HWU = HWUY \quad \text{with the substitution} \quad Y \to G2(x,W,U)$$

```
HWU=HWUY/.Y->G2[x,W,U];
```

If we connect the U input of the 2 port to the f,g output then the resulting system has Hamiltonian function HWY.

$$HWY = HWUY \quad \text{with the substitution} \quad U \to g(z, Y)$$

```
HWY=HWUY/.U->g[z,Y];
```

If we connect the two systems in feedback, that is tie off U and Y, then the resulting Hamiltonian function is

$$HW = HWY \quad \text{with the substitution} \quad Y \to G2(x, W, U)$$

```
HW=HWY/.Y->G2[x,W,U];
```

By definition (see Section I) the closed loop system being e-DISSIPATIVE corresponds to the energy balance function HW above being negative.

Note that the function HWUY contains both state and dual variables. Dual variables are defined by the gradient of the energy function, as follows:

$$p = \nabla_x e \quad \text{and} \quad PP = \nabla_z e$$

In the following, when we impose the IA assumptions (see (1.5) and (1.6)), we will often specialize to a plant which satisfies

$$D11(x) = D22(x) = 0,$$

$$D12(x)^T D12(x) = e1(x) > 0,$$

$$D21(x) D21(x)^T = e2(x) > 0.$$

and a compensator with d(z)=0. We now write out energy balance formulas for the plant and compensator after Figure 2 (under these assumptions) purely in terms of state space variables x and z (designated by the prefix s for state space).

PURE STATESPACE VARIABLES

STATESPACE -x,z. While the HWUY formulas mix x's and p 's , the next ones are purely on statespace variables x and z. Executable formulas for

$$p = \nabla_x e \quad \text{and} \quad PP = \nabla_z e$$

are inserted by the rule

```
SNC[GEx,GEz];
ruxz={p->tp[GEx[x,z]], PP->tp[GEz[x,z]]};
```

Here,

```
GEx[x,z], GEz[x,z]
```

stand for column vectors $\nabla_x^T e$, $\nabla_z^T e$, respectively. This unfortunate choice of notation is a holdover from earlier versions of the software.

The storage function e is homogeneous

```
    GEx[0,0]=0;
    GEz[0,0]=0;

sHWUY=HWUY/.ruxz;
sHWU=HWU/.ruxz;
sHWY=HWY/.ruxz;
sHW=HW/.ruxz;
```

RESULTS STORED FOR FAST EXECUTION

The following are redundant in that they can be computed from the above expressions.

This is the W which maximizes Hamiltonian

```
ruCritW= {W -> tp[B1[x]] ** GEx[x, z]/2 +
     tp[D21[x]] ** tp[b[z]] ** GEz[x, z]/2}

CritW[x_,z_,b_]:= tp[B1[x]] ** GEx[x, z]/2 +
                tp[D21[x]] ** tp[b] ** GEz[x, z]/2;

(*  and you get  *)

sHWo = tp[A[x]] ** GEx[x, z]/2 + tp[C1[x]] ** C1[x] + tp[GEx[x, z]] ** A[x]/2 +
  tp[GEz[x, z]] ** a[z]/2 + tp[a[z]] ** GEz[x, z]/2 +
  tp[C1[x]] ** D12[x] ** c[z] + tp[C2[x]] ** tp[b[z]] ** GEz[x, z]/2 +
  tp[GEx[x, z]] ** B2[x] ** c[z]/2 + tp[GEz[x, z]] ** b[z] ** C2[x]/2 +
  tp[c[z]] ** e1[x] ** c[z] + tp[c[z]] ** tp[B2[x]] ** GEx[x, z]/2 +
  tp[c[z]] ** tp[D12[x]] ** C1[x] +
  tp[GEx[x, z]] ** B1[x] ** tp[B1[x]] ** GEx[x, z]/4 +
  tp[GEx[x, z]] ** B1[x] ** tp[D21[x]] ** tp[b[z]] ** GEz[x, z]/4 +
  tp[GEz[x, z]] ** b[z] ** D21[x] ** tp[B1[x]] ** GEx[x, z]/4 +
  tp[GEz[x, z]] ** b[z] ** e2[x] ** tp[b[z]] ** GEz[x, z]/4


(* This is the U which minimizes Hamiltonian *)
ruCritU = {U -> -inv[e1[x]] ** tp[B2[x]] ** GEx[x, z]/2 -
     inv[e1[x]] ** tp[D12[x]] ** C1[x]};

CritU[x_,z_]:=-inv[e1[x]] ** tp[B2[x]] ** GEx[x, z]/2 -
     inv[e1[x]] ** tp[D12[x]] ** C1[x];


END OF FILE  ############################################ END OF FILE
```

5.2.1 DERIVATION OF CritW AND sHWo

Now the demo starts by taking the Hamiltonian for the closed loop system sHW and optimizing it in W to get what we denote sHWo. Soon the demo requires serious knowledge of [BHW] so if the reader has problems he is urged to read SYSHinf-TAC.m a file which gives a rather complete account of the terse and unmotivated calculations here.

```
In[3]:= <<NCAlgebra

In[4]:= <<../SYSDefIA.m

In[5]:= Sub[sHW,dd[z]->0]

Out[5]= -tp[W] ** W + (tp[c[z]] ** tp[D12[x]] + tp[C1[x]]) **
     (C1[x] + D12[x] ** c[z]) + ((tp[W] ** tp[B1[x]] + tp[c[z]] ** tp[B2[x]] +
          tp[A[x]]) ** GEx[x, z] +
       tp[GEx[x, z]] ** (A[x] + B1[x] ** W + B2[x] ** c[z]))/2 +
   (tp[GEz[x, z]] ** (b[z] ** (C2[x] + D21[x] ** W) + a[z]) +
       ((tp[W] ** tp[D21[x]] + tp[C2[x]]) ** tp[b[z]] + tp[a[z]]) ** GEz[x, z])
      /2

The critical W, gotten by taking the gradient of % and setting it to 0,
is

In[6]:= Crit[%,W]

Out[6]= {W -> tp[B1[x]] ** GEx[x, z]/2 +
      tp[D21[x]] ** tp[b[z]] ** GEz[x, z]/2}

In[7]:= Sub[%%,%]

Out[7]= -(tp[GEx[x, z]] ** B1[x]/2 + tp[GEz[x, z]] ** b[z] ** D21[x]/2) **
      (tp[B1[x]] ** GEx[x, z]/2 + tp[D21[x]] ** tp[b[z]] ** GEz[x, z]/2) +
   (tp[c[z]] ** tp[D12[x]] + tp[C1[x]]) ** (C1[x] + D12[x] ** c[z]) +
   (((tp[GEx[x, z]] ** B1[x]/2 + tp[GEz[x, z]] ** b[z] ** D21[x]/2) **
           tp[B1[x]] + tp[c[z]] ** tp[B2[x]] + tp[A[x]]) ** GEx[x, z] +
       tp[GEx[x, z]] ** (A[x] + B1[x] **
           (tp[B1[x]] ** GEx[x, z]/2 + tp[D21[x]] ** tp[b[z]] ** GEz[x, z]/2) \
           + B2[x] ** c[z]))/2 +
   (tp[GEz[x, z]] ** (b[z] ** (C2[x] +
             D21[x] ** (tp[B1[x]] ** GEx[x, z]/2 +
                tp[D21[x]] ** tp[b[z]] ** GEz[x, z]/2)) + a[z]) +
       (((tp[GEx[x, z]] ** B1[x]/2 + tp[GEz[x, z]] ** b[z] ** D21[x]/2) **
              tp[D21[x]] + tp[C2[x]]) ** tp[b[z]] + tp[a[z]]) ** GEz[x, z])/2


         ------Check this against the stored formula for sHWo
In[8]:= NCE[%-sHWo]

Out[8]= -tp[c[z]] ** e1[x] ** c[z] +
```

```
   tp[c[z]] ** tp[D12[x]] ** D12[x] ** c[z] -
   tp[GEz[x, z]] ** b[z] ** e2[x] ** tp[b[z]] ** GEz[x, z]/4 +
   tp[GEz[x, z]] ** b[z] ** D21[x] ** tp[D21[x]] ** tp[b[z]] ** GEz[x, z]/4

In[9]:= <<../SYSTEMS.simplify

In[10]:= Sub[%%,rue]

Out[10]= 0
```

The output sHWo is also recorded in the file SYSDefIA.m above. Note that neither sHWo nor sHWoWIA depends on a or c.

5.2.2 THE MIN/MAX IN U

Also useful is the optimum CritU for

$$Hopt := \min_U \max_W sHW = \min_U sHWo = \max_W \min_U sHW.$$

For WIA systems there is not an elegant formula for CritU and possibly is not unique. For IA systems, one gets the concrete formula:

$$CritU(x,z) := -e1(x)^{-1}B2(x)^T\nabla_x e(x,z)/2 - e1(x)^{-1}D12(x)^T C1(x)$$

5.2.3 DERIVATIONS OF IAX, Critc, AND Hopt

Now we optimize sHWo in c[z] variable and restrict to x=z to get the Doyle Glover Kargonekar Francis condition for IA systems.

```
In[3]:= <<NCAlgebra
In[4]:= <<../SYSDefIA.m
In[5]:= Crit[sHWo,c[z]]

Out[5]= {c[z] ->
    -inv[e1[x]] ** tp[B2[x]] ** GEx[x, z]/2 -
     inv[e1[x]] ** tp[D12[x]] ** C1[x]}      ----------- this is ruCritU
In[6]:= Hopt = NCE[Sub[sHWo,%5]]

Out[6]= tp[A[x]] ** GEx[x, z]/2 + tp[C1[x]] ** C1[x] +
   tp[GEx[x, z]] ** A[x]/2 + tp[GEz[x, z]] ** a[z]/2 +
   tp[a[z]] ** GEz[x, z]/2 + tp[C2[x]] ** tp[b[z]] ** GEz[x, z]/2 +
   tp[GEz[x, z]] ** b[z] ** C2[x]/2 +
   tp[GEx[x, z]] ** B1[x] ** tp[B1[x]] ** GEx[x, z]/4 -
   tp[C1[x]] ** D12[x] ** inv[e1[x]] ** tp[B2[x]] ** GEx[x, z]/2 -
   tp[C1[x]] ** D12[x] ** inv[e1[x]] ** tp[D12[x]] ** C1[x] +
   tp[GEx[x, z]] ** B1[x] ** tp[D21[x]] ** tp[b[z]] ** GEz[x, z]/4 -
```

```
   tp[GEx[x, z]] ** B2[x] ** inv[e1[x]] ** tp[B2[x]] ** GEx[x, z]/4 -
   tp[GEx[x, z]] ** B2[x] ** inv[e1[x]] ** tp[D12[x]] ** C1[x]/2 +
   tp[GEz[x, z]] ** b[z] ** D21[x] ** tp[B1[x]] ** GEx[x, z]/4 +
   tp[GEz[x, z]] ** b[z] ** e2[x] ** tp[b[z]] ** GEz[x, z]/4

In[7]:= Sub[Hopt,x->z]

Out[7]= tp[A[z]] ** GEx[z, z]/2 + tp[C1[z]] ** C1[z] +
   tp[GEx[z, z]] ** A[z]/2 + tp[GEz[z, z]] ** a[z]/2 +
   tp[a[z]] ** GEz[z, z]/2 + tp[C2[z]] ** tp[b[z]] ** GEz[z, z]/2 +
   tp[GEz[z, z]] ** b[z] ** C2[z]/2 +
   tp[GEx[z, z]] ** B1[z] ** tp[B1[z]] ** GEx[z, z]/4 -
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[B2[z]] ** GEx[z, z]/2 -
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[D12[z]] ** C1[z] +
   tp[GEx[z, z]] ** B1[z] ** tp[D21[z]] ** tp[b[z]] ** GEz[z, z]/4 -
   tp[GEx[z, z]] ** B2[z] ** inv[e1[z]] ** tp[B2[z]] ** GEx[z, z]/4 -
   tp[GEx[z, z]] ** B2[z] ** inv[e1[z]] ** tp[D12[z]] ** C1[z]/2 +
   tp[GEz[z, z]] ** b[z] ** D21[z] ** tp[B1[z]] ** GEx[z, z]/4 +
   tp[GEz[z, z]] ** b[z] ** e2[z] ** tp[b[z]] ** GEz[z, z]/4

In[8]:= <<../SYSTEMS.Hinf_control

In[9]:= ruXXYYI

Out[9]= {GEz[x_, x_] :> 0, GEx[x_, x_] :> 2*XX[x], GEx[x_, 0] :> 2*YYI[x]}

In[10]:= Sub[%7,%]

Out[10]= tp[A[z]] ** XX[z] + tp[C1[z]] ** C1[z] + tp[XX[z]] ** A[z] +
   tp[XX[z]] ** B1[z] ** tp[B1[z]] ** XX[z] -
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[B2[z]] ** XX[z] -
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[D12[z]] ** C1[z] -
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[B2[z]] ** XX[z] -
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[D12[z]] ** C1[z]

In[11]:= NCE[%-IAX[z]]
Out[11]= 0
```

5.2.4 DERIVATION OF IAYI

This is an advanced exercise in the use of the glossary at the end of this document file. The glossary stores the basic formulas for solutions to the IA H^∞ control problem. We use them here to familiarize the reader with the GLOSSARY.

```
In[3]:= <<NCAlgebra

In[4]:= <<../SYSDefIA.m

In[5]:= Sub[sHWo,z->0]
```

```
Out[5]= tp[A[x]] ** GEx[x, 0]/2 + tp[C1[x]] ** C1[x] +
   tp[GEx[x, 0]] ** A[x]/2 + tp[GEz[x, 0]] ** a[0]/2 +
   tp[a[0]] ** GEz[x, 0]/2 + tp[C1[x]] ** D12[x] ** c[0] +
   tp[C2[x]] ** tp[b[0]] ** GEz[x, 0]/2 + tp[GEx[x, 0]] ** B2[x] ** c[0]/2 +
   tp[GEz[x, 0]] ** b[0] ** C2[x]/2 + tp[c[0]] ** e1[x] ** c[0] +
   tp[c[0]] ** tp[B2[x]] ** GEx[x, 0]/2 + tp[c[0]] ** tp[D12[x]] ** C1[x] +
   tp[GEx[x, 0]] ** B1[x] ** tp[B1[x]] ** GEx[x, 0]/4 +
   tp[GEx[x, 0]] ** B1[x] ** tp[D21[x]] ** tp[b[0]] ** GEz[x, 0]/4 +
   tp[GEz[x, 0]] ** b[0] ** D21[x] ** tp[B1[x]] ** GEx[x, 0]/4 +
   tp[GEz[x, 0]] ** b[0] ** e2[x] ** tp[b[0]] ** GEz[x, 0]/4

In[6]:= ruhomog

Out[6]= {A[0] -> 0, C1[0] -> 0, C2[0] -> 0, a[0] -> 0, c[0] -> 0}

In[7]:= Sub[%%,%]

Out[7]= tp[A[x]] ** GEx[x, 0]/2 + tp[C1[x]] ** C1[x] +
   tp[GEx[x, 0]] ** A[x]/2 + tp[C2[x]] ** tp[b[0]] ** GEz[x, 0]/2 +
   tp[GEz[x, 0]] ** b[0] ** C2[x]/2 +
   tp[GEx[x, 0]] ** B1[x] ** tp[B1[x]] ** GEx[x, 0]/4 +
   tp[GEx[x, 0]] ** B1[x] ** tp[D21[x]] ** tp[b[0]] ** GEz[x, 0]/4 +
   tp[GEz[x, 0]] ** b[0] ** D21[x] ** tp[B1[x]] ** GEx[x, 0]/4 +
   tp[GEz[x, 0]] ** b[0] ** e2[x] ** tp[b[0]] ** GEz[x, 0]/4

In[8]:= Sub[%,ruXXYYI]

Out[8]= tp[A[x]] ** YYI[x] + tp[C1[x]] ** C1[x] + tp[YYI[x]] ** A[x] +
   tp[C2[x]] ** tp[b[0]] ** GEz[x, 0]/2 + tp[GEz[x, 0]] ** b[0] ** C2[x]/2 +
   tp[YYI[x]] ** B1[x] ** tp[B1[x]] ** YYI[x] +
   tp[GEz[x, 0]] ** b[0] ** D21[x] ** tp[B1[x]] ** YYI[x]/2 +
   tp[GEz[x, 0]] ** b[0] ** e2[x] ** tp[b[0]] ** GEz[x, 0]/4 +
   tp[YYI[x]] ** B1[x] ** tp[D21[x]] ** tp[b[0]] ** GEz[x, 0]/2

In[9]:= SubSym[%,ruqb]

Out[9]= tp[A[x]] ** YYI[x] + tp[C1[x]] ** C1[x] + tp[C2[x]] ** q[x, 0]/2 +
   tp[YYI[x]] ** A[x] + tp[q[x, 0]] ** C2[x]/2 +
   tp[q[x, 0]] ** e2[x] ** q[x, 0]/4 +
   tp[YYI[x]] ** B1[x] ** tp[B1[x]] ** YYI[x] +
   tp[YYI[x]] ** B1[x] ** tp[D21[x]] ** q[x, 0]/2 +
   tp[q[x, 0]] ** D21[x] ** tp[B1[x]] ** YYI[x]/2

In[10]:= Crit[%,q[x,0]]

Out[10]= {q[x, 0] ->
    -2*inv[e2[x]] ** C2[x] - 2*inv[e2[x]] ** D21[x] ** tp[B1[x]] ** YYI[x]}

In[11]:= Sub[%%,%]
```

```
Out[11]= (-2*tp[C2[x]] ** inv[e2[x]] -
       2*tp[YYI[x]] ** B1[x] ** tp[D21[x]] ** inv[e2[x]]) ** C2[x]/2 +
   tp[A[x]] ** YYI[x] + tp[C1[x]] ** C1[x] +
   tp[C2[x]] ** (-2*inv[e2[x]] ** C2[x] -
       2*inv[e2[x]] ** D21[x] ** tp[B1[x]] ** YYI[x])/2 +
   tp[YYI[x]] ** A[x] + (-2*tp[C2[x]] ** inv[e2[x]] -
       2*tp[YYI[x]] ** B1[x] ** tp[D21[x]] ** inv[e2[x]]) ** e2[x] **
     (-2*inv[e2[x]] ** C2[x] - 2*inv[e2[x]] ** D21[x] ** tp[B1[x]] ** YYI[x])/
    4 + (-2*tp[C2[x]] ** inv[e2[x]] -
       2*tp[YYI[x]] ** B1[x] ** tp[D21[x]] ** inv[e2[x]]) ** D21[x] **
     tp[B1[x]] ** YYI[x]/2 + tp[YYI[x]] ** B1[x] ** tp[B1[x]] ** YYI[x] +
   tp[YYI[x]] ** B1[x] ** tp[D21[x]] **
     (-2*inv[e2[x]] ** C2[x] - 2*inv[e2[x]] ** D21[x] ** tp[B1[x]] ** YYI[x])/
    2

In[12]:= NCE[%]

Out[12]= tp[A[x]] ** YYI[x] + tp[C1[x]] ** C1[x] + tp[YYI[x]] ** A[x] -
   tp[C2[x]] ** inv[e2[x]] ** C2[x] +
   tp[YYI[x]] ** B1[x] ** tp[B1[x]] ** YYI[x] -
   tp[C2[x]] ** inv[e2[x]] ** D21[x] ** tp[B1[x]] ** YYI[x] -
   tp[YYI[x]] ** B1[x] ** tp[D21[x]] ** inv[e2[x]] ** C2[x] -
   tp[YYI[x]] ** B1[x] ** tp[D21[x]] ** inv[e2[x]] ** D21[x] ** tp[B1[x]] **
    YYI[x]

In[13]:= NCE[%-IAYI[x]]

Out[13]= 0
```

5.2.5 DERIVATION OF CRITICAL q, k, AND bterm

```
In[2]:= <<SYSTEMSNL

In[16]:= SubSym[sHWo,ruc]

Out[16]= tp[A[x]] ** GEx[x, z]/2 + tp[C1[x]] ** C1[x] +
   tp[GEx[x, z]] ** A[x]/2 + tp[GEz[x, z]] ** a[z]/2 +
   tp[a[z]] ** GEz[x, z]/2 + tp[C2[x]] ** tp[b[z]] ** GEz[x, z]/2 +
   tp[GEz[x, z]] ** b[z] ** C2[x]/2 -
   (tp[C1[z]] ** D12[z] + tp[XX[z]] ** B2[z]) ** inv[e1[z]] ** tp[B2[x]] **
     GEx[x, z]/2 - (tp[C1[z]] ** D12[z] + tp[XX[z]] ** B2[z]) **
    inv[e1[z]] ** tp[D12[x]] ** C1[x] -
   tp[C1[x]] ** D12[x] ** inv[e1[z]] **
    (tp[B2[z]] ** XX[z] + tp[D12[z]] ** C1[z]) +
   tp[GEx[x, z]] ** B1[x] ** tp[B1[x]] ** GEx[x, z]/4 -
   tp[GEx[x, z]] ** B2[x] ** inv[e1[z]] **
     (tp[B2[z]] ** XX[z] + tp[D12[z]] ** C1[z])/2 +
   tp[GEx[x, z]] ** B1[x] ** tp[D21[x]] ** tp[b[z]] ** GEz[x, z]/4 +
   tp[GEz[x, z]] ** b[z] ** D21[x] ** tp[B1[x]] ** GEx[x, z]/4 +
```

```
   (tp[C1[z]] ** D12[z] + tp[XX[z]] ** B2[z]) ** inv[e1[z]] ** tp[D12[x]] **
    D12[x] ** inv[e1[z]] ** (tp[B2[z]] ** XX[z] + tp[D12[z]] ** C1[z]) +
   tp[GEz[x, z]] ** b[z] ** D21[x] ** tp[D21[x]] ** tp[b[z]] ** GEz[x, z]/4

In[17]:= NCE[SubSym[%,rua]]

Out[17]= tp[A[x]] ** GEx[x, z]/2 + tp[A[z]] ** GEz[x, z]/2 +
   tp[C1[x]] ** C1[x] + tp[GEx[x, z]] ** A[x]/2 + tp[GEz[x, z]] ** A[z]/2 +
   tp[C2[x]] ** tp[b[z]] ** GEz[x, z]/2 -
   tp[C2[z]] ** tp[b[z]] ** GEz[x, z]/2 + tp[GEz[x, z]] ** b[z] ** C2[x]/2 -
   tp[GEz[x, z]] ** b[z] ** C2[z]/2 +
   tp[GEx[x, z]] ** B1[x] ** tp[B1[x]] ** GEx[x, z]/4 +
   tp[GEz[x, z]] ** B1[z] ** tp[B1[z]] ** XX[z]/2 +
   tp[XX[z]] ** B1[z] ** tp[B1[z]] ** GEz[x, z]/2 -
   tp[C1[x]] ** D12[x] ** inv[e1[z]] ** tp[B2[z]] ** XX[z] -
   tp[C1[x]] ** D12[x] ** inv[e1[z]] ** tp[D12[z]] ** C1[z] -
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[B2[x]] ** GEx[x, z]/2 -
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[B2[z]] ** GEz[x, z]/2 -
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[D12[x]] ** C1[x] +
   tp[GEx[x, z]] ** B1[x] ** tp[D21[x]] ** tp[b[z]] ** GEz[x, z]/4 -
   tp[GEx[x, z]] ** B2[x] ** inv[e1[z]] ** tp[B2[z]] ** XX[z]/2 -
   tp[GEx[x, z]] ** B2[x] ** inv[e1[z]] ** tp[D12[z]] ** C1[z]/2 -
   tp[GEz[x, z]] ** B2[z] ** inv[e1[z]] ** tp[B2[z]] ** XX[z]/2 -
   tp[GEz[x, z]] ** B2[z] ** inv[e1[z]] ** tp[D12[z]] ** C1[z]/2 +
   tp[GEz[x, z]] ** b[z] ** D21[x] ** tp[B1[x]] ** GEx[x, z]/4 -
   tp[GEz[x, z]] ** b[z] ** D21[z] ** tp[B1[z]] ** XX[z]/2 -
   tp[XX[z]] ** B1[z] ** tp[D21[z]] ** tp[b[z]] ** GEz[x, z]/2 -
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[B2[x]] ** GEx[x, z]/2 -
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[B2[z]] ** GEz[x, z]/2 -
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[D12[x]] ** C1[x] +
   tp[GEz[x, z]] ** b[z] ** D21[x] ** tp[D21[x]] ** tp[b[z]] ** GEz[x, z]/4 +
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[D12[x]] ** D12[x] ** inv[e1[z]] **
    tp[B2[z]] ** XX[z] + tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[D12[x]] **
    D12[x] ** inv[e1[z]] ** tp[D12[z]] ** C1[z] +
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[D12[x]] ** D12[x] ** inv[e1[z]] **
    tp[B2[z]] ** XX[z] + tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[D12[x]] **
    D12[x] ** inv[e1[z]] ** tp[D12[z]] ** C1[z]

In[18]:= SubSym[%,tp[GEz[x,z]]**b[z]->q[x,z]]

Out[18]= q[x, z] ** C2[x]/2 - q[x, z] ** C2[z]/2 + tp[A[x]] ** GEx[x, z]/2 +
   tp[A[z]] ** GEz[x, z]/2 + tp[C1[x]] ** C1[x] +
   tp[C2[x]] ** tp[q[x, z]]/2 - tp[C2[z]] ** tp[q[x, z]]/2 +
   tp[GEx[x, z]] ** A[x]/2 + tp[GEz[x, z]] ** A[z]/2 +
   q[x, z] ** D21[x] ** tp[B1[x]] ** GEx[x, z]/4 +
   q[x, z] ** D21[x] ** tp[D21[x]] ** tp[q[x, z]]/4 -
   q[x, z] ** D21[z] ** tp[B1[z]] ** XX[z]/2 +
   tp[GEx[x, z]] ** B1[x] ** tp[B1[x]] ** GEx[x, z]/4 +
   tp[GEx[x, z]] ** B1[x] ** tp[D21[x]] ** tp[q[x, z]]/4 +
   tp[GEz[x, z]] ** B1[z] ** tp[B1[z]] ** XX[z]/2 +
```

```
   tp[XX[z]] ** B1[z] ** tp[B1[z]] ** GEz[x, z]/2 -
   tp[XX[z]] ** B1[z] ** tp[D21[z]] ** tp[q[x, z]]/2 -
   tp[C1[x]] ** D12[x] ** inv[e1[z]] ** tp[B2[z]] ** XX[z] -
   tp[C1[x]] ** D12[x] ** inv[e1[z]] ** tp[D12[z]] ** C1[z] -
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[B2[x]] ** GEx[x, z]/2 -
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[B2[z]] ** GEz[x, z]/2 -
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[D12[x]] ** C1[x] -
   tp[GEx[x, z]] ** B2[x] ** inv[e1[z]] ** tp[B2[z]] ** XX[z]/2 -
   tp[GEx[x, z]] ** B2[x] ** inv[e1[z]] ** tp[D12[z]] ** C1[z]/2 -
   tp[GEz[x, z]] ** B2[z] ** inv[e1[z]] ** tp[B2[z]] ** XX[z]/2 -
   tp[GEz[x, z]] ** B2[z] ** inv[e1[z]] ** tp[D12[z]] ** C1[z]/2 -
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[B2[x]] ** GEx[x, z]/2 -
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[B2[z]] ** GEz[x, z]/2 -
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[D12[x]] ** C1[x] +
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[D12[x]] ** D12[x] ** inv[e1[z]] **
    tp[B2[z]] ** XX[z] + tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[D12[x]] **
    D12[x] ** inv[e1[z]] ** tp[D12[z]] ** C1[z] +
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[D12[x]] ** D12[x] ** inv[e1[z]] **
    tp[B2[z]] ** XX[z] + tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[D12[x]] **
    D12[x] ** inv[e1[z]] ** tp[D12[z]] ** C1[z]

In[19]:= SubSym[%,rue]

Out[19]= q[x, z] ** C2[x]/2 - q[x, z] ** C2[z]/2 + tp[A[x]] ** GEx[x, z]/2 +
   tp[A[z]] ** GEz[x, z]/2 + tp[C1[x]] ** C1[x] +
   tp[C2[x]] ** tp[q[x, z]]/2 - tp[C2[z]] ** tp[q[x, z]]/2 +
   tp[GEx[x, z]] ** A[x]/2 + tp[GEz[x, z]] ** A[z]/2 +
   q[x, z] ** e2[x] ** tp[q[x, z]]/4 +
   q[x, z] ** D21[x] ** tp[B1[x]] ** GEx[x, z]/4 -
   q[x, z] ** D21[z] ** tp[B1[z]] ** XX[z]/2 +
   tp[GEx[x, z]] ** B1[x] ** tp[B1[x]] ** GEx[x, z]/4 +
   tp[GEx[x, z]] ** B1[x] ** tp[D21[x]] ** tp[q[x, z]]/4 +
   tp[GEz[x, z]] ** B1[z] ** tp[B1[z]] ** XX[z]/2 +
   tp[XX[z]] ** B1[z] ** tp[B1[z]] ** GEz[x, z]/2 -
   tp[XX[z]] ** B1[z] ** tp[D21[z]] ** tp[q[x, z]]/2 -
   tp[C1[x]] ** D12[x] ** inv[e1[z]] ** tp[B2[z]] ** XX[z] -
   tp[C1[x]] ** D12[x] ** inv[e1[z]] ** tp[D12[z]] ** C1[z] -
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[B2[x]] ** GEx[x, z]/2 -
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[B2[z]] ** GEz[x, z]/2 -
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[D12[x]] ** C1[x] -
   tp[GEx[x, z]] ** B2[x] ** inv[e1[z]] ** tp[B2[z]] ** XX[z]/2 -
   tp[GEx[x, z]] ** B2[x] ** inv[e1[z]] ** tp[D12[z]] ** C1[z]/2 -
   tp[GEz[x, z]] ** B2[z] ** inv[e1[z]] ** tp[B2[z]] ** XX[z]/2 -
   tp[GEz[x, z]] ** B2[z] ** inv[e1[z]] ** tp[D12[z]] ** C1[z]/2 -
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[B2[x]] ** GEx[x, z]/2 -
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[B2[z]] ** GEz[x, z]/2 -
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[D12[x]] ** C1[x] +
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** e1[x] ** inv[e1[z]] ** tp[B2[z]] **
    XX[z] + tp[C1[z]] ** D12[z] ** inv[e1[z]] ** e1[x] ** inv[e1[z]] **
    tp[D12[z]] ** C1[z] + tp[XX[z]] ** B2[z] ** inv[e1[z]] ** e1[x] **
```

```
    inv[e1[z]] ** tp[B2[z]] ** XX[z] +
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** e1[x] ** inv[e1[z]] ** tp[D12[z]] **
    C1[z]

In[20]:= Crit[%,tp[q[x,z]]]

Out[20]= {tp[q[x, z]] ->
    -2*inv[e2[x]] ** C2[x] + 2*inv[e2[x]] ** C2[z] -
     inv[e2[x]] ** D21[x] ** tp[B1[x]] ** GEx[x, z] +
     2*inv[e2[x]] ** D21[z] ** tp[B1[z]] ** XX[z]}

In[21]:= K=NCE[SubSym[%19,%20]]


Out[22]= tp[A[x]] ** GEx[x, z]/2 + tp[A[z]] ** GEz[x, z]/2 +
   tp[C1[x]] ** C1[x] + tp[GEx[x, z]] ** A[x]/2 + tp[GEz[x, z]] ** A[z]/2 -
   tp[C2[x]] ** inv[e2[x]] ** C2[x] + tp[C2[x]] ** inv[e2[x]] ** C2[z] +
   tp[C2[z]] ** inv[e2[x]] ** C2[x] - tp[C2[z]] ** inv[e2[x]] ** C2[z] +
   tp[GEx[x, z]] ** B1[x] ** tp[B1[x]] ** GEx[x, z]/4 +
   tp[GEz[x, z]] ** B1[z] ** tp[B1[z]] ** XX[z]/2 +
   tp[XX[z]] ** B1[z] ** tp[B1[z]] ** GEz[x, z]/2 -
   tp[C1[x]] ** D12[x] ** inv[e1[z]] ** tp[B2[z]] ** XX[z] -
   tp[C1[x]] ** D12[x] ** inv[e1[z]] ** tp[D12[z]] ** C1[z] -
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[B2[x]] ** GEx[x, z]/2 -
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[B2[z]] ** GEz[x, z]/2 -
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** tp[D12[x]] ** C1[x] -
   tp[C2[x]] ** inv[e2[x]] ** D21[x] ** tp[B1[x]] ** GEx[x, z]/2 +
   tp[C2[x]] ** inv[e2[x]] ** D21[z] ** tp[B1[z]] ** XX[z] +
   tp[C2[z]] ** inv[e2[x]] ** D21[x] ** tp[B1[x]] ** GEx[x, z]/2 -
   tp[C2[z]] ** inv[e2[x]] ** D21[z] ** tp[B1[z]] ** XX[z] -
   tp[GEx[x, z]] ** B1[x] ** tp[D21[x]] ** inv[e2[x]] ** C2[x]/2 +
   tp[GEx[x, z]] ** B1[x] ** tp[D21[x]] ** inv[e2[x]] ** C2[z]/2 -
   tp[GEx[x, z]] ** B2[x] ** inv[e1[z]] ** tp[B2[z]] ** XX[z]/2 -
   tp[GEx[x, z]] ** B2[x] ** inv[e1[z]] ** tp[D12[z]] ** C1[z]/2 -
   tp[GEz[x, z]] ** B2[z] ** inv[e1[z]] ** tp[B2[z]] ** XX[z]/2 -
   tp[GEz[x, z]] ** B2[z] ** inv[e1[z]] ** tp[D12[z]] ** C1[z]/2 +
   tp[XX[z]] ** B1[z] ** tp[D21[z]] ** inv[e2[x]] ** C2[x] -
   tp[XX[z]] ** B1[z] ** tp[D21[z]] ** inv[e2[x]] ** C2[z] -
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[B2[x]] ** GEx[x, z]/2 -
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[B2[z]] ** GEz[x, z]/2 -
   tp[XX[z]] ** B2[z] ** inv[e1[z]] ** tp[D12[x]] ** C1[x] +
   tp[C1[z]] ** D12[z] ** inv[e1[z]] ** e1[x] ** inv[e1[z]] ** tp[B2[z]] **
    XX[z] + tp[C1[z]] ** D12[z] ** inv[e1[z]] ** e1[x] ** inv[e1[z]] **
    tp[D12[z]] ** C1[z] - tp[GEx[x, z]] ** B1[x] ** tp[D21[x]] **
     inv[e2[x]] ** D21[x] ** tp[B1[x]] ** GEx[x, z]/4 +
   tp[GEx[x, z]] ** B1[x] ** tp[D21[x]] ** inv[e2[x]] ** D21[z] **
     tp[B1[z]] ** XX[z]/2 + tp[XX[z]] ** B1[z] ** tp[D21[z]] ** inv[e2[x]] **
     D21[x] ** tp[B1[x]] ** GEx[x, z]/2 -
   tp[XX[z]] ** B1[z] ** tp[D21[z]] ** inv[e2[x]] ** D21[z] ** tp[B1[z]] **
    XX[z] + tp[XX[z]] ** B2[z] ** inv[e1[z]] ** e1[x] ** inv[e1[z]] **
```

```
    tp[B2[z]] ** XX[z] + tp[XX[z]] ** B2[z] ** inv[e1[z]] ** e1[x] **
    inv[e1[z]] ** tp[D12[z]] ** C1[z]

In[29]:= K = NCC[NCC[K,inv[e2[x]]],inv[e1[z]]]

Out[29]= tp[A[x]] ** GEx[x, z]/2 + tp[A[z]] ** GEz[x, z]/2 +
   tp[C1[x]] ** C1[x] + tp[GEx[x, z]] ** A[x]/2 + tp[GEz[x, z]] ** A[z]/2 +
   (tp[GEx[x, z]] ** B2[x] + tp[GEz[x, z]] ** B2[z]) ** inv[e1[z]] **
    (-tp[B2[z]] ** XX[z]/2 - tp[D12[z]] ** C1[z]/2) +
   (tp[C1[z]] ** D12[z] + tp[XX[z]] ** B2[z]) ** inv[e1[z]] **
    (-tp[B2[x]] ** GEx[x, z]/2 - tp[B2[z]] ** GEz[x, z]/2 -
      tp[D12[x]] ** C1[x]) + tp[C2[x]] ** inv[e2[x]] **
    (-C2[x] + C2[z] - D21[x] ** tp[B1[x]] ** GEx[x, z]/2 +
      D21[z] ** tp[B1[z]] ** XX[z]) +
   (tp[XX[z]] ** B1[z] ** tp[D21[z]] + tp[C2[z]]) ** inv[e2[x]] **
    (C2[x] - C2[z] + D21[x] ** tp[B1[x]] ** GEx[x, z]/2 -
      D21[z] ** tp[B1[z]] ** XX[z]) +
   tp[C1[x]] ** D12[x] ** inv[e1[z]] **
    (-tp[B2[z]] ** XX[z] - tp[D12[z]] ** C1[z]) +
   tp[GEx[x, z]] ** B1[x] ** tp[B1[x]] ** GEx[x, z]/4 +
   tp[GEz[x, z]] ** B1[z] ** tp[B1[z]] ** XX[z]/2 +
   tp[XX[z]] ** B1[z] ** tp[B1[z]] ** GEz[x, z]/2 +
   (tp[C1[z]] ** D12[z] + tp[XX[z]] ** B2[z]) ** inv[e1[z]] ** e1[x] **
    inv[e1[z]] ** (tp[B2[z]] ** XX[z] + tp[D12[z]] ** C1[z]) +
   tp[GEx[x, z]] ** B1[x] ** tp[D21[x]] ** inv[e2[x]] **
    (-C2[x]/2 + C2[z]/2 - D21[x] ** tp[B1[x]] ** GEx[x, z]/4 +
      D21[z] ** tp[B1[z]] ** XX[z]/2)

In[31]:= qpart = NCE[%19-(%19//.q[x,z]->0)]

Out[31]= q[x, z] ** C2[x]/2 - q[x, z] ** C2[z]/2 +
   tp[C2[x]] ** tp[q[x, z]]/2 - tp[C2[z]] ** tp[q[x, z]]/2 +
   q[x, z] ** e2[x] ** tp[q[x, z]]/4 +
   q[x, z] ** D21[x] ** tp[B1[x]] ** GEx[x, z]/4 -
   q[x, z] ** D21[z] ** tp[B1[z]] ** XX[z]/2 +
   tp[GEx[x, z]] ** B1[x] ** tp[D21[x]] ** tp[q[x, z]]/4 -
   tp[XX[z]] ** B1[z] ** tp[D21[z]] ** tp[q[x, z]]/2

In[32]:= %//.tp[q[x,z]]->0

Out[32]= q[x, z] ** C2[x]/2 - q[x, z] ** C2[z]/2 +
   q[x, z] ** D21[x] ** tp[B1[x]] ** GEx[x, z]/4 -
   q[x, z] ** D21[z] ** tp[B1[z]] ** XX[z]/2

In[33]:= L = %//.q[x,z]->1

Out[33]= C2[x]/2 - C2[z]/2 + D21[x] ** tp[B1[x]] ** GEx[x, z]/4 -
   D21[z] ** tp[B1[z]] ** XX[z]/2

In[34]:= Q = e2[x]/4
```

```
Out[34]= e2[x]/4

In[35]:= bterm = q[x,z]+tp[L]**inv[Q]

Out[35]= 4*(tp[GEx[x, z]] ** B1[x] ** tp[D21[x]]/4 -
       tp[XX[z]] ** B1[z] ** tp[D21[z]]/2 + tp[C2[x]]/2 - tp[C2[z]]/2) **
     inv[e2[x]] + q[x, z]

In[36]:= Sub[NCE[%19-bterm**Q**tp[bterm]-K],rue]

Out[36]= 0

In[39]:= Quit
```

5.3 SPECIALIZING TO LESS GENERAL SYSTEMS

5.3.1 SPECIALIZING TO LINEAR SYSTEMS

To specialize to the linear case, just apply

- rulinearsys to make the systems linear
- rulinearEB to make the energy function quadratic
- ruGE1 then ruGEXY to make the energy function solve the Hinf problem (max entropy soln.)
- **rulinearall** contains all of the rules above and is what we usually use.

(SEE GLOSSARY)

5.3.2 SPECIALIZING USING THE DOYLE GLOVER KHARGONEKAR FRANCIS SIMPLIFYING ASSUMPTIONS

A special class of IA systems are those satisfying

$$D12(x)^T C1(x) = 0 \quad \text{and} \quad B1(x) D21(x)^T = 0$$

denoted in this paper as the Doyle-Glover-Kargonekar-Francis (DGKF) simplifying assumptions (see [DGKF]). These simplify algebra substantially so are good for tutorial purposes even though they are not satisfied in actual control problems. Look for rules ruDGKF*.

(SEE GLOSSARY)

5.3.3 DEMO: LINEAR DOYLE GLOVER KARGONEKAR FRANCIS EQUATIONS

The following demo verifies IAX and IAYI are same as DGX DGYI the Doyle Glover X and inv[Y] Riccati equations in the special case of a linear system.

```
In[25]:= NCE[IAX[x]//.rulinearall]

Out[25]= tp[x] ** XX ** A ** x + tp[x] ** tp[A] ** tp[x] ** XX +
   tp[x] ** tp[C1] ** C1 ** x + tp[x] ** XX ** B1 ** tp[B1] ** tp[x] ** XX -
   tp[x] ** XX ** B2 ** inv[e1] ** tp[B2] ** tp[x] ** XX -
   tp[x] ** XX ** B2 ** inv[e1] ** tp[D12] ** C1 ** x -
   tp[x] ** tp[C1] ** D12 ** inv[e1] ** tp[B2] ** tp[x] ** XX -
   tp[x] ** tp[C1] ** D12 ** inv[e1] ** tp[D12] ** C1 ** x

In[26]:= Sub[%,x->1]

Out[26]= XX ** A + tp[A] ** XX + tp[C1] ** C1 + XX ** B1 ** tp[B1] ** XX -
   XX ** B2 ** inv[e1] ** tp[B2] ** XX -
   XX ** B2 ** inv[e1] ** tp[D12] ** C1 -
   tp[C1] ** D12 ** inv[e1] ** tp[B2] ** XX -
   tp[C1] ** D12 ** inv[e1] ** tp[D12] ** C1

In[27]:= NCE[%-DGX]

Out[27]= 0

In[28]:= NCE[IAYI[x]//.rulinearall]

Out[28]= tp[x] ** inv[YY] ** A ** x + tp[x] ** tp[A] ** tp[x] ** inv[YY] +
   tp[x] ** tp[C1] ** C1 ** x - tp[x] ** tp[C2] ** inv[e2] ** C2 ** x +
   tp[x] ** inv[YY] ** B1 ** tp[B1] ** tp[x] ** inv[YY] -
   tp[x] ** inv[YY] ** B1 ** tp[D21] ** inv[e2] ** C2 ** x -
   tp[x] ** tp[C2] ** inv[e2] ** D21 ** tp[B1] ** tp[x] ** inv[YY] -
    tp[x] ** inv[YY] ** B1 ** tp[D21] ** inv[e2] ** D21 ** tp[B1] ** tp[x] **
    inv[YY]

In[29]:= Sub[%,x->1]

Out[29]= inv[YY] ** A + tp[A] ** inv[YY] + tp[C1] ** C1 -
   tp[C2] ** inv[e2] ** C2 + inv[YY] ** B1 ** tp[B1] ** inv[YY] -
   inv[YY] ** B1 ** tp[D21] ** inv[e2] ** C2 -
   tp[C2] ** inv[e2] ** D21 ** tp[B1] ** inv[YY] -
   inv[YY] ** B1 ** tp[D21] ** inv[e2] ** D21 ** tp[B1] ** inv[YY]

In[30]:= NCE[YY**%**YY-DGY]

Out[32]= 0

In[34]:= Quit
```

Chapter 6

REFERENCES

[AV] B. D. O. Anderson and S. Vongpanitlerd, Network Analysis and Synthesis, Prentice Hall, Englewood Cliffs, 1973.

[BHW] J. A. Ball, J. W. Helton, and M. L. Walker, IEEE Trans Aut Contrl April 1993, originally the computer file SYSHinfTAC.m included here.

[BH5] J. A. Ball and J. W. Helton, Nonlinear H^∞ control theory for stable plants, Math. Control Signals Systems 5 (1992), pp. 233-261.

[DGKF] J. C. Doyle, K. Glover, P. P. Khargonekar and B. A. Francis, State–space solutions to standard H_2 and H_∞ control problems, IEEE Trans. Auto. Control 34 (1989), 831–847.

[HM] D. J. Hill and P. J. Moylan, "Dissipative Dynamical Systems: Basic Input-Output and State Properties", J. Franklin Inst. v.309, No.5, May 1980, pp.327-357

[HWZ] J.W. Helton, M.L. Walker, W. Zhan, Compensators which have access to the command signal, preprint

[IA] A. Isidori and A. Astolfi, Nonlinear H^∞ control via measurement feedback, J. Math. Systems, Estimation and Control 2 (1992), pp. 31-34

[IB] A. Isidori and C. I. Byrnes, "Output Regulation of Nonlinear Systems", IEEE Trans. Aut. Contr., V.35, No.2, Feb.1990, pp. 131-140

[PAJ] I. R. Petersen, B. D. O. Anderson and E. A. Jonckheere, A First Principles Solution to the Non-Singular H^∞ Control Problem, Int. J. Robust and Nonlinear Control, v.1, 1991, pp.171-185

[V2] A. J. Van der Schaft, L_2-gain analysis of nonlinear systems and nonlinear H_∞ control, IEEE Trans. Auto. Control 37 (1992), pp.770-784.

[W] J. C. Willems, Dissipative Dynamical Systems, Part I: general theory, Arch. Rat. Mech. Anal. 45 (1972), 321-351.

Chapter 7

GLOSSARY FOR SYSTEM HAMILTONIAN CALCULATIONS

This glossary contains expressions available for manipulation, functions available for evaluation, and substitution rules all prefaced by ru.

HAMILTONIANS

```
These formulas contain mixed state and dual variables (IA systems, sec 5.2):
HWUY  = Hamiltonian for the system in SYSDefIA.m .
HWU = HWUY/.Y->G2[x,W,U];
HWY = HWUY/.U->g[z,Y];
HW = HWY/.Y->G2[x,W,U];

These formulas contain state variables only (IA systems):
sHWUY = HWUY/.ruxz;
sHWU = HWU/.ruxz;
sHWY = HWY/.ruxz;
sHW = HW/.ruxz;

These formulas contain dual variables only (IA systems):
dHWUY = HWUY//.rudual;
dHWU = HWU//.rudual;
dHWY = HWY//.rudual;
dHW = HW//.rudual;
```

EXTREMIZING HAMILTONIANS

```
ricd = Riccati from bounded real lemma for SISO systems (sec 5.1)

CritW[x_,z_,b_] = value of W which makes Grad[sHW,W]=0(both IA and WIA systems).
```

```
sHWo  = max sHW (IA systems)
            W

sHWoWIA  = max sHW (WIA systems)
               W

CritU[x_,z_] = value of U which makes Grad[sHWU,U]=0 (IA systems).

RULES FOR CRITICAL VALUES:

ruCritW = {W -> tp[B1[x]] ** tp[GEx[x, z]] +
                tp[D21[x]] ** tp[b[z]] ** tp[GEz[x, z]]};

ruCritU = {U -> -inv[e1[x]] ** tp[B2[x]] ** tp[GEx[x, z]] -
                             inv[e1[x]] ** tp[D12[x]] ** C1[x]}
```

NONLINEAR RICCATIS

THE DGKF RICCATIS

```
IAX[x_] = XX[x] ** (A[x] - B2[x] ** inv[e1[x]] ** tp[D12[x]] ** C1[x])+
   (-tp[C1[x]] ** D12[x] ** inv[e1[x]] ** tp[B2[x]] + tp[A[x]]) **
   tp[XX[x]] + tp[C1[x]] ** C1[x] +
    XX[x] ** (B1[x] ** tp[B1[x]] - B2[x] ** inv[e1[x]] ** tp[B2[x]]) **
    tp[XX[x]] - tp[C1[x]] ** D12[x] ** inv[e1[x]] ** tp[D12[x]] ** C1[x]

IAYI[x_] = YYI[x] ** (A[x] - B1[x] ** tp[D21[x]] ** inv[e2[x]]**C2[x]) +
(-tp[C2[x]]**inv[e2[x]]**D21[x]**tp[B1[x]] + tp[A[x]])**tp[YYI[x]] +
      tp[C1[x]] ** C1[x] + YYI[x] **
       (B1[x]** tp[B1[x]] - B1[x]**tp[D21[x]] ** inv[e2[x]] ** D21[x] **
          tp[B1[x]]) ** tp[YYI[x]] - tp[C2[x]] ** inv[e2[x]] ** C2[x]


k[x_,z_]   = minimum in b of sHWo evaluated at a* and c* (see [BHW])
                  This is the 2 variable generalization of DGKF Riccati.


Hopt[x,z,a,b]  = min max sHWU, where you can specify functions a[z],b[z]
                  U   W
```

THE CENTRAL CONTROLLER

```
    ruc = c[z_]:> -inv[e1[z]]**(tp[B2[z]]**XX[z] + tp[D12[z]]**C1[z]);
    rua = a[z_]:> A[z] + B2[z]**c[z] - b[z]**C2[z] +
                         (B1[z]- b[z]**D21[z])**tp[B1[z]]**XX[z]//.ruc ;
    rublin= the rule in the linear case for the b in the central soln
                to the H-infinity control problem.
```

```
ruaWIA = a[z_]:> AB[z,c[z]] - b[z]**C2[z] +
                        (B1[z]- b[z]**D21[z])**tp[B1[z]]**XX[z];
```

```
Plug rua and ruc into the main Hamiltonian sHWo to get
     sHWo= k[x,z]  +  tp[bterm]**bterm
        which defines bterm.
```

SPECIAL SYSTEMS

LINEAR SYSTEMS

Linear systems satisfying the DGKF simplifying assumption are special cases of an IA system. To take an expression in H^{∞} control which has been derived for an IA system and specialize it to linear systems apply

```
SubSym[expr,rulinearall]
```

where rulinearall is a long list of rules for specializing IA systems formulas described below:

```
ruGEXY={ruGEXY1,ruGEXY2};
                      where ruGEXY1=tp[GEx[x_,0]]:>tp[x]**inv[YY];
                            ruGEXY2=tp[GEx[x_,x_]]:>tp[x]**XX;

rulinearall=Union[rulinearsys,ruGE1, ruGEXY,rulinearXY];
              where rulinearsys={A[x_]:>A**x, B1[x_]:>B1,
                                          B2[x_]:>B2,C1[x_]:>C1**x,C2[x_]:>C2**x,
                                  D21[x_]:>D21,D12[x_]:>D12,a[x_]:>a**x,
                                          b[x_]:>b,c[x_]:>c**x,dd[x_]:>dd,
                                          e1[x_]:>e1,
                                          e2[x_]:>e2,tp[e1]->e1,tp[e2]->e2};
                    rulinearXY={XX[x_]:>XX**x,YY[p_]:>p**YY,
                                                YYI[x_]:>inv[YY]**x};

                            ruinvYY=YYI->inv[YY];
```

```
Energy Ansatzes  which  are true for linear systems:

ruGEz0=GEz[x_,x_]:>0


 Ansatz in [BHW] %(5.1):

ruGE1=Flatten[{ruGEz0,SubSym[{ruGEz0,ruGEx1,ruGEz1},ruXXYYI]}];
             where ruGEx1=GEx[x_,z_]:>GEx[x,x]+GEx[x-z,0]-GEx[x-z,x-z];
                   ruGEz1=GEz[x_,z_]:>-GEx[x-z,0]+GEx[x-z,x-z];
```

HOMOGENEOUS SYSTEMS:

```
ruhomog = {A[0]->0, C1[0]->0, C2[0]->0, a[0]->0, c[0]->0};
```

DOYLE GLOVER KHARGONEKAR FRANCIS SIMPLIFYING ASSUMPTIONS:

```
ruDGKFNL = Apply DGKF simplifying assumptions for IA systems

ruDGKFlin = Apply DGKF simplifying assumptions for linear systems

ruCRcDGKF=c[z_]:>-tp[B2[z]]**tp[XX[z]];
ruCRqDGKF=q[x_,z_]:>-tp[C2[x-z]];

ruaDGKF = a[z_]:>A[z]+B1[z]**tp[B1[z]]**tp[XX[z]] -
               B2[z]**tp[B2[z]]**tp[XX[z]] - b[z]**C2[z]+cheat[z];

ruaDGKF2=a[z_]:> AA1[z,z]-b[z]**C2[z];
```

MORE RULES AVAILABLE FOR SUBSTITUTION

```
CHANGE OF VARIABLES:
Convert energy Hamiltonians from dual or mixed state/dual to state
variables only:
ruxz={p->GEx[x,z], PP->GEz[x,z]};

Convert energy Hamiltonian from dual or mixed state/dual to dual variables only:
rudual={rudualx,rudualz};
                              where rudualx = x->IGE1[p,PP];
                                    rudualy = z->IGE2[p,PP];

CHANGE OF NOTATION:
rue = {tp[D12[x_]] ** D12[x_]:> e1[x], D21[x_] ** tp[D21[x_]] :> e2[x]}
rubtoq=tp[GEz[x_,z_]]**b[z_]:>q[x,z]

ruXXYYI={ruGEz0,ruXX, ruYYI}
                      where   ruXX = GEx[x_,x_]:>2*XX[x]
                              ruYYI = GEx[x_,0]:>2*YYI[x]

ruIAXYI={ruIAX,ruIAYI};
                      where ruIAX = replace terms linear in A or tp[A]
                                          using IAX[x]=0 (see 4.4b).
                            ruIAYI= replace terms linear in A or tp[A]
                                          using IAYI[x]=0 (see 4.4c).
```

```
FOR ENERGY ANSATZ EXPERIMENTATION AS IN [BHW]:

    EnergyGuess[ruleGE_] = generate sHWo with the Ansatz on the energy function
   given by ruleGE and a, b, and c are substituted out
   according to the RECIPE.
    EnergyGuess2[ruleGE,rulea] = same but a is substituted out by the
                                 user provided rulea.

    EnergyGuessDGKF[ruleGE_] = same with DGKF simplifying assumptions
    EnergyGuess2DGKF[ruleGE,ruaDGKF] = same with DGKF simplifying assumptions
```