```
%
% designarc.m
%
% 角速度制御器を設計しシミュレーションする。
%

% =====
% 線形モデルの読み込みと制御系設計用モデルへの変換
% =====

% 線形モデルの読み込み
loadlinmdl;

% 縦・横姿勢角速度モデルの抽出
ist_ssr = [2 3 13 14];  % La, Ma, xintg1(=gx), xintg2(=gy)
io_ssr  = [13 14];      % gx, gy
ii_ssr  = [2 3];        % ulon, ulat
Pssr    = ss(Pf.a(ist_ssr, ist_ssr), Pf.b(ist_ssr, ii_ssr), ...
    Pf.c(io_ssr, ist_ssr), Pf.d(io_ssr, ii_ssr));
Pssr.StateName  = Pf.StateName(ist_ssr);
Pssr.OutputName = Pf.OutputName(io_ssr);
Pssr.InputName  = Pf.InputName(ii_ssr);

% 誤差システム（積分器拡大系）の作成
% x = [e_x e_eys]'
% u = [e_u]'
%  e_x … 制御対象の状態変数 x から、時刻 t=∞での値 x_∞を引いた値、e_x = x - x_∞
%  e_y … 制御対象の出力 y を、時刻 t=∞での値 y_∞=r（目標値）から引いた値、e_y = r - y
%  e_eys … e_ys から、その時刻 t=∞での値 e_ys∞を引いた値、e_eys = e_ys - e_ys∞
%    e_ys … r - y の積分値
%  e_u … 制御対象の入力 u から、時刻 t=∞での値 u_∞を引いた値、e_u = u - u_∞
Aa  = [Pssr.a zeros(4, 2); -Pssr.c zeros(2, 2)];
Ba  = [Pssr.b; zeros(2, 2)];


% =====
% 制御器の設計
% =====

% 重み行列
Q1  = diag([1 1]);
Q2  = diag([1 1]);
R   = 0.0001 * eye(2);

% 補償器設計
```

```
Q   = blkdiag(Pssr.c' * Q1 * Pssr.c, Q2);
Pa  = icare(Aa, Ba, Q, R, [], [], []);  % 連続時間代数 Riccati 方程式を解く
Pa11    = Pa(1:4, 1:4);
Pa12    = Pa(1:4, 5:6); % = Pa21'
Pa22    = Pa(5:6, 5:6);
Fa  = -inv(R) * Pssr.b' * Pa11;
Ga  = -inv(R) * Pssr.b' * Pa12;
Msys    = [Pssr.a Pssr.b; Pssr.c Pssr.d];
Ha  = [(-Fa + Ga * inv(Pa22) * Pa12') eye(2)] * inv(Msys) * [zeros(4, 2); eye(2)];

% 補償器を State-space 形式にまとめる
% dxc/dt = Ac*xc + Bc*uc
% yc     = Cc*xc + Dc*uc
% xc = [e_ys]'
% yc = [ulon ulat]'
% uc = [xa^' ya' r']'
%    xa^ = [La^ Ma^ xintg1^(=gx^) xintg2^(=gy^)]'
%    ya  = [gx gy]'
%    r   = [x_s1 y_s1]'
Ac  = zeros(2, 2);
Bc  = [ 0 0 0 0 -1  0 1 0;
        0 0 0 0  0 -1 0 1];
Cc  = Ga;
Dc  = [Fa zeros(2, 2) Ha];
Kc  = ss(Ac, Bc, Cc, Dc);
StateName_K{1}      = 'es_gx';  % = ∫(x_s1 - gx)dt
StateName_K{2}      = 'es_gy';  % = ∫(y_s1 - gy)dt
OutputName_K(1:2)   = Pssr.InputName(1:2);
InputName_K{1}      = [Pssr.StateName{1} '_est'];
InputName_K{2}      = [Pssr.StateName{2} '_est'];
InputName_K{3}      = [Pssr.StateName{3} '_est'];
InputName_K{4}      = [Pssr.StateName{4} '_est'];
InputName_K(5:6)    = Pssr.OutputName(1:2);
InputName_K{7}      = 'x_s1';
InputName_K{8}      = 'y_s1';
Kc.StateName        = StateName_K;
Kc.OutputName       = OutputName_K;
Kc.InputName        = InputName_K;

% 簡易型オブザーバの設計
%   角速度 gx, gy はセンサ計測値をそのまま用いる。
%   モーメント Mx, My はモデルが安定なのでモデル＋入力のみで推定する。
%   （フィードバックのない特殊な推定法。）
% x = [La^ Ma^]'
% y = [La^ Ma^ xintg1^(=gx^) xintg2^(=gy^)]'
% u = [ulon ulat gx gy]'
```

```
Ao  = Pssr.a(1:2, 1:2);
Bo  = [Pssr.b(1:2, :) zeros(2, 2)];
Co  = [eye(2); zeros(2, 2)];
Do  = [zeros(2, 2) zeros(2, 2); zeros(2, 2) eye(2)];
Pobs    = ss(Ao, Bo, Co, Do);
StateName_O{1}      = [Pssr.StateName{1} '_est'];
StateName_O{2}      = [Pssr.StateName{2} '_est'];
OutputName_O{1}     = StateName_O{1};
OutputName_O{2}     = StateName_O{2};
OutputName_O{3}     = [Pssr.StateName{3} '_est'];
OutputName_O{4}     = [Pssr.StateName{4} '_est'];
InputName_O(1:2)    = Pssr.InputName(1:2);
InputName_O(3:4)    = Pssr.OutputName(1:2);
Pobs.StateName      = StateName_O;
Pobs.OutputName     = OutputName_O;
Pobs.InputName      = InputName_O;
Pobsd   = c2d(Pobs, 0.00125, 'zoh');    % 離散化
Aod = Pobsd.a;
Bod = Pobsd.b;
Cod = Pobsd.c;
Dod = Pobsd.d;  % 代数ループ注意


% =====
% 線形特性解析
% =====

% グラフ番号のオフセット
nfigofst    = 10;

% 閉ループ系の作成
OutputName_cl{1}    = 'gx';
OutputName_cl{2}    = 'gy';
OutputName_cl{3}    = 'ulon';
OutputName_cl{4}    = 'ulat';
InputName_cl{1}     = 'x_s1';
InputName_cl{2}     = 'y_s1';
InputName_cl{3}     = 'ulon';
InputName_cl{4}     = 'ulat';
Pcl = connect(Pssr, Kc, Pobs, InputName_cl, OutputName_cl);

% シミュレーション：ステップ目標値応答・ステップ外乱応答
Pcl_step    = Pcl * diag([0.2 0.2 50 50]);  % ステップの振幅を指定する
Pcl_step.InputName  = InputName_cl;
hfig    = figure(nfigofst + 1);
clf reset;
```

```
138  step(Pcl_step, 0:0.01:4);
139  grid on;
140  set(hfig.Children(2:end), 'FontSize', 12);
141  set(hfig, 'Name', '閉ループ特性 (1) ステップ目標値応答・ステップ外乱応答');
142  set(hfig, 'Position', [([100 100] + [40 20] * (get(hfig, 'Number') - nfigofst)) 700 600]);
143
144  % 閉ループ系のボード線図
145  hfig    = figure(nfigofst + 2);
146  clf reset;
147  bode(Pcl(1:2, [1 3]), {0.01 1000}); % 縦系
148  grid on;
149  set(hfig.Children(2:end), 'FontSize', 12);
150  set(hfig, 'Name', '閉ループ特性 (2) ボード線図・縦系');
151  set(hfig, 'Position', [([100 100] + [40 20] * (get(hfig, 'Number') - nfigofst)) 1000 600]);
152  hfig    = figure(nfigofst + 3);
153  clf reset;
154  bode(Pcl(1:2, [2 4]), {0.01 1000}); % 横系
155  grid on;
156  set(hfig.Children(2:end), 'FontSize', 12);
157  set(hfig, 'Name', '閉ループ特性 (3) ボード線図・横系');
158  set(hfig, 'Position', [([100 100] + [40 20] * (get(hfig, 'Number') - nfigofst)) 1000 600]);
159
160  % 開ループ系の作成
161  Pssr_ol = Pssr;
162  Pssr_ol.InputName{1}    = [Pssr.InputName{1} '_in'];
163  Pssr_ol.InputName{2}    = [Pssr.InputName{2} '_in'];
164  Kc_ol   = Kc;
165  Kc_ol.OutputName{1}     = [Kc.OutputName{1} '_out'];
166  Kc_ol.OutputName{2}     = [Kc.OutputName{2} '_out'];
167  Pobs_ol = Pobs;
168  Pobs_ol.InputName{1}    = [Pobs.InputName{1} '_out'];
169  Pobs_ol.InputName{2}    = [Pobs.InputName{2} '_out'];
170  OutputName_ol{1}    = 'ulon_out';
171  OutputName_ol{2}    = 'ulat_out';
172  InputName_ol{1}     = 'ulon_in';
173  InputName_ol{2}     = 'ulat_in';
174  Pol = connect(Pssr_ol, Kc_ol, Pobs_ol, InputName_ol, OutputName_ol);
175
176  % MIMO 安定余有の計算
177  Smg = allmargin(-Pol);
178  disp('Stability margin:')
179  disp(['  I/O #1:     Stable = ' num2str(Smg(1).Stable)])
180  disp(['    Gain margin       [-] : ' num2str(Smg(1).GainMargin)]);
181  disp(['    GM. frequency [rad/s] : ' num2str(Smg(1).GMFrequency)]);
182  disp(['    Phase margin    [deg] : ' num2str(Smg(1).PhaseMargin)]);
183  disp(['    PM. frequency [rad/s] : ' num2str(Smg(1).PMFrequency)]);
```

```
184  disp(['  I/O #2:     Stable = ' num2str(Smg(2).Stable)])
185  disp(['    Stable                : ' num2str(Smg(2).Stable)]);
186  disp(['    Gain margin       [-] : ' num2str(Smg(2).GainMargin)]);
187  disp(['    GM. frequency [rad/s] : ' num2str(Smg(2).GMFrequency)]);
188  disp(['    Phase margin    [deg] : ' num2str(Smg(2).PhaseMargin)]);
189  disp(['    PM. frequency [rad/s] : ' num2str(Smg(2).PMFrequency)]);
190
191
192  % =====
193  % 非線形シミュレーションのためのパラメータ設定
194  % =====
195
196  % パラメータ
197  ct2.ts_ctrl_i   = 0.00125;      % [s] インナーループ制御のサンプリング時間
198  ct2.z_kp2       = 1000;         % [LSB/(rad/s)] 比例ゲイン、Z 軸（ヨー）角速度
199  ct2.z_ki2       = 0;            % [LSB s/(rad/s)] 積分ゲイン、Z 軸（ヨー）角速度
200  ct2.z_kd2       = 0;            % [LSB/(rad/s)/s] 微分ゲイン、Z 軸（ヨー）角速度
201  ct2.coef_dflt   = 0.025;        % 角速度偏差微分値を通すフィルタの係数
202  ct2.Aod         = Aod;          % 状態推定器 A 行列
203  ct2.Bod         = Bod;          % 状態推定器 B 行列
204  ct2.Cod         = Cod;          % 状態推定器 C 行列
205  ct2.Dod         = Dod;          % 状態推定器 D 行列 ※代数ループの警告が出るため実際には使わない
206  ct2.Fa          = Fa;           % 制御器行列 F
207  ct2.Ga          = Ga;           % 制御器行列 G
208  ct2.Ha          = Ha;           % 制御器行列 H
209  ct2.C_dist_thr  = C_dist_thr;   % コマンド分配則、推力
210  ct2.C_dist_lon  = C_dist_lon;   % コマンド分配則、ピッチ
211  ct2.C_dist_lat  = C_dist_lat;   % コマンド分配則、ロール
212  ct2.C_dist_dir  = C_dist_dir;   % コマンド分配則、ヨー
213
214  % パラメータの書き出し
215  save sim_param_ct2.mat ct2;
216
```