DINESH KRISHNAMOORTHY

Project for KP8115
Advanced Process Control.

Changes made to cola_commands.m

`G = pck(A,B,C,D); Gd = pck(A,Bd,C,D);`

Returns G with class double. This is problematic in the functions ozde.m and izde.m, where the system is checked whether the class of the system is of type `'tf'|'ss'|'zpk'|'frd'`

To rectify this, I modified the G to be a state space model using the ss function,

`GSS = ss(A,B,C,D);` instead of `G = pck(A,B,C,D);`

GSS is now given as the input argument to the functions ozde.m and izde.m

In the functions ozde.m and izde.m

The command `[ny,nu,nx]=size(sys);`

Does not provide the right dimensions of the system. Probably due to the fact that now the system given as input argument to this function is now a state space system rather than a packed system.

Therefore, replaced this line with getting the no of states nx, no of inputs nu and the no of outputs ny explicitly from the system.

```
[nx,nx] = size(sys.A);[nx,nu] = size(sys.B);[ny,nx] = size(sys.C);
```

The logic to verify that the eigenvalues are finite, the function finite does not work anymore, instead replaced it with isfinite which returns 1 if the argument is finite and 0 otherwise.

Similar changes were also made in izde.m function.

The changes are highlighted in the code attached.

```
% This is file cola_commands.m
% It contains are some useful commands for MIMO controllability
 analysis:
% Poles/zeros and their directions, RGA, PRGA, CLDG, singular values
% Uses the Mu-toolbox.

% Start from G and Gd:

% start model:
% Here: Distillation column model with 5 states and disturbances
% Scaled LV-model (Example 10.10 in book)
% The two inputs are (LV);
% the two disturbances are (F,zF)
% The two outputs are (yd,xb)

addpath 'M:\Courses\KP8115_Advanced Process Control\Project\matlab_m'
A = [ -5.131e-3 0 0 0 0; 0 -7.366e-2 0 0 0; 0 0 -1.829e-1 0 0;
       0 0 0 -4.620e-1  9.895e-1; 0 0 0 -9.895e-1 -4.620e-1]
B = [-.629 .624; .055 -0.172; 0.030 -0.108;
      -0.186 -0.139; -1.23 -0.056]
C= [-0.7223 -0.5170 0.3386 -0.1633e-1 0.1121;
    -0.8913 0.4728 0.9876 0.8425 0.2186]
D = [ 0 0; 0 0]
Bd = [-0.062 -0.067; 0.131 0.040; 0.022 -0.106;
     -0.188 0.027; -0.045 0.014]
G = pck(A,B,C,D); Gd = pck(A,Bd,C,D);
GSS = ss(A,B,C,D);
% end model
clc
% We now have G and Gd .....  Start analysis.
format short e

spoles(G)                         % Compute poles   (Distillation:
 "slow" is at 0.0052=1/194 min)
[A,B,C,D]=unpck(G); poles=eig(A) %   should be the same ...
[T,Po] = eig(A);   YP = C*T      %   pole output vectors (to obtain
 directions normalize columns to 1)
[Q,Pi] = eig(A'); Q=conj(Q);
UP = B'*Q                        %   pole input vectors
Shouldbezero=Po-Pi;               % if nonzero: WARNING - ordering of
 eigenvalue differs in YP and UP
if norm(Shouldbezero)>1.e-7 disp('WARNING: Eigenvalue order differs
 for inputs and output pole vectors'),
   disp ('Use opde and ipde instead:  [Po, Yp, Xpo, Spo] = opde(G),
 [Pi, Up, Xpi, Spi] = ipde(G)'); end

Szeros=szeros(G)                  % zeros (the ones "far away" are not
 important)
Tzero=tzero(A,B,C,D)            %   should be the same ... (If they
 differ I would trust tzero)
[Zy,YZ,Xy] = ozde(GSS); Zy, YZ     %   zero output directions
[Zu,UZ,Xu] = izde(GSS); Zu, UZ     %   zero input directions
```

```
G0 = frsp(G,0)                    % steady-state plant gains
G00= -C*inv(A)*B + D              %   should be the same ...
RGA0 = G00.*pinv(G00.')           % steady-state RGA
vrga(G0)                          %   should be the same ... (using
 subroutine vrga)
PRGA0 = mmult(vdiag(vdiag(G0)),minv(G0)) % Performance RGA
Gd0= frsp(Gd,0)                   % steady-state disturbance gains
CLDG0 = mmult(PRGA0,Gd0)          % Closed-loop disturbance gain
GinvGd0 = mmult(minv(G0),Gd0)     % inputs for perfect disturbance
 rejection

w = logspace(-3,1,41);            % Now do the same as a function of
 frequency
Gf = frsp(G,w); Gdf=frsp(Gd,w);   % Frequency repsonse of G and Gd

figure()
vplot('liv,lm',vsvd(Gf),1,':');   % Plot singular values of G
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black';

figure()
vplot('liv,lm',Gdf,1,':');        % Plot elements in Gd
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black';

figure()
vplot('liv,lm',vrga(Gf),1,':');   % RGA-elements
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black';

figure()
gdiag=vdiag(vdiag(Gf)); prga = mmult(gdiag,minv(Gf));
vplot('liv,lm',prga,1,':');       % PRGA-elements
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black';
cldg=mmult(prga,Gdf);

figure()
vplot('liv,lm',cldg,1,':');       % CLDG-elements
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black';
% Some closed-loop analysis ..........
```

```
% Two simple PI controllers
k1 = nd2sys([3.76 1], [3.76 1.e-4],  0.261);
k2 = nd2sys([3.31 1], [3.31 1.e-4], -0.375);
K = daug(k1,k2); GK = mmult(G,K);
S = minv(madd(eye(2),GK));                % sensitivity function
spoles(S)                                 % closed-loop poles (stable)
Sf=frsp(S,w);

figure()
vplot('liv,lm',Sf,1,':'); % sensitivity function elements
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black';

figure()
vplot('liv,lm',vsvd(Sf),1,':');           % singular values of
 sensitivity
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black';                                    % (which is less
 than 1 as desired)

SGd = mmult(S,Gd); SGdf=frsp(SGd,w);

figure()
vplot('liv,lm',vsvd(SGdf),1,':');  % closed-loop effect of
 disturbances
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black';                                     % (which is
 less than 1 as desired)


y = trsp(SGd,[1 0]',100,0.2); vplot(y);
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black';                                    % (which is less
 than 1 as desired)
% Time response to disturbance in F

y = trsp(SGd,[0 1]',100,0.2); vplot(y);
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black';                                     % (which is
 less than 1 as desired)
% Time response to disturbance in zF


A =
```

```
  -5.1310e-03            0            0            0            0
            0  -7.3660e-02            0            0            0
            0            0  -1.8290e-01            0            0
            0            0            0  -4.6200e-01   9.8950e-01
            0            0            0  -9.8950e-01  -4.6200e-01


B =

  -6.2900e-01   6.2400e-01
   5.5000e-02  -1.7200e-01
   3.0000e-02  -1.0800e-01
  -1.8600e-01  -1.3900e-01
  -1.2300e+00  -5.6000e-02


C =

  -7.2230e-01  -5.1700e-01   3.3860e-01  -1.6330e-02   1.1210e-01
  -8.9130e-01   4.7280e-01   9.8760e-01   8.4250e-01   2.1860e-01


D =

     0     0
     0     0
```

System G represented in state space

System Gd which has the same A, C and D matrices as G.

```
Bd =

  -6.2000e-02  -6.7000e-02
   1.3100e-01   4.0000e-02
   2.2000e-02  -1.0600e-01
  -1.8800e-01   2.7000e-02
  -4.5000e-02   1.4000e-02
```

spoles(G)

```
ans =

  -5.1310e-03 + 0.0000e+00i
  -7.3660e-02 + 0.0000e+00i
  -1.8290e-01 + 0.0000e+00i
  -4.6200e-01 + 9.8950e-01i
  -4.6200e-01 - 9.8950e-01i
```

These two are the same, as expected

```
poles =

  -4.6200e-01 + 9.8950e-01i
  -4.6200e-01 - 9.8950e-01i
  -5.1310e-03 + 0.0000e+00i
  -7.3660e-02 + 0.0000e+00i
  -1.8290e-01 + 0.0000e+00i
```

Output pole vectors

```
YP =

  Columns 1 through 2

   1.1547e-02 - 7.9267e-02i   1.1547e-02 + 7.9267e-02i
  -5.9574e-01 - 1.5457e-01i  -5.9574e-01 + 1.5457e-01i

  Columns 3 through 4

  -7.2230e-01 + 0.0000e+00i  -5.1700e-01 + 0.0000e+00i
  -8.9130e-01 + 0.0000e+00i   4.7280e-01 + 0.0000e+00i

  Column 5

   3.3860e-01 + 0.0000e+00i
   9.8760e-01 + 0.0000e+00i
```

Input pole vectors

```
UP =

  Columns 1 through 2

  -1.3152e-01 - 8.6974e-01i  -1.3152e-01 + 8.6974e-01i
  -9.8288e-02 - 3.9598e-02i  -9.8288e-02 + 3.9598e-02i

  Columns 3 through 4

  -6.2900e-01 + 0.0000e+00i   5.5000e-02 + 0.0000e+00i
   6.2400e-01 + 0.0000e+00i  -1.7200e-01 + 0.0000e+00i

  Column 5

   3.0000e-02 + 0.0000e+00i
  -1.0800e-01 + 0.0000e+00i


Szeros =

  -2.8614e+00
  -2.7459e-01
  -1.5163e-01
```

These two are the same

```
Tzero =

  -2.8614e+00
  -2.7459e-01
  -1.5163e-01


Zy =
```

```
-1.3892e+15
-2.8614e+00
-1.5163e-01
-2.7459e-01
```

have one zero extra, but the ones far away are not important?

these three are the same as before

Output zero vectors

```
YZ =

  Columns 1 through 2

   8.3367e-01 - 1.0209e-16i   9.0620e-01 - 1.1098e-16i
  -5.5227e-01 + 6.7633e-17i  -4.2286e-01 + 5.1785e-17i

  Columns 3 through 4

   8.6970e-01 + 0.0000e+00i   9.0925e-01 + 0.0000e+00i
  -4.9357e-01 + 0.0000e+00i  -4.1625e-01 + 0.0000e+00i


Zu =

  -2.8614e+00
  -2.7459e-01
  -1.5163e-01
```

these three are the same as above

Input zero vectors

```
UZ =

   8.2399e-01   7.0796e-01   3.0780e-01
   5.6661e-01   7.0625e-01   9.5145e-01


G0 =

   8.8197e+01  -8.6822e+01            0
   1.0879e+02  -1.1015e+02            0
            0   1.0000e+00          Inf


G00 =

   8.8197e+01  -8.6822e+01
   1.0879e+02  -1.1015e+02
```

same as expected

```
RGA0 =

   3.6066e+01  -3.5066e+01
  -3.5066e+01   3.6066e+01
```

diagonal pairing preferred, due to positive RGA elements.

vrga(G0)

```
ans =

   3.6066e+01  -3.5066e+01            0
```

gives the same RGA matrix as above

```
  -3.5066e+01   3.6066e+01            0
            0   1.0000e+00          Inf


PRGA0 =

   3.6066e+01  -2.8429e+01            0
  -4.4485e+01   3.6066e+01            0
            0   1.0000e+00          Inf
```

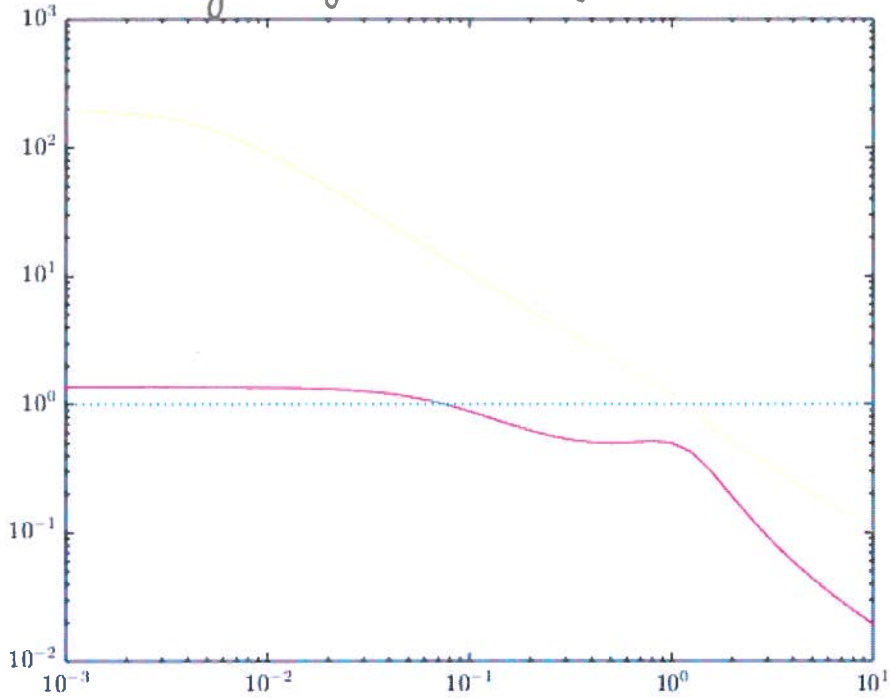
diagonal elements of PRGA same as RGA as expected

```
Gd0 =

   7.8665e+00   8.9525e+00            0
   1.1667e+01   1.1338e+01            0
            0   1.0000e+00          Inf
```

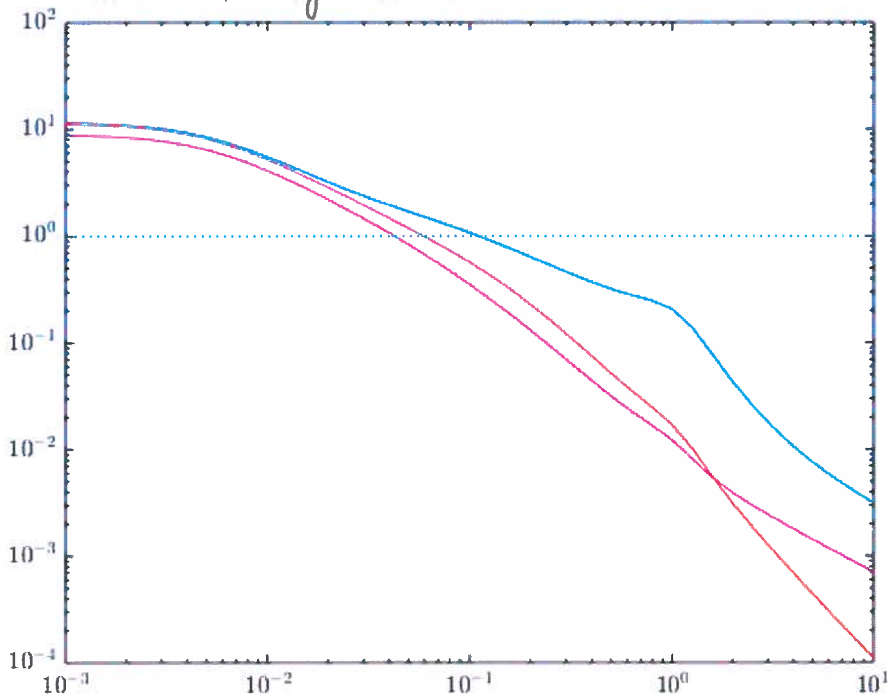
closed loop disturbance gain.

```
CLDG0 =

  -4.7971e+01   5.5809e-01            0
   7.0837e+01   1.0649e+01            0
            0   1.0000e+00          Inf


GinvGd0 =

  -5.4391e-01   6.3277e-03            0
  -6.4312e-01  -9.6685e-02            0
            0   1.0000e+00          Inf


ans =

  -4.4989e-01 + 9.9120e-01i
  -4.4989e-01 - 9.9120e-01i
  -1.9997e-01 + 2.7488e-01i
  -1.9997e-01 - 2.7488e-01i
  -6.1451e-02 + 6.1312e-02i
  -6.1451e-02 - 6.1312e-02i
  -1.6931e-01 + 0.0000e+00i

interpolating input vector (zero order hold)
interpolating input vector (zero order hold)
```

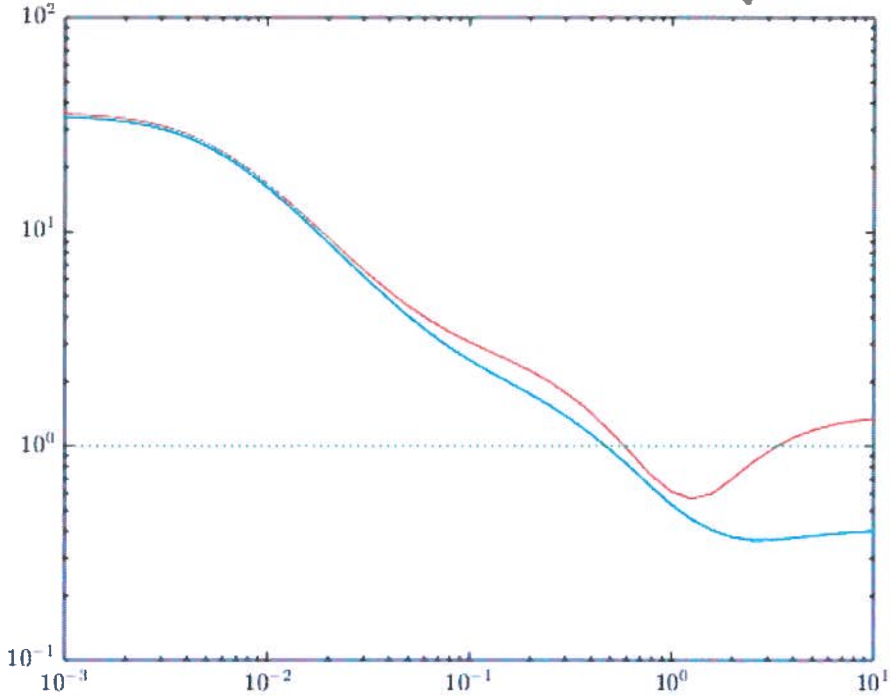
Plotting singular values of G



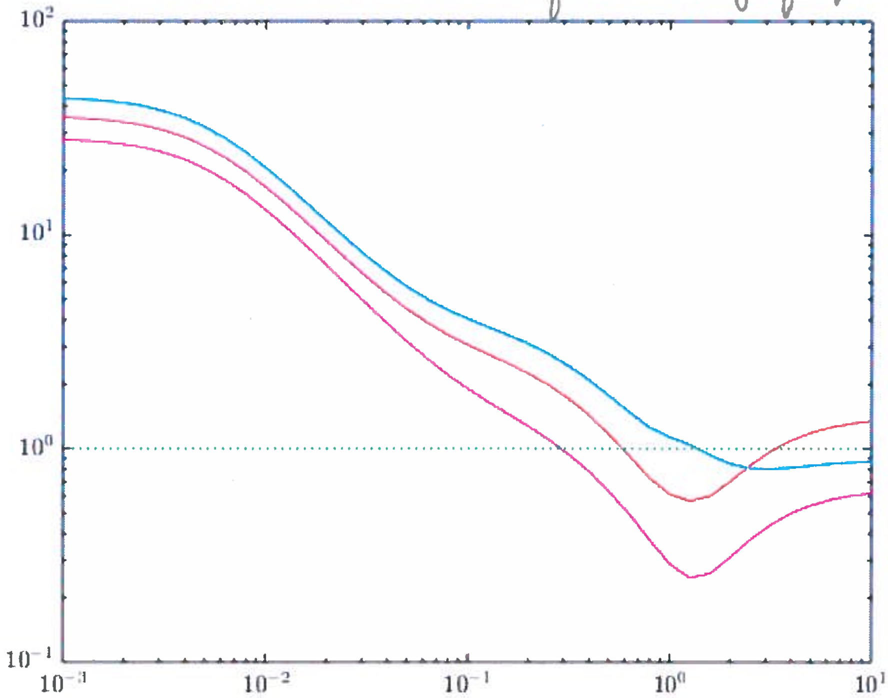
Plotting elements in $G_d(\omega)$



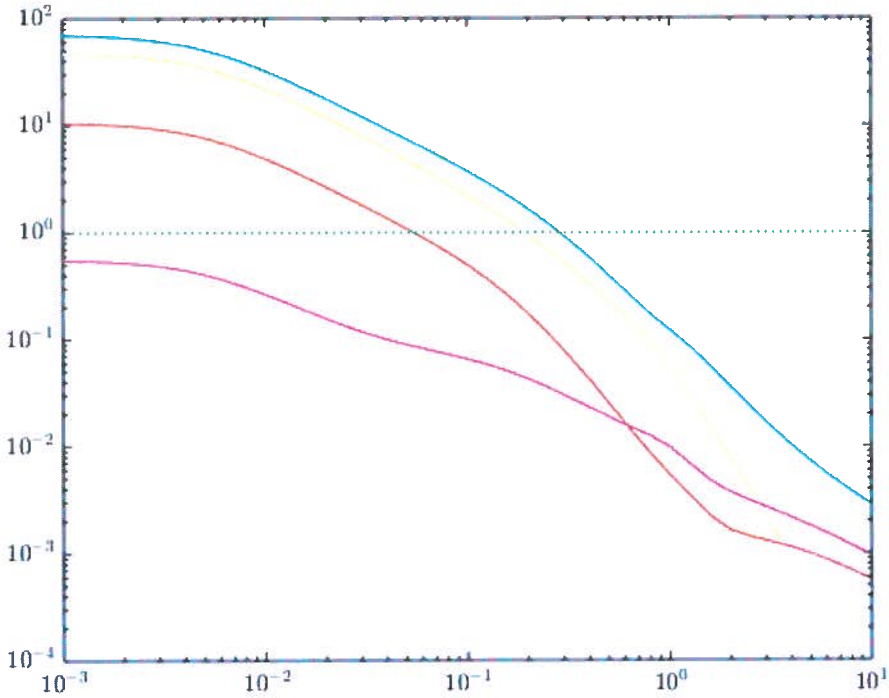
RGA as a function of frequency , $\Lambda(G(j\omega))$



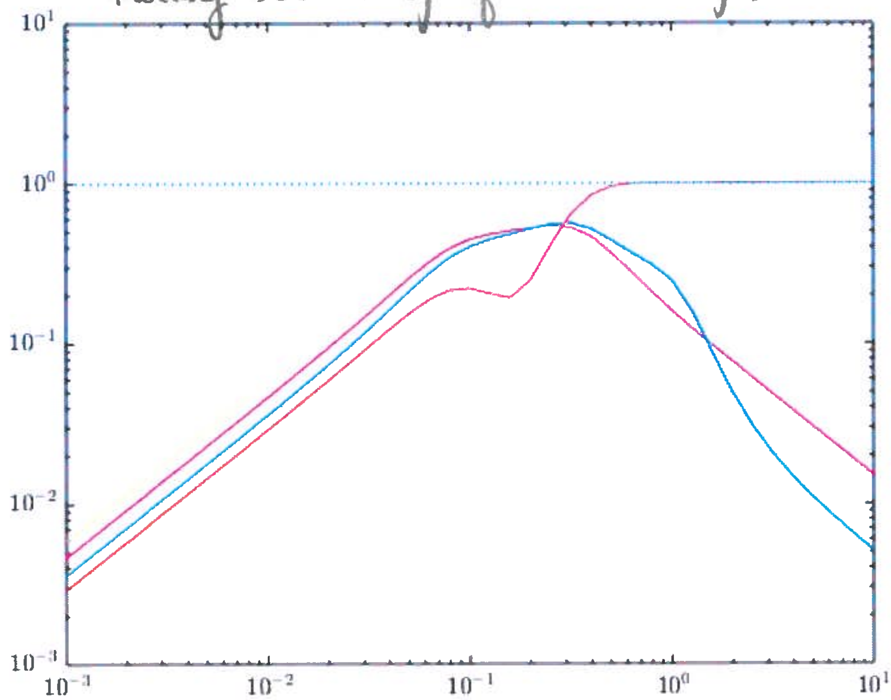
PRGA elements as a function of frequency



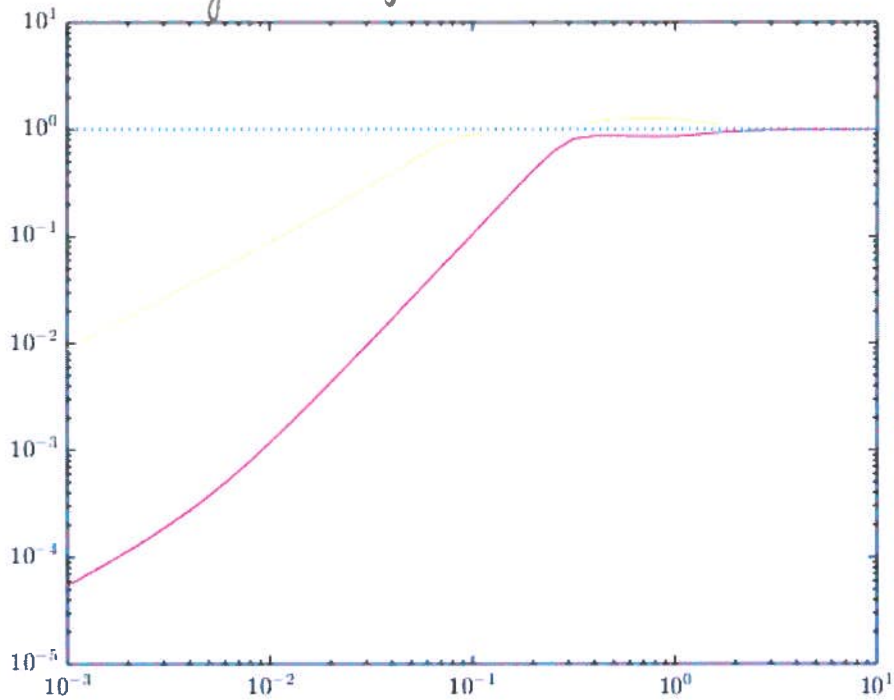
CLDG elements as a function of frequency



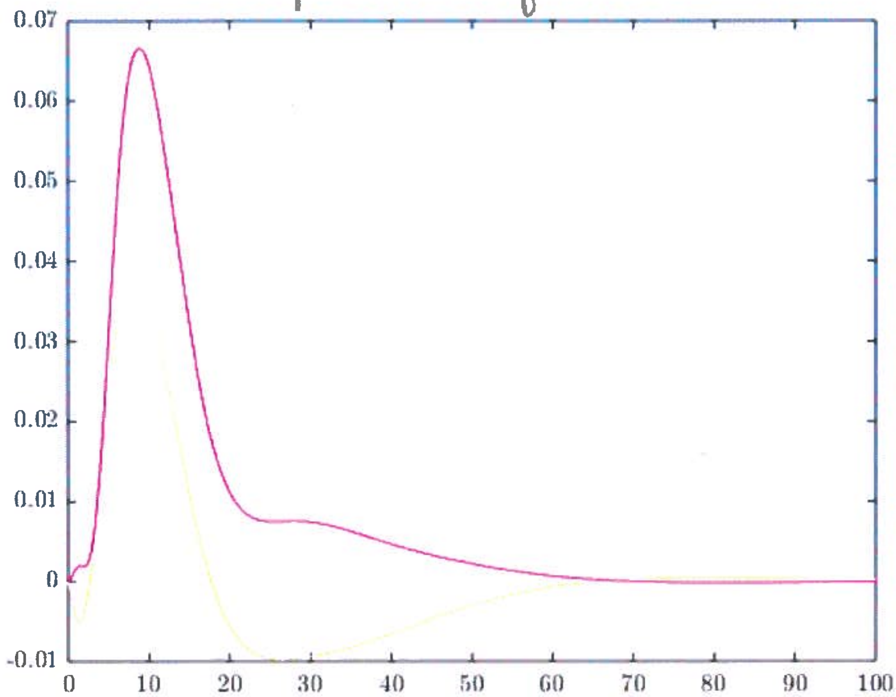
Plotting sensitivity function $S(j\omega)$



Plotting $\sigma_i(S(j\omega))$



closed loop effect of disturbances



which is less than 1 as desired

Published with MATLAB® R2016a

```
% function [Z, Y, X] = ozde(G,epp)
%
%   Inputs:  G   - system matrix in mu-tools format.
%            EPP - tolerance, see below, default value EPS.
%   Outputs: Z   - zeros.
%            Y   - output zero directions, stored as column vectors.
%            X   - state directions, stored as column vectors.
%            Each column X(:,i) and U(:,i) corresponds to the zeros
Z(i).
%
%   This is a modifaction of szeros.m written by Kjetil Havre
%
%   This Output-Zero-Direction-"through genaralized Eigenvalue"
decomposition
%   OZDE function is a modification of the szeros function contained
in mu
%   - toolbox. The modification consists of returning the output zero
%   directions and the state zero directions in addition to the zeros.
%   The output zero directions are defined as:
%                y'*G(z) = 0,
%   where s = z is a zero of G(s) and ' denotes conjugate transposed.
%   This is done by solving the generalized eigenvalue problem of the
%   transposed system:
%
%             | x' | y' | * | A-Iz |  B  |  =  |  0 | 0  |
%                           |------------|
%                           |  C   |  D  |
%
%   are solved by genaralized eigenvalues:
%
%             | A"-Iz |  C"  | * | xi |   =   | 0 |
%             |--------------|   |----|       |---|
%             |  B"   |  D"  |   | yi |       | 0 |
%
%   x = conj(xi); y = conj(yi);
%
%   The prime ' denotes the conjugate transposed and " denotes the
ordinary
%   transposed.
%
%   OZDE finds the transmission zeros z of a SYSTEM matrix.
Occasionally,
%   large zeros are included which may actually be at infinity.
Solving for
%   the transmission zeros of a system involves two generalized
eigenvalue
%   problems. EPP (optional) defines if the difference between two
generalized
%   eigenvalues is small. OZDE also finds the output y and the state x
%   directions of the zeros.
%
```

```
%   The output zero directions are stored as column vectors in Y, and
 the y's
%   are normalized so that Y'*Y = I. The state zero directions are
 stored as
%   columns in X. The degree of freedom to normalize the generalized
 eigen-
%   vectors is used to normalize the y part of the vectors. So the
 length of
%   x is not equal to one. Each column in Y and X corresponds to the
 element
%   in Z with same place.
%
%   For systems with more outputs than inputs the output zero
 direction
%   is not a complete basis for the left nullspace of G(z).
%   Zeros with multiplicity greater than one (rare cases which may
%   occure in non-minimal realizations), may (not sure) cause wrong
%   directions.
%
%   Comments, corrections and malfunctions, can be e-mailed to:
%           havre@kjemi.unit.no or skoge@kjemi.unit.no
%
%   See also: EIG, SZEROS, IZDE and SPOLES.
%   Algorithm based on Laub & Moore 1978 paper, Automatica
%

%   Note that when the number of outputs is larger than the number of
 inputs,
%   the output zero direction is not complete. A bit clearer: If z is
 a zero
%   of a non-square plant with number of outputs greater than number
 of inputs,
%   the zero direction is not a line but a surface. As an example,
 consider
%   G(s) with dimensions 3x2 (3 rows and 2 columns). Let s=z be a zero
 of
%   G(s) such that Yz'*G(z) = [0 0]; Since s=z is a zero then the rank
 of
%   G(z) has to be less than the normal rank of G(s), which at maximum
 can
%   be 2. This implies that rank of G(z) must be less than 2.
%   y is element in the three dimensional field of real numbers. Since
 the
%   rank of G(z) is maximum one the zero direction is a actually a
 subspace
%   in this three dimensional field of real numbers given by two basis
%   vectors. Since this function only gives one zero direction for a
 given
%   zero this direction does not describe the zero space on the output
%   completely. Two basis vectors are requiered.
%
%   Modification for square systems was made by:      Kjetil Havre
 24/4-1995.
```

```
%   Modification for non square systems was made by:  Kjetil Havre 2/5
 -1995.
%   Including the state zero dircetions was made by:  Kjetil Havre
 15/5-1995.
%   Modified so that first element of U(:,i) is real: Kjetil Havre
 3/2-1996.
%  Copyright 1996-2003 Sigurd Skogestad & Ian Postlethwaite
%  $Id: ozde.m,v 1.2 2004/01/19 14:52:11 aske Exp $

function [Z, Y, X] = ozde(sys,epp)
  if nargin < 1
    disp('usage:  [Z,Y,X] = ozde(G) ')
    return
  end
  if nargin == 1
    epp = eps;
  end
%   [ny,nu,nx]=size(sys);
[nx,nx] = size(sys.A);[nx,nu] = size(sys.B);[ny,nx] = size(sys.C);

  if class(sys) == 'tf'|'ss'|'zpk'|'frd'
    [a,b,c,d] =ssdata(sys);
    if nx == 0
      disp('SYSTEM has no states')
    end
    sysu = [a b; c d];
    sysu = sysu';

% find generalized eigenvalues of a square system matrix

    if ny == nu
      x = zeros(nx+nu,nx+nu);
      x(1:nx,1:nx) = eye(nx);
      [vech, ev] = eig(sysu, x);

      % Add something here to check for not a number or infinite.
      % remove corresponding vectors.
      % Also remove top part corresponding to the states
      % and normalize bottom part. KH 21/4 - 1995.

      z = diag(ev);                       % Extract the eigenvalues.
      kc=0;                               % Counter for eigenvalues.

      for k=1:max(size(z)),
         logic = ~isnan(z(k)) & isfinite(z(k));
         if logic
            kc= kc+1;
            Z(kc,1) = z(k);
            vech2(:,kc) = vech(:,k);
         end
      end
% Split in x and y.
      vx = vech2(1:nx, : );
      vy = vech2(nx+1:nx+ny,:);
```

Computing the sizes of no. of states, inputs and outputs from sys

changed finite to isfinite.

→ Similar changes were also done in the function izde.m and therefore the code is not attached.

```
% Normalize columns.
      [nvr, nvc] = size( vy );
      for i=1:nvc,
          nrmy    = norm(vy(:,i));
   if nrmy > 1000*epp
      vx(:,i) = vx(:,i)/nrmy;
      vy(:,i) = vy(:,i)/nrmy;
   else
      vy(:,i) = zeros(ny,1);
   end
          Inz = find( abs(vy(:,i) ) > 1000*epp );
   if isempty(Inz) == 0
             X(:,i) = conj(vx(:,i) * exp( -
angle(vy(Inz(1),i))*sqrt(-1) ) );
             Y(:,i) = conj(vy(:,i) * exp( -
angle(vy(Inz(1),i))*sqrt(-1) ) );
   else
      X(:,i) = conj(vx(:,i));
      Y(:,i) = conj(vy(:,i));
   end
      end

    else                                      % Non-square systems
      nrm = norm(sysu,1);
      if nu > ny
        x1 = [ sysu  nrm*(rand(nx+nu,nu-ny)-.5)];
        x2 = [ sysu  nrm*(rand(nx+nu,nu-ny)-.5)];
      else
        x1 = [ sysu; nrm*(rand(ny-nu,nx+ny)-.5)];
        x2 = [ sysu; nrm*(rand(ny-nu,nx+ny)-.5)];
      end
      [x]= zeros(size(x1));
      x(1:nx,1:nx) = eye(nx);

      [v1h z1h] = eig(x1,x);              % Compute the genaralized
 eigenvalues
      [v2h z2h] = eig(x2,x);              % for the two augumented
 systems.

      z1h2 = diag( z1h );
      z2h2 = diag( z2h );
      z2 = z2h2(~isnan(z2h2) & finite(z2h2));
      kc=0;                               % Counter for eigenvalues.

      for k=1:max(size(z1h2)),
         logic = ~isnan(z1h2(k)) & finite(z1h2(k));
         if logic
            kc= kc+1;
            z1(kc,1) = z1h2(k);
            vech2(:,kc) = v1h(:,k);
         end
      end
      nz = length(z1);
      vech3 = [];
```

```
        Z = [];
        for i=1:nz,
          if any(abs(z1(i)-z2) < nrm*sqrt(epp))
            Z = [Z; z1(i)];
            vech3 = [vech3 vech2(:,i)];
  end
        end
        if isempty(vech3)
           Z = []; Y = []; X = [];
   return;
        end
% Split columns in x and y.
        vx = vech3(1:nx,:);
        vy = vech3(nx+1:nx+ny,:);
% Normalize columns.
        [nvr, nvc] = size( vy );
        for i=1:nvc,
            nrmy    = norm(vy(:,i));
     if nrmy > 1000*epp
        vx(:,i) = vx(:,i)/nrmy;
        vy(:,i) = vy(:,i)/nrmy;
     else
        vy(:,i) = zeros(ny,1);
     end
            Inz = find( abs(vy(:,i) ) > 1000*epp );
     if isempty(Inz) == 0
               X(:,i) = conj(vx(:,i) * exp( -
angle(vy(Inz(1),i))*sqrt(-1) ) );
               Y(:,i) = conj(vy(:,i) * exp( -
angle(vy(Inz(1),i))*sqrt(-1) ) );
     else
        X(:,i) = conj(vx(:,i));
        Y(:,i) = conj(vy(:,i));
     end
        end

      end

   else
      error('matrix is not a SYSTEM matrix')
      return
   end
%
% Copyright MUSYN INC 1991,  All Rights Reserved
% Copyright MUSYN INC 1995,  All Rights Reserved
%

usage:  [Z,Y,X] = ozde(G)
```

Published with MATLAB® R2016a

Ha en god dag!