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' | 'spk' | '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 -4.620e-1 9.895e-1; 0 0 0 9.895e-1 -4.620e-1]
B = [-.629 .624; .055 -.172; .030 -.108;
     -.186 -.139; -.23 -.056]
C = [-.7223 -.5170 .3386 -.1633e-1 0.1121;
     -.8913 .4728 .9876 .8425 .2186]
D = [0 0; 0 0]
Bd = [-.062 -.067; .131 .040; .022 -.106;
      -.188 .027; -.045 .014]
G = pck(A,B,C,D); Gd = pck(A,Bd,C,D);
G = 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)
[2y,Yz,Yx] = ozde(GSS); 2y, YZ % zero output directions
[2u,Uz,Xu] = izde(GSS); 2u, 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 gains
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 response of G and Gd

figure()
vplot('l1v,lm',vsvd(Gf),1,';');  % Plot singular values of G
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black';

figure()
vplot('l1v,lm',Gdf,1,';');  % Plot elements in Gd
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black';

figure()
vplot('l1v,lm',vrGA(GF),1,';');  % RGA-elements
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black';

gdiag=vdiag(vdiag(Gf)); prga = mmult(gdiag,minv(Gf));
vplot('l1v,lm',prga,1,';');  % PRGA-elements
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black';
cldg=mmult(prga,Gdf);

figure()
vplot('l1v,lm',cldg,1,';');  % CLDG-elements
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black';

% Some closed-loop analysis ........
Two simple PI controllers

```matlab
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);
```

```matlab
figure()
vplot('l1v,lm',Sf,1,':'); % sensitivity function elements
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black';

figure()
vplot('l1v,lm',svsvd(Sf),1,':'); % singular values of sensitivity
axs = gca;
axs.TickLabelInterpreter = 'latex';
axs.XColor = 'black';
axs.YColor = 'black'; % (which is less than 1 as desired)

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

figure()
vplot('l1v,lm',svsvd(SGd),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 =
\[ -5.1310e-03 \quad 0 \quad 0 \quad 0 \quad 0 \\
0 \quad -7.3660e-02 \quad 0 \quad 0 \quad 0 \\
0 \quad 0 \quad -1.8290e-01 \quad 0 \quad 0 \\
0 \quad 0 \quad 0 \quad -4.6200e-01 \quad 9.8950e-01 \\
0 \quad 0 \quad 0 \quad -9.8950e-01 \quad -4.6200e-01 \\
\]

\[ B = \\
-6.2900e-01 \quad 6.2400e-01 \\
5.5000e-02 \quad -1.7200e-01 \\
3.0000e-02 \quad -1.0800e-01 \\
-1.8600e-01 \quad -1.3900e-01 \\
-1.2300e+00 \quad -5.6000e-02 \\
\]

\[ C = \\
-7.2230e-01 \quad -5.1700e-01 \quad 3.3860e-01 \quad -1.6330e-02 \quad 1.1210e-01 \\
-8.9130e-01 \quad 4.7280e-01 \quad 9.8760e-01 \quad 8.4250e-01 \quad 2.1860e-01 \\
\]

\[ D = \\
0 \quad 0 \\
0 \quad 0 \\
\]

\[ Bd = \\
-6.2000e-02 \quad -6.7000e-02 \\
1.3100e-01 \quad 4.0000e-02 \\
2.2000e-02 \quad -1.0600e-01 \\
-1.8800e-01 \quad 2.7000e-02 \\
-4.5000e-02 \quad 1.4000e-02 \\
\]

\[ \text{ans} = \\
-5.1310e-03 \quad + \quad 0.0000e+001 \\
-7.3660e-02 \quad + \quad 0.0000e+001 \\
-1.8290e-01 \quad + \quad 0.0000e+001 \\
-4.6200e-01 \quad + \quad 9.8950e-011 \\
-4.6200e-01 \quad - \quad 9.8950e-011 \\
\]

\[ \text{poles} = \\
-4.6200e-01 \quad + \quad 9.8950e-011 \\
-4.6200e-01 \quad - \quad 9.8950e-011 \\
-5.1310e-03 \quad + \quad 0.0000e+001 \\
-7.3660e-02 \quad + \quad 0.0000e+001 \\
-1.8290e-01 \quad + \quad 0.0000e+001 \\
\]

System Gd, which has the same A, C, and D matrices as G, has the same poles as G. These two are the same, as expected.
Output pole vectors

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

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

Tzero =

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

By -
-1.3882e+15 \leftarrow \text{have one zero extra, but the ones far away are not important?} \\
-2.8614e+00 \\
-1.5163e-01 \\
-2.7459e-01 \\
\text{Output zero vectors}

YZ =

Columns 1 through 2

\begin{align*}
8.3367e-01 & - 1.0209e-16i \\ -5.5227e-01 & + 6.7633e-17i \\
9.0620e-01 & - 1.1098e-16i \\ -4.2286e-01 & + 5.1785e-17i \\
\end{align*}

Columns 3 through 4

\begin{align*}
8.6970e-01 & + 0.0000e+00i \\ -4.9357e-01 & + 0.0000e+00i \\
9.0925e-01 & + 0.0000e+00i \\ -4.1625e-01 & + 0.0000e+00i \\
\end{align*}

\text{Zu =}

\begin{align*}
-2.8614e+00 \\
-2.7459e-01 \\
-1.5163e-01 \\
\end{align*}

\text{then these are the same as above}

\text{UZ = Input zero vectors}

\begin{align*}
8.2399e-01 & \quad 7.0796e-01 \\ 5.6661e-01 & \quad 7.0625e-01 \\
\end{align*}

\text{GO =}

\begin{align*}
8.8197e+01 & \quad -8.6822e+01 \\
1.0879e+02 & \quad -1.1015e+02 \\
0 & \quad 1.0000e+00 \\
\end{align*}

\begin{align*}
0 & \quad 0 \\
\text{same as expected} \\
\text{Inf} \\
\end{align*}

\text{GO0 =}

\begin{align*}
8.8197e+01 & \quad -8.6822e+01 \\
1.0879e+02 & \quad -1.1015e+02 \\
\end{align*}

\text{RGA0 =}

\begin{align*}
3.6066e+01 & \quad -3.5066e+01 \\
-3.5066e+01 & \quad 3.6066e+01 \\
\end{align*}

\text{\texttt{\textbackslash rga(G0)} ans =}

\begin{align*}
3.6066e+01 & \quad -3.5066e+01 \\
\end{align*}

\text{diagonal pairing preferred, due to positive RGA elements}

\text{\texttt{\textbackslash rga(G0)} ans gives the same RGA matrix as above}
\[
\begin{array}{ccc}
-3.5066e+01 & 3.6066e+01 & 0 \\
0 & 1.0000e+00 & Inf
\end{array}
\]

\text{PRGA0 =}

\[
\begin{array}{ccc}
3.6066e+01 & -2.8429e+01 & 0 \\
-4.4485e+01 & 3.6066e+01 & 0 \\
0 & 1.0000e+00 & Inf
\end{array}
\]

same as RGA as expected

\text{Gd0 =}

\[
\begin{array}{ccc}
7.8665e+00 & 8.9525e+00 & 0 \\
1.1667e+01 & 1.1338e+01 & 0 \\
0 & 1.0000e+00 & Inf
\end{array}
\]

closed loop disturbance gain

\text{CLDg0 =}

\[
\begin{array}{ccc}
-4.7971e+01 & 5.5809e-01 & 0 \\
7.0837e+01 & 1.0642e+01 & 0 \\
0 & 1.0000e+00 & Inf
\end{array}
\]

\text{GinvGd0 =}

\[
\begin{array}{ccc}
-5.4391e-01 & 6.3277e-03 & 0 \\
-6.4312e-01 & -9.6685e-02 & 0 \\
0 & 1.0000e+00 & Inf
\end{array}
\]

\text{ans =}

\[
\begin{array}{c}
-4.4989e-01 + 9.9120e-01i \\
-4.4989e-01 - 9.9120e-01i \\
-1.9997e-01 + 2.7488e-01i \\
-1.9997e-01 - 2.7488e-01i \\
-6.1451e-02 + 6.1312e-02i \\
-6.1451e-02 - 6.1312e-02i \\
-1.6931e-01 + 0.0000e+00i
\end{array}
\]

interpolating input vector (zero order hold)
interpolating input vector (zero order hold)
Plotting $\sigma_i(\omega)$

closed loop effect of disturbances

which is less than 1 as desired
% 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 modification of zeros.m written by Kjetil Hovre
% This Output-Zero-Direction-*through genralized Eigenvaule* decomposition
% OZDE function is a modification of the zeros 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:
% \[
% \begin{bmatrix}
%     x' & y' & * \\
%     \text{A-Iz} & \text{B} & = & 0 & 0 \\
%     \text{C} & \text{D} & 
% \end{bmatrix}
% \]
% are solved by genralized eigenvalues:
% \[
% \begin{bmatrix}
%     \text{A''-Iz} & \text{C''} & * \\
%     \text{B''} & \text{D''} & 
% \end{bmatrix}
% \begin{bmatrix}
%     x_i & = & 0 \\
%     y_i & 
% \end{bmatrix}
% \]
% \( x = \text{conj}(xi); y = \text{conj}(yi); \)
% 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 eigenvectors 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 occur 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=2 be a zero of G(s) such that Yz'*G(z) = [0 0]; Since s=2 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 required.

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 directions 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)
in if nargin < 1
  disp('usage: [Z, Y, X] = ozde(U) ')
  return
end
if nargin == 1
  epp = eps;
end
[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);
  kc = 0;
  % Extract the eigenvalues.
  % 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, :);

% 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,
        nrmv = norm(vy(:,i));
        if nrmv > 1000*epp
            vx(:,i) = vx(:,i)/nrmv;
            vy(:,i) = vy(:,i)/nrmv;
        else
            vy(:,i) = zeros(ny,1);
        end
    end
    Inz = find(abs(vy(:,i)) > 1000*epp);
    if isempty(Inz) == 0
        X(:,i) = conj(vx(:,i) * exp(-angle(vy(Inz(:,i)))*sqrt(-1)));
        Y(:,i) = conj(vy(:,i) * exp(-angle(vy(Inz(:,i)))*sqrt(-1)));
    else
        X(:,i) = conj(vx(:,i));
        Y(:,i) = conj(vy(:,i));
    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(1:1:nx,1:nx) = eye(nx);
    [v1h z1h] = eig(x1,x);
    % Compute the generalized eigenvalues
    [v2h z2h] = eig(x2,x);
    % for the two augmented systems.
    z1h2 = diag(z1h);
    z2h2 = diag(z2h);
    z2 = z2h2(-isin(z2h2) & finite(z2h2));
    kc=0;
    % Counter for eigenvalues.
    for k=1:max(size(z1h2)),
        logic = -isin(z1h2(k)) & finite(z1h2(k));
        if logic
            kc = kc+1;
            zl(kc,1) = z1h2(k);
            vech2(:,kc) = v1h(:,k);
        end
    end
    nz = length(zl);
    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 vr(:,i) = zeros(ny,1); end end if 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 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] = ordm(G) Published with MATLAB® R2016a