รายการอ้างอิง

1. K. Ogata, *Modern Control Engineering.* Singapore: Prentice-Hall International, Inc., 1995.

2. G. J. Balas, J. C. Doyle, K. Glover, A. Packard, and R. Smith, *μ-Analysis and Synthesis Toolbox for use with MATLAB.* Natick, MA: The MathWorks Inc., 1998.

3. Y. Wang, R. Zhou, and C. Wen, "Robust Load-Frequency Controller Design for Power Systems.," *IEE Proc.*, 140, 1, (1993): 11-16.

4. G. Ray, A. N. Prasad, and G. D. Prasad, "A New Approach to the Design of Robust Load-Frequency Controller for Large Scale Power Systems.," *Electric Power Systems Research*, 51, 1, (1999): 13-22.

5. D. Rerkpreedapong, A. Hasanovic, and A. Feliachi, "Robust Load Frequency Control Using Genetic Algorithms and Linear Matrix Inequalities.," *IEEE Trans. Power Syst*, 18, 2, (2003): 855-861.

6. A. I. Lur'e and V. N. Postnikov, "On the Theory of Stability of Control Systems.," *Applied Mathematics and Mechanics*, 8, 3, (1944): 3–13.

7. V. M. Popov, "Absolute Stability of Nonlinear Systems of Automatic Control.," *Automation and Remote Control*, 22, (1962): 857–875.

8. S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory.* Philadelphia: SIAM, 1994.

9. D. Banjerdpongchai, *Parametric Robust Controller Synthesis Using Linear Matrix Inequalities.* PhD thesis, Stanford University, Stanford, October 1997.

10. D. Banjerdpongchai and J. P. How, "Parametric Robust $\mathcal{H}_2$ Control Design Using Iterative Linear Matrix Inequalities Synthesis.," *AIAA J. of Guidance Control and Dynamics*, 23, 1, (2000): 138–142.

11. ฐาปนา นามประดิษฐ์ และ เดวิด บรรเจิดพงศ์ชัย, "การออกแบบตัวควบคุม $\mathcal{H}_2$ คงทนสำหรับระบบลูเรที่มีการจำกัดความชันโดยวิธีอสมการเมทริกซ์เชิงเส้นฮอมอโทปี.," *วิศวกรรมสาร ฉบับวิจัยและพัฒนา*, 15, 3, (2547): 68–81.

12. K. Zhou and J. C. Doyle, *Essentials of Robust Control.* New Jersey: Prentice Hall, 1998.

13. S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control Analysis and Design.* England: JOHN WILEY & SONS, 1996.

14. O. Toker and H. Ozbay, "On The NP-Hardness of Solving Bilinear Matrix Inequalities and Simultaneous Stabilization with Static Output Feedback.," *Proc. American Conrol Conf*, (1995): 2525–2526.

15. L. El Ghaoui and V. Balakrishnan, "Synthesis of Fixed-Structure Controllers via Numerical Optimization.," *Proc. IEEE Conf. on Decision and Control*, (1994): 2678–2683.

16. K. C. Goh, T. L., S. M. G., Papavassilopoulos, G. P., and J. H. Ly, "Biaffine Matrix Inequality Properties and Computational Methods.," *Proc. American Conrol Conf*, (1994): 850–855.

17. A. A. Stoorvogel, "The Robust $\mathcal{H}_2$ Control Problem: A Worst-Case Design.," *IEEE Trans. Aut. Control*, AC-38, 9, (1993): 1358–1370.

18. A. Packard, K. Zhou, P. Pandey, and G. Becker, "A Collection of Robust Control Problems Leading to LMI's.," in *Proc. IEEE Conf. on Decision and Control*, (1991): 1245–1250.

19. H. Werner, P. Korba, and T. C. Yang, "Robust Tuning of Power System Stabilizers Using LMI-Techniques.," *IEEE Trans. Control Sys. Tech.*, 11, 1, (2003): 147–152.

20. H. Saadat, *Power System Analysis.* New York: McGraw-Hill, 1998.

21. P. Kundur, *Power System Stability and Control.* New York: McGraw-Hill, Inc., 1994.

22. A. L. Richter and R. A. De Carlo, "Continuation Methods: Theory and Applications," *IEEE Trans. Aut. Control*, AC-28, 6, (1983): 660–665.

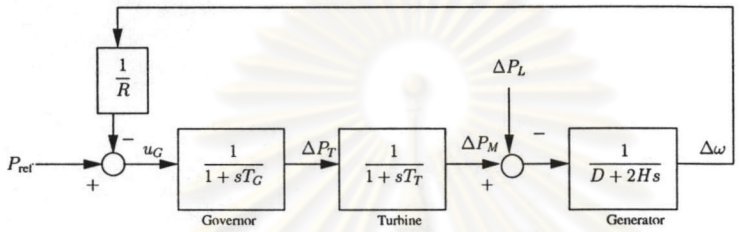
23. P. Gahinet, A. Nemirovski, A. J. Laub, and M. Chilali, *LMI Control Toolbox for Use with MATLAB.* Natick, MA: The MathWorks, 1995.

ภาคผนวก

ภาคผนวก ก

แบบจำลองระบบควบคุมกำลังการผลิตและความถี่

ในภาคผนวก ก. นำเสนอแบบจำลองของระบบควบคุมกำลังผลิตและความถี่ ซึ่งเริ่มจากการควบคุมกำลังผลิตและความถี่ 1 พื้นที่ ดังแสดงในรูปที่ ก.1



รูปที่ ก.1: แผนภาพบล็อกการควบคุมกำลังการผลิตและความถี่ 1 พื้นที่

ซึ่งมีสมการสถานะเป็นดังนี้

$$\begin{aligned} \dot{x} &= Ax + B_u u + B_w w \\ y &= C_y x \end{aligned}$$

เมื่อ $x = \begin{bmatrix} \Delta\omega & \Delta P_T & \Delta P_M \end{bmatrix}$ เป็นตัวแปรสถานะของระบบ โดยที่ $\Delta\omega$ คือ ค่าเบี่ยงเบนความเร็วรอบของเครื่องกำเนิดไฟฟ้า, ΔP_T คือ ค่าเบี่ยงเบนกำลังงานที่จ่ายให้กับเทอร์ไบน์, ΔP_M คือค่าเบี่ยงเบนกำลังเชิงกลที่จ่ายให้กับเครื่องกำเนิดไฟฟ้า, u_G คือ สัญญาณควบคุมที่เข้าตัวบังคับเทอร์ไบน์ และ $w = \Delta P_L$ คือ ความต้องการกำลังไฟฟ้าที่เปลี่ยนแปลง

$$\begin{aligned} A &= \begin{bmatrix} -D/(2H) & 1/(2H) & 0 \\ 0 & -1/T_T & 1/T_T \\ -1/(RT_G) & 0 & -1/T_G \end{bmatrix} \\ B_u &= \begin{bmatrix} 0 & 0 & 1/T_G \end{bmatrix}^T \\ C_y &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \\ B_w &= \begin{bmatrix} -1/2H & 0 & 0 \end{bmatrix}^T \end{aligned}$$

โดยที่ T_G คือ ค่าคงตัวทางเวลาของตัวบังคับเทอร์ไบน์ (วินาที)

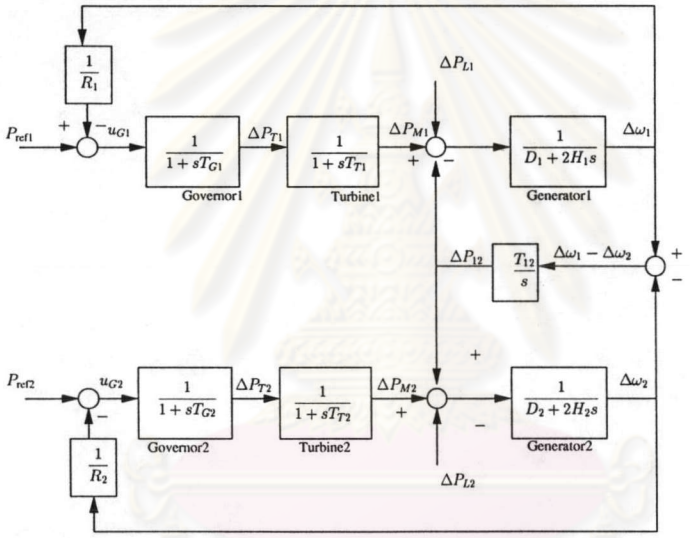
T_T คือ ค่าคงตัวทางเวลาของเทอร์ไบน์ (วินาที)

H คือ ค่าคงที่ความเฉื่อย (initia constant) (MW-sec/MV A)

D คือ ค่าคงที่การหน่วงเนื่องจากโหลด (load-damping constant)

R คือ ค่าคงที่การคุมค่าความเร็ว

จุดประสงค์ในการควบคุมระบบนี้คือการรักษาความเร็วรอบของเครื่องกำเนิดไฟฟ้าเพื่อให้ความถี่ของไฟฟ้าคงที่ภายใต้สัญญาณรบกวน (ความต้องการกำลังไฟฟ้า) ที่เปลี่ยนแปลง พิจารณาโครงสร้างของระบบควบคุมดังกล่าวพบว่ามีการควบคุมเบื้องต้น (primary control) เป็นการป้อนกลับความเร็วรอบของเครื่องกำเนิดไฟฟ้าผ่านค่าคงที่ $1/R$ ซึ่งเทียบเท่ากับเป็นการควบคุมแบบสัดส่วน (proportional control) แต่ในการทำงานจริงระบบจะมีการเชื่อมต่อกับระบบผลิตไฟฟ้าจากหลายๆพื้นที่จึงได้รับผลกระทบจากการเปลี่ยนแปลงความถี่ จากระบบในพื้นที่ติดกันทำให้การควบคุมแบบสัดส่วนไม่เพียงพอในการควบคุมความเร็วรอบ โดยทั่วไปจึงนิยมเพิ่มตัวควบคุมอื่นๆขึ้นมาเพื่อนำผลจากความถี่ในพื้นที่เชื่อมต่อกันมาพิจารณาด้วย ตัวควบคุมที่เพิ่มขึ้นนี้ถูกเรียกว่าตัวควบคุมประกอบ (supplementary control) ระบบที่มีการเชื่อมต่อกันใน 2 พื้นที่และตัวควบคุมเบื้องต้น แสดงดังรูป ก.2



รูปที่ ก.2: แผนภาพบล็อกการควบคุมกำลังการผลิตและความถี่ 2 พื้นที่เมื่อมีการควบคุมเบื้องต้น

การเชื่อมต่อแหล่งผลิตไฟฟ้าหลายพื้นที่เข้าด้วยกันต้องพิจารณาความเร็วรอบของเครื่องกำเนิดไฟฟ้าให้ซิงโครไนส์กัน เพื่อไม่ให้ระบบขาดเสถียรภาพ ดังนั้นเราจึงนำผลต่างของค่าเบี่ยงเบนความเร็วรอบใน 2 พื้นที่มาผ่านการอินทิกรัลและคูณด้วยค่าสัมประสิทธิ์ซิงค์โครไนซ์ทอร์ก T_{12} ส่งผลให้กำลังเชื่อมต่อเบี่ยงเบนด้วยปริมาณ (ΔP_{12}) กำลังงานในปริมาณนี้จะถูกป้อนกลับมาชดเชยเพื่อให้ความเร็วรอบซิงโครไนส์กัน ดังนั้นเพื่อควบคุมไม่ให้การเปลี่ยนแปลงความต้องการไฟฟ้าที่เกิดขึ้นในพื้นที่หนึ่งมีผลต่อปริมาณไฟฟ้าเชื่อมต่อที่อาจส่งผลไปยังการผลิตไฟฟ้าในพื้นที่ติดกัน จึงนำสัญญาณที่ต้องการควบคุมทั้งสองมารวมกันเป็นผลบวกเชิงเส้น ทำให้ได้สัญญาณที่ต้องการควบคุมที่เรียกว่า ACE (area control error) $ACE_i = \Delta P_{12} + B_i \Delta\omega_i$ เมื่อ B_i คือ ตัวประกอบไบแอสความถี่ (frequency bias factor) โดยที่ $B_i = \frac{1}{R_i} + D_i$ และ Δ หมายถึง ดังนั้นตัวควบคุมประกอบจึงมีหน้าที่ในการควบคุม ACE ให้อยู่ในค่ามาตรฐานที่กำหนด สำหรับสมการสถานะของระบบควบคุมกำลังผลิตและความถี่ 2 พื้นที่อ้างอิงได้จาก §3.2

ภาคผนวก ข

ชุดคำสั่งในการคำนวณ

ภาคผนวก ข. จะนำเสนอชุดคำสั่งสำหรับ การวิเคราะห์ความไม่แน่นอนเชิงพารามิเตอร์ค่าจริง การวิเคราะห์สมรรถนะ $\mathcal{H}_2$ คงทน และการสังเคราะห์ตัวควบคุม $\mathcal{H}_2$ คงทน ชุดคำสั่งดังกล่าวเป็นรูทีน (routine) ในโปรแกรม MATLAB และแก้ปัญหาการโปรแกรมกึ่งแน่นอน (semidefinite programming) โดยใช้ชุดคำสั่งใน LMI Control Toolbox [23] ซึ่งแก้ปัญหาโดยใช้วิธีจุดภายใน (interior point method)

ข.1 การวิเคราะห์ความไม่แน่นอนเชิงพารามิเตอร์ค่าจริง

```
function [A,Bp,Bu,Cq,Dqp,Dqu,Cz,Dzp,Dzu,Cy,Dyp,Dyu,L]=FormLure [A_set,Bu_set,Cy_set,Dyu_set,...
                                                                Cz_set,Dzu_set]

% Developed by : Jeeranuch Juengudomporn
% Date : February 2005
% [A,Bp,Bu,Cq,Dqp,Dqu,Cz,Dzp,Dzu,Cy,Dyp,Dyu,L]=FormLure [A_i,Bu_i,Cy_i,Dyu_i,Cz_i,Dzu_i]
%
% FUNCTION
%         FormLure : This function is used to formulate a set of system model subject to real
%                           parametric uncertainty  in the form of to Lur'e system.
%
% INPUT
%         A_set,Bu_set,Cy_set,...             A family of state-space system matrices.
%         Dyu_set,Cz_set,Dzu_set
%
% OUTPUT
%         A,Bu,Cy,Dyu,Cz,...                   Nominal parameters.
%         Dzp,Dzu,Bp,Dyp,Cq,Dqu,...            Matrices which represent real parametric uncertainty.
%         L                                    The diagonal matrix and represents sector bound.

% Initialize---------------------------------------------------------------------

N = size(A_set,2) ;          %  N is no.of state-space models
n = size(A_set{1},1);        %  n is no.of dynamics state
n_u =size(Bu_set{1},2);      %  n_u is no.of input
n_y =size(Cy_set{1},1);      %  n_y is no.of output
n_z =size(Cz_set{1},1);      %  n_z is no.of performance output

% Find nominal parameter and uncertainty matrices------------------------

[A,Delta_A]         =  FindNominal(A_set);
[Bu,Delta_Bu]       =  FindNominal(Bu_set);
[Cy,Delta_Cy]       =  FindNominal(Cy_set);
[Cz,Delta_Cz]       =  FindNominal(Cz_set);
[Dzu,Delta_Dzu]     =  FindNominal(Dzu_set);

% Singular value decomposition-------------------------------------------------
```

```
uncertainty_matrix = [Delta_A  Delta_B; Delta_Cy  Delta_Dyu; Delta_Cz  Delta_Dzu];
[U_bar,Delta_bar,V_bar] = svd(uncertainty_matrix);
rank_s = rank(Sbar);

% Scale matrices dimension-----------------------------------------------------

row_no = 1 ;
col_no = 1 ;
for row_no = 1:1:rank_s
  for col_no = 1:1:rank_s
    Delta  = Delta_bar(row_no,col_no);
  end
end

U = U_bar * Delta;
V = Delta * V_bar;

row_no = 1;
row_Bp = 1;
row_Dyp = 1;
row_Dzp = 1;

% Form matrices Bp, Dyp, Dzp, Cq and Dqu----------------------------------------

while row_no <= n+n_y+n_z
  if row_no <= n
    Bp(row_Bp,:) = U(row_no,:);
    row_no = row_no +1;
    row_Bp = row_Bp +1;
  else
    if  row_no <= n+n_y
      Dyp(row_Dyp,:) = U(row_no,:);
      row_no = row_no +1;
      row_Dyp = row_Dyp +1;
    else
      Dzp(row_Dzp,:) = U(row_no,:);
      row_no = row_no +1;
      row_Dzp = row_Dzp +1;
    end
end

col_no = 1;
col_Cq = 1;
col_Dqu = 1;
while col_no <= n+n_u
  if col_no <= n
    Cq(:,col_Cq) = V(:,col_no)';
    col_no = col_no +1;
    col_Cq = col_Cq +1;
  else
    Dqu(:,col_Dqu) = V(:,col_no)';
    col_no = col_no +1;
    col_Dqu = col_Dqu +1;
end
L = Delta;
```

FindNominal.m

```
function [nominal_P, Delta_P] = FindNominal(set_P)

% Developed by : Jeeranuch Juengudomporn
% Date : February 2005
% [nominal_P, Delta_P] = FindNominal(set_P)
%
% FUNCTION
%             FindNominal        The  function which is used to find nominal value matrix.
%
% INPUT
%             set_P               A set of  whole parameter matrices.
%
% OUTPUT
%             nominal_P          Nominal value matrix (this program set to be the center of system parameter.
%             Delta_P            The difference between nominal value matrix and maximum value matrix.

% Initialize ------------------------------------------------------------------------

pt1 = 1;
ro = 1;
co =1 ;
N = size(set_P,2);
[n_row,n_col] = size(set_P{1});

% Find upper and lower matrices-----------------------------------------------

while pt1 <= N % N is number of data
       if pt1 == 1
           while ro <= n_row
             while co <= n_col
                max_P(ro,co) = (set_P{pt1}(ro,co));
                co = co +1;
             end
             ro = ro +1;
             co = 1;
           end
           ro = 1;
           co = 1;
           while ro <= n_row
             while co <= n_col
               min_P(ro,co) = (set_P{pt1}(ro,co));
               co = co +1;
             end
             ro = ro +1;
             co = 1;
           end
       else
        ro =1;
        co =1;
           while ro <= n_row
             while co <= n_col
                 if (set_P{pt2}(ro,co)) > max_P(ro,co)
                    max_P(ro,co) = (set_P{pt1}(ro,co));
                 end
                 co = co +1;
             end
             ro = ro +1;
```

```
            co =1;
          end
          ro = 1;
          co = 1;
          while ro <= n_row
             while co <= n_col
                if (set_P{pt2}(ro,co)) < min_P(ro,co)
                    min_P(ro,co) = (set_P{pt1}(ro,co));
                end
                co = co +1;
            end
            ro = ro +1;
            co =1;
          end
       end
       pt1 =pt1+1 ;
end

% Find nominal P and Delta P-----------------------------------------------

nominal_P = (max_P+min_P)/2;
Delta_P = max_P-nominal_P;
```

ข.2 การวิเคราะห์สมรรถนะ $\mathcal{H}_2$ คงทน

```
function [Lambda_,T_,h2cost_,infeasible_,Eu_norm_V] = viter1(A,Bp,Bw,Bu,...
                                    Cq,Dqp,Dqw,Dqu,...
                                    Cz,Dzp,Dzw,Dzu,...
                                    Cy,Dyp,Dyw,Dyu,...
                                    L,Ac,Bc,Cc)

% Developed by : Thapana Nampradit
% Date : February 2003
% FUNCTION
%      Perform V-Iteration in Controller Synthesis Procedure
%
% USAGE
%      [Lambda_,T_,h2cost_,infeasible_] = viter1(A,Bp,Bw,Bu,...
%                                  Cq,Dqp,Dqw,Dqu,...
%                                  Cz,Dzp,Dzw,Dzu,...
%                                  Cy,Dyp,Dyw,Dyu,...
%                                  L,Ac,Bc,Cc)
%
% INPUT
%      A,Bp,Bw,Bu,     System parameters
%      Cq,Dqp,Dqw,Dqu,
%      Cz,Dzp,Dzw,Dzu,
%      Cy,Dyp,Dyw,Dyu
%
%      L            The diagonal matrix represents sector bounds
%                   i.e. 0 <= phi_i(sigma)/sigma <= l_i
%      Ac,Bc,Cc       Controller's parameters
%
% OUTPUT
%      Lambda,T       Optimal multipliers for the given controller
%      h2cost         Upper bound of the worst-case H2 performace of
%                     the system for the given controller
%      infeasible      0:feasible, 1:infeasible
```

```
% Initialize---------------------------------------------------------------

ns = size(A,1);
np = size(Bp,2);
tA  = [ A,     Bu*Cc;
  Bc*Cy,  Ac+Bc*Dyu*Cc];
tBp = [ Bp;
  Bc*Dyp];
tBw = [ Bw;
  Bc*Dyw];
tCq = [ Cq,    zeros(size(Cq,1),size(Bu*Cc,2))];
tCz = [ Cz,    Dzu*Cc];
tDzp = Dzp;
setlmis([])

   % Define LMI variables-----------------------------------------------

   tP = lmivar(1,[2*ns 1]);

   % Lambda, and T
   for j=1:2,
      [Temp,n1,sTemp] = lmivar(1,[1 0]);
        for i=1:np-1,
         [TAdd,n2,sTAdd] = lmivar(1,[1 0]);
         [Temp,n1,sTemp] = lmivar(3,[sTemp,zeros(i,1);...
               zeros(1,i),sTAdd]);
      end

      switch j
         case 1,  Lambda = Temp;
         case 2,  T = Temp;
      end
   end

   % Define LMIs--------------------------------------------------------

    % LMI#1
   lmiterm([1 1 1 tP],1,tA,'s');
   lmiterm([1 1 1 0],tCz'*tCz);

   lmiterm([1 2 1 tP],tBp',1);
   lmiterm([1 2 1 Lambda],1,tCq*tA);
   lmiterm([1 2 1 T],1,L*tCq);
   lmiterm([1 2 1 0], tDzp' *tCz)

   lmiterm([1 2 2 Lambda],1,tCq*tBp,'s');
   lmiterm([1 2 2 T],-2,1);
   lmiterm([1 2 2 0], tDzp' *tDzp)
   % LMI#2
   lmiterm([-2 1 1 tP],1,1);

   % LMI#3
   lmiterm([-3 1 1 Lambda],1,1);

   % LMI#4
   lmiterm([-4 1 1 T],1,1);
```

```
LMISYS = getlmis;

% Define the objective function-----------------------------------------

nx = decnbr(LMISYS);
c  = zeros(nx,1);
for j = 1:nx,
  [tPj,Lambdaj] = defcx(LMISYS,j,tP,Lambda);
  c(j) = trace(tBw'*(tPj+tCq'*L*Lambdaj*tCq)*tBw);
 end

% Solve minimization problem--------------------------------------------

relacc =1e-12;
maxiter = 1000;
quiet =1;
feasp_reg = 1e9;
options = [relacc,maxiter,feasp_reg,10,quiet];
[copt,xopt] = mincx(LMISYS,c,options);

% Return output---------------------------------------------------------

if (~isempty(xopt))
   Lambda_ = dec2mat(LMISYS,xopt,Lambda);
   T_ = dec2mat(LMISYS,xopt,T);
   P_ = dec2mat(LMISYS,xopt,tP);

   h2cost_ = copt;
   infeasible_ = 0;
   Eu_norm_V = sqrt(xopt'*xopt);
   fprintf(' --> h2cost in v iteration = %6.4f\n',h2cost_);
else
   Lambda_ = zeros(np,np);
   T_ = zeros(np,np);
   h2cost_ = 10000
   infeasible_ = 1;
end
```

ข.3 การสังเคราะห์ตัวควบคุม $\mathcal{H}_2$ คงทน

```
function [Ac_,Bc_,Cc_,Dc_] = popovh2syn(A,Bp,Bw,Bu,...
                             Cq,Dqp,Dqw,Dqu,...
                             Cz,Dzp,Dzw,Dzu,...
                             Cy,Dyp,Dyw,Dyu,...
                             Lmin,Lmax,abstol,reltol,...
                             maxiter,minpoint)

% Developed by : Thapana Nampradit
% Date : February 2003
% FUNCTION
%      Popov Robust H2 Controller Synthesis
%
% USAGE
%      [Ac,Bc,Cc,Dc] = popovh2syn(A,Bp,Bw,Bu,...
%                           Cq,Dqp,Dqw,Dqu,...
%                           Cz,Dzp,Dzw,Dzu,...
%                           Cy,Dyp,Dyw,Dyu,...
%                           Lmin,Lmax,abstol,reltol,...
```

```
%                          maxiter,maxpoint)
%
% INPUT
%      A,Bp,Bw,Bu,     System parameters
%      Cq,Dqp,Dqw,Dqu,
%      Cz,Dzp,Dzw,Dzu,
%      Cy,Dyp,Dyw,Dyu
%
%      Lmin,Lmax       The diagonal matrices represent sector bounds
%                   i.e. l_{min,i} <= phi_i(sigma)/sigma <= l_{max,i}
%      abstol         Absolute tolerance (optional)
%      reltol         Relative tolerance (optional)
%      maxiter        Maximum iteration (optional)
%      maxpoint       Maximum point in homotopy (optional)
%
% OUTPUT
%      Ac,Bc,Cc,Dc     Controller's parameters

% Initialize---------------------------------------------------------------

if nargin < 22,
   maxpoint = 2^8;
   if nargin < 21,
      maxiter = 100;
      if nargin < 20,
         reltol = 1e-3;
         if nargin < 19,
            abstol = 1e-3;
            if nargin < 18,
               help popovh2syn;
               error('Not enough input arguments.');
            end
         end
      end
   end
end

% Balance realization  of system-------------------------------------------------------------------------

[A,Bp,Cq,G,T,Ti] = balreal(A,Bp,Cq);
Bu = Ti*Bu;
Bw = Ti*Bw;
Cy = Cy*T;
Cz = Cz*T;

% Design LQG Controller for The Nominal Plant------------------------------------------------------------

W = [  Cz'*Cz,    Cz'*Dzu;
  Dzu'*Cz,   Dzu'*Dzu];
V = [  Bw*Bw',    Bw*Dyw';
 Dyw*Bw',   Dyw*Dyw'];

[Alqg,Blqg,Clqg,Dlqg] = lqg(A,Bu,Cy,Dyu,W,V);
Clqg = -Clqg;
Ac = Alqg;
Bc = Blqg;
Cc = Clqg;
Dc = Dlqg;
```

```
% Initialize homotopy-------------------------------------------------------

numofpoint = 1;
n = 0;
infeasible = 1;

% Start algorithm-----------------------------------------------------------

while (infeasible ~= 0),
   n=n+1;
   lambdan = n/numofpoint;
   currentL = lambdan*L

   k = 1;
   abserr_k = 1;
   relerr_k = 1;
   h2cost(1) = inf;

   fprintf('\n [+] SOLVE POINT %d OF %d\n',n,numofpoint);

   % V-Iteration-------------------------------------------------------------

   fprintf(' [V1]\n');
   [Lambda,T,h2cost(k),infeasible] = viter1(A,Bp,Bw,Bu,...
                                Cq,Dqp,Dqw,Dqu,...
                                Cz,Dzp,Dzw,Dzu,...
                                Cy,Dyp,Dyw,Dyu,...
                                currentL,Ac,Bc,Cc);
  fprintf('\n');
     %fprintf(' --> h2cost = %6.3f\n',h2cost(k));
     while (((abserr_k > abstol) | (relerr_k > reltol)) & ...
        (k < maxiter) & (infeasible == 0)),
      k=k+1
     % K-Iteration-----------------------------------------------------------
     fprintf(' [K]\n');

     [P,Z,Q,Y,X,infeasible,h2cost(k)] = kiter1(A,Bp,Bw,Bu,...
                           Cq,Dqp,Dqw,Dqu,...
                           Cz,Dzp,Dzw,Dzu,...
                           Cy,Dyp,Dyw,Dyu,...
                           currentL,Lambda,T);
     if infeasible == 0,

        % Solve for Ac-------------------------------------------------------

        [Ac,Bc,Cc] = solveac1(A,Bp,Bw,Bu,...
                     Cq,Dqp,Dqw,Dqu,...
                     Cz,Dzp,Dzw,Dzu,...
                     Cy,Dyp,Dyw,Dyu,...
                     currentL,Lambda,T,...
                     P,Z,Q,Y,X,Alqg);

        % V-Iteration-------------------------------------------------------
fprintf(' [V2]\n');
        [Lambda,T,h2cost(k),infeasible] = viter1(A,Bp,Bw,Bu,...
                                     Cq,Dqp,Dqw,Dqu,...
                                     Cz,Dzp,Dzw,Dzu,...
```

```
                                    Cy,Dyp,Dyw,Dyu,...
                                    currentL,Ac,Bc,Cc);
        fprintf('\n');

        % Update error-----------------------------------------------------
         if h2cost(k-1) >= h2cost(k)
           abserr_k = abs(h2cost(k-1)-h2cost(k)) ;
           relerr_k = abserr_k/h2cost(k);

        end


      end

   end       % inner loop
   % Return output-----------------------------------------------------


% Restart synthesis loop-------------------------------------------


   if infeasible ~= 0,
     if numofpoint >= maxpoint,

error('Algorithm fail!');
      else
         numofpoint = numofpoint*2;
         n = 0;
         fprintf('\n [*] RESTART ALGORITHM WITH %d POINTS\n',numofpoint);
         Ac = Alqg;
         Bc = Blqg;
         Cc = Clqg;
      end
   end

end          % outer loop

% Return output-----------------------------------------------------------

fprintf('\n');
fprintf('-----------------------------------------------------------\n');
fprintf('Synthesis is complete\n');
fprintf('-----------------------------------------------------------\n');
if nargout<4,
     fprintf('Ac =\n');
     disp(Ac);
     fprintf('-----------------------------------------------------------\n');
     fprintf('Bc =\n');
     disp(Bc);
     fprintf('-----------------------------------------------------------\n');
    fprintf('Cc =\n');
     disp(Cc);
    fprintf('-----------------------------------------------------------\n');
     else
       Ac_ = Ac    ;
    Bc_ = Bc;
     Cc_ = Cc;
```

```
    Dc_ = Dc;
end
```

kiter1

```
function [P_,Z_,Q_,Y_,X_,infeasible_,h2cost_] = kiter1(A,Bp,Bw,Bu,...
                                    Cq,Dqp,Dqw,Dqu,...
                                    Cz,Dzp,Dzw,Dzu,...
                                    Cy,Dyp,Dyw,Dyu,...
                                    L,Lambda,T)
% Developed by : Thapana Nampradit
% Date : February 2005
%FUNCTION
%      Perform K-Iteration in Controller Synthesis Procedure
%
%USAGE
%      [P_,Z_,Q_,Y_,X_,infeasible] = kiter1(A,Bp,Bw,Bu,...
%                                Cq,Dqp,Dqw,Dqu,...
%                                Cz,Dzp,Dzw,Dzu,...
%                                Cy,Dyp,Dyw,Dyu,...
%                                L,Lambda,T)
%
%INPUT
%      A,Bp,Bw,Bu,     System parameters
%      Cq,Dqp,Dqw,Dqu,
%      Cz,Dzp,Dzw,Dzu,
%      Cy,Dyp,Dyw,Dyu
%
%      L            The diagonal matrix represents sector bounds
%                   i.e. 0 <= phi_i(sigma)/sigma <= l_i
%      Lambda,T       Fixed multipliers
%
%OUTPUT
%      P_,Z_,Q_,Y_,X_      Synthesis variables
%      infeasible     0:feasible, 1:infeasible

% Initialize---------------------------------------------------------------

ns = size(A,1);
np = size(Bp,2);
ny = size(Cy,1);
nu = size(Bu,2);
setlmis([])

   % Define LMI variables-----------------------------------------------

   P = lmivar(1,[ns 1]);
   Q = lmivar(1,[ns 1]);
   Z = lmivar(2,[ns ny]);
   Y = lmivar(2,[nu ns]);
   X = lmivar(1,[ny 1]);

   % Define LMIs--------------------------------------------------------

   % LMI#1
   lmiterm([1 1 1 P],1,A,'s');
   lmiterm([1 1 1 Z],1,Cy,'s');
   lmiterm([1 1 1 0],Cz'*Cz);
```

```
   lmiterm([1 2 1 P],Bp',1);
   lmiterm([1 2 1 -Z],Dyp',1);
   lmiterm([1 2 1 0],Lambda*Cq*A+T*L*Cq);
   lmiterm([1 2 1 0], Dzp' * Cz)

   lmiterm([1 2 2 0],Lambda*Cq*Bp+Bp'*Cq'*Lambda-2*T);
   lmiterm([1 2 2 0],Dzp' * Dzp);

   % LMI#2
   lmiterm([2 1 1 Q],A,1,'s');
   lmiterm([2 1 1 Y],Bu,1,'s');

   lmiterm([2 2 1 Q],Lambda*Cq*A,1);
   lmiterm([2 2 1 Q],T*L*Cq,1);
   lmiterm([2 2 1 Y],Lambda*Cq*Bu,1);
   lmiterm([2 2 1 Q], Dzp' *Cz, 1);
   lmiterm([2 2 1 0],Bp');

   lmiterm([2 2 2 0],Lambda*Cq*Bp+Bp'*Cq'*Lambda-2*T);
   lmiterm([2 2 2 0],Dzp' * Dzp);

   lmiterm([2 3 1 Q],Cz,1);
   lmiterm([2 3 1 Y],Dzu,1);
   lmiterm([2 3 3 0],-1);

   % LMI#3
   lmiterm([-3 1 1 X],1,1);
   lmiterm([-3 2 1 Z],1,1);
   lmiterm([-3 2 2 P],1,1);
   lmiterm([-3 3 1 0],0);
   lmiterm([-3 3 2 0],1);
   lmiterm([-3 3 3 Q],1,1);

LMISYS = getlmis;

% Define the objective function-----------------------------------------

nx = decnbr(LMISYS);
c  = zeros(nx,1);
L
for j = 1:nx,
  [Pj,Zj,Xj] = defcx(LMISYS,j,P,Z,X);
  c(j) = trace(Bw'*(Pj+Cq'*L*Lambda*Cq)*Bw + (Bw'*Zj*Dyw) + (Dyw'*Zj'*Bw) + (Dyw'*Xj*Dyw));
end

% Solve minimization problem-------------------------------------------
relacc = 1e-12;
maxiter = 1000;
quiet =1;
options = [relacc,maxiter,-1,10,quiet];

[copt,xopt] = mincx(LMISYS,c,options);

% Return output--------------------------------------------------------
if (~isempty(xopt))
  P_ = dec2mat(LMISYS,xopt,P);
  Q_ = dec2mat(LMISYS,xopt,Q);
```

```
  Z_ = dec2mat(LMISYS,xopt,Z);
  Y_ = dec2mat(LMISYS,xopt,Y);
  X_ = dec2mat(LMISYS,xopt,X);
   infeasible_ = 0;
   h2cost_ =copt;
    Eu_norm_K = sqrt(xopt'*xopt);
  fprintf(' --> h2cost in k iteration = %6.9f\n',h2cost_);
 else
   P_ = zeros(ns,ns);
   Q_ = zeros(ns,ns);
   Z_ = zeros(ns,ny);
   Y_ = zeros(nu,ns);
   X_ = zeros(ny,ny);
   infeasible_ = 1;
   h2cost_ = inf ;
end
```

solveAc.m

```
function [Ac_,Bc_,Cc_,P] = solveac(A,Bp,Bw,Bu,...
                        Cq,Dqp,Dqw,Dqu,...
                        Cz,Dzp,Dzw,Dzu,...
                        Cy,Dyp,Dyw,Dyu,...
                        L,Lambda,T,P,Z,Q,Y,X);
% Developed by : Thapana Nampradit
% Data : February 2005
%FUNCTION
%      Solve for a controller dynamic Ac
%
%USAGE
%      [Ac,Bc,Cc] = solveac(A,Bp,Bw,Bu,...
%                      Cq,Dqp,Dqw,Dqu,...
%                      Cz,Dzp,Dzw,Dzu,...
%                      Cy,Dyp,Dyw,Dyu,...
%                      L,Lambda,T,P,Z,Q,Y,X);
%
%INPUT
%      A,Bp,Bw,Bu,     System parameters
%      Cq,Dqp,Dqw,Dqu,
%      Cz,Dzp,Dzw,Dzu,
%      Cy,Dyp,Dyw,Dyu
%
%      L            The diagonal matrix represents sector bounds
%                   i.e. 0 <= phi_i(sigma)/sigma <= l_i
%      Lambda,T      Multipliers (from V-Iteration)
%      P,Z,Q,Y,X     Synthesis variables (from K-Iteration)
%
%OUTPUT
%      Ac,Bc,Cc       Controller's parameters

% Initialize---------------------------------------------------------------

ns = size(A,1);
np = size(Bp,2);
% Calculate Bc, and Cc-----------------------------------------------------

Bc = Z;
Cc = Y*inv(eye(size(P*Q,1))-P*Q);
% Initialize---------------------------------------------------------------
```

```
R  = inv(P-inv(Q)); % R =P22
P12 = eye(ns);
tP  = [ P      P12;
  P12'    R];

tA  = [ A,     Bu*Cc;
  Bc*Cy,  Bc*Dyu*Cc];

tBp = [ Bp;
      Bc*Dyp];
tBw = [ Bw;
      Bc*Dyw];
tCq = [ Cq,          zeros(size(Cq,1),size(Bu*Cc,2))];
tCz = [ Cz,          Dzu*Cc];
tJ  = [ zeros(ns,ns);   eye(ns)];


setlmis([])

   % Define LMI variables---------------------------------------------------

   Ac = lmivar(2,[ns ns]);
   % Define LMI-------------------------------------------------------------

   % LMI#1
   lmiterm([1 1 1 0],tA'*tP+tP*tA+tCz'*tCz);
   lmiterm([1 1 1 Ac],tP*tJ,tJ','s');
   lmiterm([1 2 1 0],tBp'*tP+Lambda*tCq*tA+T*L*tCq);
   lmiterm([1 2 2 0],Lambda*tCq*tBp+tBp'*tCq'*Lambda-2*T);

LMISYS = getlmis;

% Solve feasibility problem-----------------------------------------------

maxiter = 1000;
quiet = 1;
target = 0;
options = [0,maxiter,-1,10,1];
[tmin,xfeas] = feasp(LMISYS,options,target);

if tmin <=  target+1e-6
  Ac_ = dec2mat(LMISYS,xfeas,Ac);
  Bc_ = Bc;
  Cc_ = Cc;
end
```

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวจีรนุช จึงอุดมพร เกิดเมื่อวันที่ 27 กุมภาพันธ์ 2520 ที่กรุงเทพมหานครเป็นบุตรนายบรรหาร จึงอุดมพรและนางวารุณี พฤกษ์ธาราธิกูล สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมระบบควบคุม จากภาควิชาวิศวกรรมระบบควบคุม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยี พระจอมเกล้าเจ้าธนบุรี ในปีการศึกษา 2541 หลังจากนั้นศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย สังกัดห้องปฏิบัติการวิจัยระบบควบคุมเมื่อปีการศึกษา 2545