เอกสารอ้างอิง

1. Proudfoot, C. G., Gawthrop, P. J., and Jacobs, O. L. R., "Self-tuning PI control of a pH neutralisation process," Proc. IEE, 130 (9), 267-273, 1983.
2. Astrom, K. J., and Wittenmark, B., Computer controlled sysytem theory and design, pp. 93-398, Prentice-Hall, Inc., N.J., 1984.
3. Clarke, D. W., and Gawthrop, P. J., "Self-tuning control," Proc. IEE, 126 (6), 633-644, 1979.
4. Clarke, D. W., "Self-tuning controller design and implementation," Real time computer control, (S. Bennett, and D. A. Linkens, eds.), vol. 24, pp. 44-70, Peter Peregrinus Ltd., London, 1984.
5. Gawthrop, P. J., "Self-tuning PID controller : algorithm and implementation", TRANS. IEEE, 31 (3), 201-209, 1986.
6. Hesketh, T., "State space pole placing self-tuning regulator using input output values," Proc. IEE, 129 (7), 123-128, 1982.
7. Warwick, K., "Self-tuning property of state space self tuner," Proc. IEE, 129 (5), 96-100, 1982.
8. Dexter, A. L., "Self-tuning control algorithm for single chip microcomputer implementation," Proc. IEE, 130 (9), 255-260 1983.
9. Gertler, J., and Banyasz, C., "A recursive maximum likelihood identification method," IEEE TRANS, Vol AC-19 (6), 816-819, 1974.

10. Clarke, D. W., Hodgson, A. J. F, and Tuffs, P. S., "Offset problem and k-incremental predictors in self-tuning control," Proc. IEE, 130 (9), 217-223, 1983.

11. Wellstead, P. E., Edmunds, J. M., Prager, D., and Zanker, P., "Self-tuning pole/zero assignment regulators," Int. J. Control, Vol.30 (1), 1-36, 1979.

12. Wellstead, P. E., Prager, D. L., and Zanker, P., "Pole assignment self-tuning regulators," Proc. IEE, 126 (8), 781-787, 1979.

13. Astrom, K. J., and Wittenmark, B., "Self-tuning controllers based on pole-zero placement," Proc. IEE, 127 (5), 120-130, 1980.

14. Allidina, A. Y., and Hughes, F. M., "Generalised self-tuning controller with pole placement," Proc. IEE, 127 (1), 13-18, 1980.

15. Prager, D. L., and Wellstead, P. E., "Multivariable pole assignment self-tuning regulators," Proc. IEE, 128 (1), 9-18, 1981.

16. Tsay, Y. T., and Shieh, L. S., "State space approach for self tuning feedback control with pole assignment," Proc. IEE, 128 (5), 93-101, 1981.

17. Shieh, L. S., Wang, C. T., and Tsay, Y. T., "Fast suboptimal state space self tuner for linear stochastic multivariable system," Proc. IEE, 130 (7), 143-154, 1983.

18. Fuchs, J. J., "Recursive least squares algorithm revisited," Proc. IEE, 128 (3), 74-76, 1981.

19. Osorio, C. A., and Mayne, D. Q., "Deterministic convergence of self-tuning regulator with variable forgetting factor," Proc. IEE, 128 (1), 19-23, 1981.

20. Gawthrop, P. J., "Hybrid self-tuning control," Proc. IEE, 127 (9), 229-236, 1980.

21. Grimble, M. J., "Controller for LQG self-tuning application with colored measurement noise and dynamic costing," Proc. IEE, 133 (1), 19-29, 1986.

22. Stepanopoulos, G., Chemical process control : An introduction to theory and pratice, pp. 45-279, Prentice-Hall, Inc., N.J., 1984.

23. Kenney, T. B., Process automation : A 14-part series, pp. 19-24 McGraw-Hill Book Co., New York, 3rd ed., 1985.

24. ยุทธศิลป์ วิภาตะศิลิน และ รศ.ประยูร เชี่ยววัฒนา "Industrial instrumentation control" เอกสารของวิชา Senior Project 162-499 พ.ศ. 2529

25. National Semiconductor Corporation, Linear data book, pp. 156-162, 1982.

26. Omani, F. K., and Sinha, N. K., "A modified state space approach to multivariable self-tuning control with pole assignment," Proc. IEE, 134 (1), 31-37, 1987.

27. Shih, D. H., and Kung, F. C., "Shifted Legendre approach to the analysis and identification of a linear delayed system with a nonlinear gain," Proc. IEE, 133 (3), 127-132, 1986.

28. Bezanson, L. W., and Harris, S. L., " Identification and control of extruder using multivariable algorithms," Proc. IEE, 133 (4), 145-152, 1986.

29. Yokogawa Hokushin Electric, "Yew series 80," Electronic control system, Japan, 1985.

30. Yokogawa Hokushin Electric, "Yewpack mark II," Packaged control system, Japan, 1986.

31. Kraus, T. W., and Myron, T. J., "Self-tuning controller uses pattern recognition approach," Control engineering macazine, June, 1984.

32. Porter, B., Jones, A. H., and McKeown, C. B., "Real-time expert tuners for PI controller," Proc IEE, 134 (4), 260-263, 1987.

ภาคผนวก

ภาคผนวก ก

โปรแกรมที่ใช้ทดสอบ

การควบคุมแบบจูนปรับตัวเองที่ทำในวิทยานิพนธ์นี้ใช้อัลกอริทึมในการควบคุมทั้งการจำลองและการควบคุมจริงดังนี้

<u>โปรแกรมที่ 1</u> เป็นการควบคุมแบบจูนปรับตัวเองชนิดเล็งเลิศ ซึ่งภายในโปรแกรมสามารถกำหนดการควบคุมได้ 2 แบบ คือ Minimum variance (MV) และ Generalized minimum variance (GMV) โดยตั้งค่า $P(z^{-1}),Q(z^{-1})$ ในโปรแกรมใหม่ ตัวอย่างเช่น ถ้าต้องการควบคุมระบบด้วยตัวควบคุมแบบ MV ก็กำหนดให้ P(1) เท่ากับ 1 Q(1) เท่ากับ 0

<u>โปรแกรมที่ 2</u> เป็นโปรแกรมของการควบคุมแบบจูนปรับตัวเองชนิดกำหนดโพล ใช้อัลกอริทึมแบบ Implicit สามารถกำหนดค่าโพลโดยการตั้งค่า ζ และ w ของผลตอบของกระบวนการและตัวควบคุมได้ ซึ่งในโปรแกรมนี้จะตั้งค่าโดยใช้ตัวแปร DR และ freq

<u>โปรแกรมที่ 3</u> เป็นโปรแกรมชนิดเดียวกับโปรแกรมที่ 2 แต่ใช้อัลกอริทึมแบบ Explicit สามารถตั้งค่าตัวแปรเหมือนในโปรแกรมที่ 2

<u>โปรแกรมที่ 4</u> เป็นโปรแกรมของการควบคุมแบบจูนปรับตัวเองชนิด PID ซึ่งอาศัยการจูนค่าด้วยวิธีของ Ziegler & Nichols ภายในโปรแกรมสามารถเลือกการควบคุมได้ทั้งแบบ P,PI,PID โดยการตั้งค่าตัวแปรในโปรแกรมเสียใหม่ โปรแกรมนี้แสดงให้เห็นการประยุกต์ใช้ตัวควบคุมแบบจูนปรับตัวเองอย่างง่าย ๆ

<u>เงื่อนไขของการใช้โปรแกรม</u>

1. จะต้องจำนวน Order ของระบบที่ต้องการควบคุม ได้แก่ Nofy,Nofu เป็นต้น
2. จะต้องทราบลักษณะทั่วไปของกระบวนการเช่น Time constant และ Delay time เป็นต้น เพื่อใช้เป็นข้อมูลเบื้องต้นของการตั้งค่าเริ่มต้น
3. จะต้องกำหนดคาบของการสุ่มในโปรแกรม

โปรแกรมร่วม

```
{ # include file }

Function Equation ( a : Vector ; flowi,ti,qq,kkk,area,h : real ) : real ;
Begin
        Equation := (flowi*(ti-a[1]) + qq*kkk)/(area*h) ;
End ;


Procedure GenFunConSys ( Var a : Vector ; flowi,ti,qq,kkk,area,h : Real ) ;
Var
        a0      :  Vector ;
        k       :  Matrix ;
        t0,t    :  Real ;
        Nst2,i  :  Integer ;
Begin
        Nst2 := 1 ;
        t := 0 ;
        For i := 1 to Nst2 Do
            a0[i] := a[i] ;
        t0 := t ;
        For i := 1 to Nst2 Do
            Begin
            k[1,i] := Delta*Equation ( a,flowi,ti,qq,kkk,area,h ) ;
            a[i] := a0[i] + 0.5*k[1,i] ;
            End ;
        t0 := t + 0.5*delta ;
        For i := 1 to Nst2 Do
            Begin
            k[2,i] := Delta*Equation ( a,flowi,ti,qq,kkk,area,h ) ;
            a[i] := a0[i] + 0.5*k[2,i] ;
            End ;
        For i := 1 to Nst2 Do
            Begin
            k[3,i] := Delta*Equation ( a,flowi,ti,qq,kkk,area,h ) ;
            a[i] := a0[i] + k[3,i] ;
            End ;
        t0 := t + delta ;
        For i := 1 to Nst2 Do
            Begin
            k[4,i] := Delta*Equation ( a,flowi,ti,qq,kkk,area,h ) ;
            a[i] := a0[i] + ( k[1,i] + 2*k[2,i] + 2*k[3,i] + k[4,i] )/6 ;
            End ;
        t := t0 ;
End ;


Procedure SaveData ( a : DataVec ; n : Integer);
Var
        i       :  Integer;
        outfile :  String[20];
        Fn      :  File of Real;
        data    :  Real;
```

```
Begin
        Gotoxy(10,12) ;
        Write ( '     OUTPUT FILENAME  :     ' ) ;
        Readln ( outfile ) ;
        outfile := outfile + '.dat' ;
        Assign ( Fn,outfile ) ;
        Rewrite ( Fn ) ;
        data := n ;
        Write ( Fn,data ) ;
        For i := 1 to n Do
              Begin
              data := a[i] ;
              Write ( Fn,data ) ;
              End ;
End;


Procedure InitialM ( Var p : Matrix ; d : Real ) ;
Var
        i,j      :  Integer ;
Begin
        For i := 1 to Norder Do
              Begin
              For j := 1 to Norder Do p[i,j] := 0 ;
              End ;
        For i := 1 to Norder Do p[i,i] := d ;
End ;
Procedure InitialV ( Var p : Vector ; d : Real ) ;
Var
        i        :  Integer ;
Begin
        For i := 1 to Norder Do p[i] := d ;
End ;
Procedure InitialE ( Var a : Estpartyp ; d : Real ) ;
Var
        i        :  Integer ;
Begin
        With a Do
             Begin
             For i := 1 to Npar Do diag[i] := d ;
             For i := 1 to Npar Do th[i] := 0 ;
             For i := 1 to Npar Do fi[i] := 0 ;
             For i := 1 to Npar Do phi[i] :=  1 ;
             for i := 1 to Noff do offdiag[i] := 0 ;
{             THETA[1] := -0.5 ;
             THETA[2] := -0.1 ;
             THETA[3] := - 0.2 ;
             THETA[4] :=  0.1 ;}
{             THETA[5] :=  - 0.01;
             THETA[6] :=   0.133;
             THETA[7] :=  -  0.02;}
             End ;
```

```
End ;
Procedure InitialC ( Var a : Clgroup ; d : Real ) ;
Var
        i       :  Integer ;
Begin
        With a Do
            Begin
            For i := 1 to Npar Do th[i] := 0 ;
            For i := 1 to Npar Do fi[i] := 0 ;
            For i := 1 to Npar Do phi[i] := 0 ;
            For i := 1 to Npar Do g[i] := 0 ;
            for i := 1 to Nsum do s[i] := d ;
            End ;
End ;


Procedure Sq ( u,y : Real ; var perr : real ; Var Clstate : Clgroup ) ;
Var
        i,j,ij,ji,jl    : integer ;
        si,ss,fj        : real ;
        a,b,c,d         : real ;
        al,eb           : real ;
Begin

        With clstate Do
            Begin
            for i := 1 to ny + nu Do perr := perr - th[i]*fi[i] ;
            d := perr ;
            for i := ny+nu+1 to n do perr := perr - th[i]*fi[i] ;
            si := Lamda ;
            ss := si*si ;
            ij := 0 ;
            ji := 0 ;
            for j := 1 to n do
                  begin

                  for i := 1 to j do
                        begin
                        ji := ji + 1 ;
                        fj := fj + s[ji]*fi[i] ;
                        end ;
                  a := si/Lamda ;
                  b := fj/ss ;
                  ss := ss + fj*fj ;
                  si := sqrt ( ss ) ;

                  a := a / si ;
                  g[j] := s[ji]*fj ;
                  s[ji] := s[ji] * a ;
                  jl := j - 1 ;
                  if (jl  > 0 ) then
                        begin
                        for i := 1 to jl do
```

```
                    begin
                    ij := ij + 1 ;
                    c := s[ij] ;
                    s[ij] := a * ( c - b * g[i] ) ;
                    g[i] := g[i] + c * fj ;
                    end ;
                end ;

            ij := ij + 1 ;

            end ;

            for i := 1 to n do th[i] := th[i] + a* g[i] ;
            for i := 1 to n - 1 do fi[n - i + 1] := fi[n- i] ;
            fi[1] := - y ;
            fi[ny + 1] := u ;
            fi[nu+ny+1] := d ;
{
            al := 0.8 ;
            eb := eb1*al + (1 - al)*d ;

            fi[ny+nu+1]:= d - eb ;
            eb1 := eb ;
            perr := d ;}

            end ;
end ;
```

โปรแกรมที่ 1

```
Program SimulationTest;
Const
          Norder   =  10  ;
          Nofy     =  2  ;
          Nofu     =  1  ;
          Nofp     =  2 ;
          Nofn     =  0  ;
          Nstate   =  1  ;
          QQ       =  0.01 ;{0.1steady state}
          PO       =  0.50 ;   {5damping CLIMBING}
          TO       =  117 ; {0.1dec u & for smooth & sharp response}
          Lamda    =  0.98 ;
          Delta    =  6 ;
          DR       =  0.707 ;
          FREQ     =  0.1 ;
          Xp       =  1 ;{1.7239 ;}
          Delay    =  5  ;
          Max      =  10 ;
          DataMax  =  450 ;
          Npar     =  20 ;
          Noff     =  190 ;
          Nsum     =  210 ;
          Value    =  1905 ;{1 = 476.25}
          MaxU     =  4.0 ;
          Cv       =  100 ;
          Sigma    =  1 ;
Type
            Vec1   = Array [1..Npar] of Real ;
            Vec2   = Array [1..Noff] of Real ;
            Vec3   = Array [1..Nsum] of Real ;
         Estpartyp = Record
            n,ny   :  Integer ;
            nu     :  Integer ;
            th     :  Vec1 ;
            fi     :  Vec1 ;
            phi    :  Vec1 ;
            diag   :  Vec1 ;
            offdiag : Vec2 ; { like PO }
            End ;
          Clgroup  =  Record
            n      :  Integer ;
            ny     :  Integer ;
            nu     :  Integer ;
            nn     :  Integer ;
            fi     :  Vec1 ;
            phi    :  Vec1 ;
            th     :  Vec1 ;
            g      :  Vec1 ;
            s      :  Vec3 ;
            End ;
          Matrix   =  Array [1..4,1..DataMax] of Real ;
          Vector   =  Array [1..Max] of Real ;
          DataVec  =  Array [1..DataMax] of Real ;
```

```
Var
          yf,fis,uu,yy,ww    :  DataVec ;
          a,b      :  vector ;
          t,w,u,y  :  Real ;
          m,nn,mm  :  Integer ;
          i,naa,nb :  Integer ;
          Ul,QQ,TAB      :  VECl ;
          clstate  :  Clgroup ;
          perr     :  Real ;
          da       :  Vecl ;
          iou,ioy  :  Integer ;
          iof,ioc  :  Integer ;
          Kpid,ub       :  Real    ;
          flowi,ti,kkk,area,h : real ;
          pole : real ;
          ki,kd : real ;
          p1,p2,k : real ;
          hh : vecl ;
          pn,pd : vecl ;
          fac : real ;


procedure discr ( var hh : Vecl ; p1,p2,k,h : real ) ;
var
          i,j    : integer ;
          a,b,c  : real ;
          phi,tua : vecl ;
begin
          tua[1] := k*(h*h/2 + h*h*h*(p1+p2)/6) ;
          tua[2] := k*(h - h*h*(p1+p2)   - h*h*h*p1*p2/6 + h*h*h*(p1+p2)*(p1+p2)/6) ;
          phi[1] := 1 - p1*p2*h*h/2 ;
          phi[2] := h - h*h*(p1+p2)/2 ;
          phi[3] := -p1*p2*h + p1*p2*h*h*(p1+p2)/2 ;
          phi[4] := 1 - h*(p1+p2) - h*h*p1*p2/2 + h*h*(p1+p2)*(p1+p2)/2 ;
          hh[1] := tua[1] ;
          hh[2] := - phi[4]*tua[1] + phi[2]*tua[2] ;
          hh[3] := 1 ;
          hh[4] := - (phi[1] + phi[4]) ;
          hh[5] := phi[1]*phi[4] - phi[2]*phi[3] ;
          for i := 1 to 5 do writeln(i,'  ',hh[i] ) ;
end ;



Function Control_UGG ( w : Real ; var Ul,QQ,a,b : Vecl ; n,ny,m : integer ; fac : real ) : Real ;
Var
          u        :  Real ;
          i        :  Integer ;
Begin
```

```
        for i := 1 to n  do u := u - a[i]*b[i] ;
        u := u / a[ny + 1]*b[ny + 1] ;

        U1[3] := U1[2] ;
        U1[2] := U1[1] ;
        U1[1] := u ;
        u := (U1[1]*QQ[1]+U1[2]*QQ[2]+U1[3]*QQ[3]-(a[ny+1]*QQ[2]-
        fac)*b[ny+1]-QQ[3]*a[ny+1]*b[ny+2])/(a[ny+1]*QQ[1]+fac) ;

        Control_UGG := u ;
End ;

Procedure Polys(var QQ : Vec1) ;
var
        a,b,c : real ;
begin
        a := 0.1 ;
        b := 0.1 ;
        c := a*a ;
        b := b*b ;
        QQ[1] := 1 ;
        QQ[2] := - 2*a ;
        QQ[3] := c + b ;
        writeln(QQ[1],QQ[2],QQ[3]) ;
end ;

Procedure Roots(QQ : vec1 ) ;
var
        a,b,c : real ;
begin
        c := -4*QQ[1]*QQ[3] ;
        b := QQ[2]*QQ[2] ;
        a := sqrt(abs(b+c)) ;
        if (b+c) >= 0 then
             begin
             c := (- QQ[2] + a)/(2*QQ[1]) ;
             writeln ( ' roots   :   ',c) ;
             c := (- QQ[2] - a)/(2*QQ[1]) ;
             writeln ( '         :   ',c) ;
             end ;
        if (b+c) < 0 then
             begin
             c := - QQ[2]/(2*QQ[1]) ;
             writeln (' complex roots  :  ',c,'   +/- ',a/(2*QQ[1])) ;
             end ;
 end ;


Begin
        With clstate Do
             begin
              flowi := 0.05 ;
```

```
area := 3 ;
h := 2 ;
ti := 30 ;
kkk := 0.05 ;

For i := 1 to Nstate Do
       a[i] := ti ;
ny := Nofy+Nofp ;
nu := Delay + Nofu ;
n := ny + nu + Nofn ;   { beware Nofn # Nofn = 0 }
        for i := 1 to Npar do
              begin
              uu[i] := 0 ;
              yy[i] := ti ;
              ww[i] := 0 ;
              da[i] := 0 ;
              qq[i] := 0 ;
              tab[i] := 0 ;
              pn[i] := 0 ;
              pd[i] := 0 ;
              hh[i] := 0 ;
              yf[i] := 0 ;
              fis[i] := 0 ;
              ul[i] := 0 ;
              end ;
t := 0 ;
w := Cv ;
y := 0 ;
m := 2 + Delay ;
yy[m] := ti ;
InitialC ( clstate,10000 ) ;
th[ny + 1] := T0 ;
pole := 1/10 ;
p1 := pole ;
p2 := pole ;
k := pole*pole ;
Discr ( hh,p1,p2,k,delta ) ;
{              repeat until keypressed ;}
kd := hh[1]+hh[2] ;
pd[1] := hh[1]/kd ;
pd[2] := hh[2]/kd ;
ki := hh[3]+hh[4]+hh[5] ;
pn[1] := hh[3]/ki ;
pn[2] := hh[4]/ki ;
pn[3] := hh[5]/ki ;
Roots(pn) ;
{              pn[1] := 1 ;
pn[2] := 0 ;
pn[3] := 0 ;
pd[1] := 1 ;
pd[2] := 0 ;}
writeln(pd[1]+pd[2],pn[1]+pn[2]+pn[3]) ;
ki := 10 ;
```

```
kd := 0 ;
Kpid := 1 + delta*Ki + Kd/delta ;
QQ[2] := (delta + 2*Kd)/(delta*Kpid) ;
QQ[3] := Kd/(delta*Kpid) ;
QQ[1] := Kpid*kp ;
QQ[2] := -Kpid*QQ[2]*kp ;
QQ[3] := Kpid*QQ[3]*kp ;
QQ[4] := 1 ;
QQ[5] := - 1 ;
{ Polys(QQ) ;}
for i := 1 to 5 do writeln(QQ[i]);
QQ[4] := QQ[1]+QQ[2]+QQ[3] ;
QQ[1] := QQ[1]/QQ[4] ;
QQ[2] := QQ[2]/QQ[4] ;
QQ[3] := QQ[3]/QQ[4] ;
Roots(QQ) ;
{ qq[1] := 1 ;
qq[2] := 0 ;qq[3] := 0 ;qq[4] := 0 ;}

{ repeat until keypressed ;}
writeln(n,ny,nu) ;
fac := q0 ;
While m < DataMax do
        Begin
        u := Control_UG6 ( w,U1,QQ,th,da,n,ny,m,fac ) ;
        if m < 50 then u := Cv-yy[m] ;

        if u > 4000 then u := 4000 ;
        if u < 0 then u := 0 ;
        uu[m] := u ;
        if m = 220 then h := 0.6*h ;
{        if m = 220 then kkk := 1.6*kkk ;}
        if m = 250 then ti := 40 ;
        if m = 300 then ti := 20 ;
        if m = 350 then flowi := 1.1*flowi ;
        if m = 400 then flowi := 0.9*flowi ;
        GenFunConSys ( a,flowi,ti,uu[m-delay],kkk,area,h) ;
        yy[m+1] := a[1] ;
        if (abs((yy[m+1]-yy[m])/yy[m]) < eps ) then fac := q0 else fac := 0 ;
        yf[m+1] := (yy[m+1] - pd[2]*yf[m])/pd[1] ;
        fis[m+1] := (pn[1]*yy[m+1] + pn[2]*yy[m] + pn[3]*yy[m-1] - pd[2]*fis[m])/pd[1] ;
        perr := fis[m+1] ;
        Sq ( uu[m-Delay+1],- yf[m-Delay+2],perr,clstate ) ;
        for i := 1 to n - 1 do da[n + 1 - i] := da[n - i] ;
        da[1] := yf[m+1] ;
        da[ny + 1] := uu[m] ;      {beware of Nofn in control u}
{        da[nu+ny+1] := fi[nu+ny+1] ;}
        m := m + 1 ;

        writeln ( th[1],yy[m-1],uu[m-1],'  ',m) ;

        End ;
end ;
```

```
        for i := 1 to 250 do yy[i] := yy[i+200] ;
        savedata(yy,249) ;
        for i := 1 to 250 do uu[i] := uu[i+200] ;
        savedata(uu,249) ;

End .
```

โปรแกรมที่ 2

```
Program SimulationTest;
Const
          Norder   =   10  ;
          Nofy     =   2   ;
          Nofu     =   2   ;
          Nofn     =   0   ;
          Nstate   =   1   ;
          Error    =   0.01 ;
          Q0       =   0.1 ;{0.1steady state}
          B0       =   1 ;
          P0       =   2 ;   {5damping CLIMBING}
          T0       =   0.26 ; {0.1dec u & for smooth & sharp response}
          Lamda    =   0.98 ;
          Delta    =   6 ;
          Dr       =  0.707 ;
          Freq     =  0.055 ;
          Drf      =  0.8 ;
          Freqf    =  1.3 ;
          Delay    =   5 ;
          Max      =   10 ;
          DataMax  =  450 ;
          Npar     =   20 ;
          Noff     =   190 ;
          Nsum     =   210 ;
          Value    =   1905 ;{1 = 476.25}
          MaxU     =   4.0 ;
          Cv       =   100 ;
          Sigma    =   1 ;


Type
             Vec1   = Array [1..Npar] of Real ;
             Vec2   = Array [1..Noff] of Real ;
             Vec3   = Array [1..Nsum] of Real ;
          Estpartyp  = Record
             n,na   :  Integer ;
             ny     :  integer ;
             nu     :  Integer ;
             th     :  Vec1 ;
             fi     :  Vec1 ;
             phi    :  Vec1 ;
             diag   :  Vec1 ;
             offdiag : Vec2 ; { like P0 }
             End ;

           Clgroup   =  Record
             n      :  Integer ;
             ny     :  Integer ;
             nu     :  Integer ;
             nn     :  Integer ;
             fi     :  Vec1 ;
             phi    :  Vec1 ;
             th     :  Vec1 ;
```

```
            g      :  Vec1 ;
            s      :  Vec3 ;
            End ;
          Matrix   =  Array [1..4,1..DataMax] of Real ;
          Vector   =  Array [1..Max] of Real ;
          DataVec  =  Array [1..DataMax] of Real ;


Var
          uuu,yyy,us,uu,yy    :  DataVec ;
          fil,rr,tt,ss : Vec1 ;
          a,b      :  vector ;
          t,w,u,y  :  Real ;
          m,nn,mm  :  Integer ;
          i,naa,nb :  Integer ;
          clstate  :  Clgroup ;
          perr     :  Real ;
          da       :  Vec1 ;
          iou,ioy  :  Integer ;
          iof,ioc  :  Integer ;
          ub       :  Real    ;
{ee****}  tr,ee,kk :  Vec1 ;
          u0       :  Real ;
          pol      :  vec1 ;
          ff,tab   :  Vec1 ;
          flowi,ti,area,kkk,h : real ;


Function Control_UPCG ( uu,yy : DataVec ; th,tab : Vec1 ; w : real ; n : integer ) : Real ;
var
        u     :  real ;
        i     :  integer ;
begin
        u := 0 ;
        u := tab[4]   ;
        for i := 1 to Nofy do u := u - yy[n-i]*th[i] ;
        for i := 2 to Nofu+Delay do u := u - uu[n-i+1]*th[Nofy+i] ;
        u := u/th[Nofy+1] ;
        Control_UPCG := u ;
end ;


Begin
        With clstate Do
             begin
             flowi := 0.05 ;
             area := 3 ;
             h := 2 ;
             kkk := 0.05 ;
             ti := 30 ;
             For i := 1 to Nstate Do
                    a[i] := ti ;
```

```
ny := Nofy ;
nu := Nofu + delay ;
n := ny + nu + Nofn ;   { beware Nofn 0 Nofn = 0 }
        for i := 1 to Npar do
              begin
              uu[i] := 0 ;
              rr[i] := 0 ;
              tt[i] := 0 ;
              ss[i] := 0 ;
              uuu[i] := 0 ;
              yyy[i] := ti ;
              yy[i] := ti ;
              us[i] := 0 ;
              end ;
 t := 0 ;
 w := Cv ;
 y := 0 ;
 m := n + Delay ;
 yy[m] := ti ;
 InitialC ( clstate,10000 ) ;
 th[ny + 1] := T0 ;
 us[m] := Cv ;
 uu[m] := Cv ;
 writeln(n,ny,nu) ;
 for i := 1 to Npar do
       begin
       ee[i] := Cv ;
       kk[i] := 0 ;
       tab[i] := 0 ;
       ff[i] := 0 ;
       end ;
 tab[2] := - 2*exp(- DR*Freq*Delta)*cos(Freq*delta*sqrt(1-DR*DR)) ;
 tab[3] := exp(- 2*Dr*Freq*Delta) ;

 tab[4] := tab[1] + tab[2] + tab[3] ;
 for i := 1 to 4 do tab[i] := tab[i]*1000 ;
 for i := 1 to 3 do writeln(tab[i]) ;
 fil[2] := - 2*exp(- DRf*Freqf*Delta)*cos(Freqf*delta*sqrt(1-DRf*DRf)) ;
 fil[3] := exp(- 2*Drf*Freqf*Delta) ;
 fil[4] := 1 + fil[2] + fil[3] ;
 fil[1] := 1 / fil[4] ;
 fil[2] := fil[2]/fil[4] ;
 fil[3] := fil[3]/fil[4] ;
 fil[1] := 1 ;
 fil[2] := 0 ;
 fil[3] := 0 ;
{ repeat until keypressed ;}
 th[1] := 1 ;
 writeln(tab[1],tab[2],tab[3],SQRT(4)) ;
 While m < DataMax do
        Begin
        if m > 10 then
        u := Control_UPCG ( uu,yy,th,tab,w,m) ;
```

```
                if u > 4000 then u := 4000 ;
                if u < 0 then u := 0 ;
                uu[m] := u ;
                if m < 50 then uu[m] := cv - yy[m-1] ;
{                if m = 220 then h := 0.6*h ;}
                if m = 220 then kkk := 1.6*kkk ;
                if m = 250 then ti := 40 ;
                if m = 300 then ti := 20 ;
                if m = 350 then flowi := 1.1*flowi ;
                if m = 400 then flowi := 0.9*flowi ;
                GenFunConSys ( a,flowi,ti,uu[m-delay],kkk,area,h ) ;
                yy[m] := a[1] ;
                perr := tab[1]*yy[m] + tab[2]*yy[m-1] + tab[3]*yy[m-2];
                uuu[m] := (uu[m] - fil[2]*uuu[m-1] - fil[3]*uuu[m-2])/fil[1] ;
                yyy[m] := (yy[m] - fil[2]*yyy[m-1] - fil[3]*yyy[m-2])/fil[1] ;
                Sq ( uuu[m-Delay+1],-yyy[m-delay+1],perr,clstate ) ;
{                Writeln ( th[ny+1],th[2],yy[m],uu[m],'  ',m) ;}
                m := m + 1 ;
                End ;

            end ;
            for i := 1 to 250 do yy[i] := yy[i+200] ;
            savedata(yy,249);
            for i := 1 to 250 do uu[i] := uu[i+200] ;
            savedata(uu,249);
End .
```

โปรแกรมที่ 3

```
Program SimulationTest;
Const
          Norder   =  10  ;
          Nofy     =  2  ;
          Nofu     =  2  ;
          Nofn     =  0  ;
          Nstate   =  1  ;
          Error    =  0.01 ;
          Q0       =  0.1 ;{0.1steady state}
          P0       =  2 ;   {Sdamping CLIMBING}
          B0       =  1 ;
          T0       =  0.26 ; {0.1dec u & for smooth & sharp response}
          Lamda    =  0.98 ;
          Delta    =  6;
          Dr       =  0.707 ;
          Freq     =  0.055 ;
          Drf      =  0.707 ;
          Freqf    =  1.5 ;
          Delay    =  5 ;
          Max      =  10 ;
          DataMax  =  450 ;
          Npar     =  20 ;
          Noff     =  190 ;
          Nsum     =  210 ;
          Value    =  1905 ;{1 = 476.25}
          OneVol   =  476.25 ;
          MaxU     =  4.0 ;
          Cv       =  100 ;
          Sigma    =  1 ;


Type
            Vec1   = Array [1..Npar] of Real ;
            Vec2   = Array [1..Noff] of Real ;
            Vec3   = Array [1..Nsum] of Real ;

         Estpartyp  = Record
            n,na   :  Integer ;
            ny     :  integer ;
            nu     :  Integer ;
            th     :  Vec1 ;
            fi     :  Vec1 ;
            phi    :  Vec1 ;
            diag   :  Vec1 ;
            offdiag : Vec2 ; { like P0 }
            End ;

          Clgroup   =  Record
            n      :  Integer ;
            ny     :  Integer ;
            nu     :  Integer ;
            nn     :  Integer ;
            fi     :  Vec1 ;
```

```
            phi    : Vec1 ;
            th     : Vec1 ;
            g      : Vec1 ;
            s      : Vec3 ;
            End ;
          Matrix   = Array [1..4,1..DataMax] of Real ;
          Vector   = Array [1..Max] of Real ;
          DataVec  = Array [1..DataMax] of Real ;


procedure Multipole ( var a,b,c : vec1 ; na,nb : integer ) ;
var
        i,j : integer ;
begin
        for i := 1 to na + nb do c[i] := 0 ;
        for i := 1 to nb do
             begin
             for j := 1 to na do
                  c[j+i-1] := c[j+i-1] + b[i]*a[j] ;
             end ;
{             writeln ;
        for i := 1 to n  do write(a[i]) ;
        writeln ;
        for i := 1 to n  do write(b[i]) ;
        writeln ;
        for i := 1 to nb+na-1 do write(c[i]) ;
        writeln ;}
end ;
{ procedure Multiloop ( value : vec1 ; n : integer ; var tr : vec1 );
var
        a,b,c : vec1 ;
        i,j : integer ;
begin
        for i := 1 to n+1 do
        begin
        a[i] := 0 ;
        b[i] := 0 ;
        end ;
        a[1] := 1 ;
        b[1] := 1 ;
        b[2] := pol[1] ;
        for i := 2 to n do
             begin
             a[2] := pol[i];
             Multipole ( a,b,c,2,i ) ;
             for j :=1 to n+1 do b[j] := c[j] ;
             end ;
        for i := 2 to n+1 do tr[i-1] := b[i] ;
end ;}
procedure Case1 ( var tr,a,kk : vec1 ) ;
Var
        i,j : integer ;
begin
```

```
        { na=2,nb=0,ns=2,k=1,nqr=1 }
        kk[1] := (tr[1]-a[1]+1)/a[3] ;
        kk[2] := (tr[2]-a[2]+a[1])/a[3] ;
        kk[3] := (tr[3]+a[2])/a[3] ;
end;
procedure Case2 ( var tr,aa,g,kk : vec1 ) ;
var
        i,j : integer ;
        min : integer ;
        a   : Vec1 ;
begin
        for i := 1 to Nofy do a[i] := aa[i] ;
        for i := 1 to Delay - 1 do
              begin
              g[i] := tr[i] ;
              min := Nofy ;
              if (i-1) < min then min := i - 1 ;
              if i > 1 then
                   for j := 1 to min do
                        begin
                        g[i] := g[i] - g[i-j]*a[j] ;
                        end;
              if i <  Nofy  then g[i] := g[i] - a[i] ;

{              writeln(i,'  ',g[i]) ;}
              end ;

        for i := 1 to Nofy  do
              begin
              kk[i] := tr[Delay+i-1] ;
              for j := i to Nofy do kk[i] := kk[i] - g[Delay-j+i-1]*a[j] ;
              kk[i] := kk[i]/aa[Nofy+1] ;
{              writeln(kk[i]);        }
              end;
 end ;
procedure Case2W ( var tr,aa,ff,kk : vec1 ) ;
var
        i,j : integer ;
        min : integer ;
        g,a   : Vec1 ;
begin
        for i := 1 to Npar do a[i] := 0 ;
        for i := 2 to Nofy do a[i] := aa[i]-aa[i-1] ;
        a[1] := aa[1] - 1 ;
        a[Nofy + 1] := - aa[Nofy] ;{ Nofy := Nofy + 1 }
        for i := 1 to Delay - 1 do
              begin

              min := Nofy + 1 ;{}
              if (i-1) < min then min := i - 1 ;
              if i > 1 then
                   for j := 1 to min do
                        begin
```

```
                    g[i] := g[:] - g[i-j]*a[j] ;
                    end;
               if i <= Nofy + 1  then g[i] := g[i] - a[i] ;{}
               ff[i] := g[i] ;

{               writeln(i,'  ',g[i]) ;}
               end ;

          for i := 1 to Nofy + 1 do
               begin
               kk[i] := tr[Delay+i-1] ;
               for j := i to Nofy+1 do kk[i] := kk[i] - g[Delay-j+i-
               1]*a[j] ;{}
               kk[i] := kk[i]/aa[Nofy+1] ;
{               writeln(kk[i]);}
               end;
     end ;

     procedure CaseJA ( var tab,th,rr,tt,ss : vec1 ) ;
     var
          i,j : integer ;
          min : integer ;
          a,amaO   : Vec1 ;
          Z1,Z2    : REAL ;
     begin
          for i := 1 to Npar do
               begin
               tt[i] := 0 ;
               A[I] := 0 ;
               amaO[i] := 0 ;
               end ;

          FOR I := 1 TO 2 DO AMAO[I] := TAB[I] ;

{              Z1 := - 0.98450386638 ;
               Z2 := 0 ;
               AMAO[1] := TAB[1] + Z1 ;
               AMAO[2] := TAB[2] + Z1*TAB[1] + Z2 ;
               AMAO[3] := Z1*TAB[2] + Z2*TAB[1] ;
               AMAO[4] := Z2*TAB[2] ;
               tt[1] := tab[Nofy+1] ;
               TT[2] := TAB[NOFY+1]*Z1 ;
               TT[3] := TAB[nOFY+1]*Z2 ;}
               for i := 1 to Nofy do a[i] := th[i] ;
               for i := 1 to Delay - 1 do
                    begin
                    rr[i] := amaO[i] ;
                    min := Nofy ;
                    if (i-1) < min then min := i - 1 ;
                    if i > 1 then
                         for j := 1 to min do
                              begin
                              rr[i] := rr[i] - rr[i-j]*a[j] ;
```

```
                    end;
            if i <= Nofy  then rr[i] := rr[i] - a[i] ;

             writeln(i,'  ',rr[i]) ;}
            end ;

        for i := 1 to Nofy  do
            begin
            ss[i] := ama0[Delay+i-1] ;
            for j := i to Nofy do ss[i] := ss[i] - rr[Delay-j+i-1]*a[j] ;
             writeln(ss[i]);}
            end;

 end ;


 Var
            us,uu,yy,uuu,yyy    :  DataVec ;
            r1,b1       :  vec1 ;
            rr,tt,ss : Vec1 ;
            a,b      :  vector ;
            t,w,u,y  :  Real ;
            m,nn,mm  :  Integer ;
            i,naa,nb :  Integer ;
            clstate  :  Clgroup ;
            perr     :  Real ;
            da       :  Vec1 ;
            iou,ioy  :  Integer ;
            iof,ioc  :  Integer ;
            ub       :  Real    ;
{ee****}    tr,ee,kk :  Vec1 ;
            u0       :  Real ;
            pol      :  vec1 ;
            ff,tab,fil   :  Vec1 ;
            flowi,ti,kkk,area,h : real ;


 Function Control_UPA ( uu,yy : DataVec ; th,rr,tt,ss : Vec1 ; n : integer ) : Real ;
 Var
          u        :  Real ;
          i        :  Integer ;
          aux      :  Vec1 ;
 Begin
          u := 0 ;
          for i := 1 to Nofy do u := u - ss[i]*yy[n-i] ;
          for i := 1 to 3 do u := u + tt[i]*Cv ;
          for i := 2 to Delay+Nofu do
          u := u - rr[i]*uu[n-i+1] ;


          Control_UPA := u ;
 End ;
```

```
Begin
        With clstate Do
              begin
              flowi := 0.05 ;
              ti := 30 ;
              kkk := 0.05 ;
              area := 3 ;
              h := 2 ;

              For i := 1 to Nstate Do
                     a[i] := ti ;
              ny := Nofy ;
              nu := 1 + Nofu ;
              n := ny + nu + Nofn ;   { beware Nofn # Nofn = 0 }
                     for i := 1 to Npar do
                            begin
                            uu[i] := 0 ;
                            rr[i] := 0 ;
                            tt[i] := 0 ;
                            r1[i] := 0 ;
                            b1[i] := 0 ;
                            ss[i] := 0 ;
                            uuu[i] := 0 ;
                            yyy[i] := ti ;

                            yy[i] := ti ;
                            us[i] := 0 ;
                            end ;
              t := 0 ;
              w := Cv ;
              y := 0 ;
              m := n + Delay ;
              yy[m] := ti ;
              InitialC ( clstate,10000 ) ;
              th[ny + 1] := T0 ;
              us[m] := Cv ;
              rr[1] := 1 ;
              uu[m] := Cv ;
              writeln(n,ny,nu) ;
              for i := 1 to Npar do
                    begin
                    ee[i] := Cv ;
                    kk[i] := 0 ;
                    tab[i] := 0 ;
                    ff[i] := 0 ;
                    end ;
              tab[1] := - 2*exp(- DR*Freq*Delta)*cos(Freq*delta*sqrt(1-DR*DR)) ;
              tab[2] := exp(- 2*Dr*Freq*Delta) ;
              tab[3] := 1 + tab[1] + tab[2] ;

              fil[2] := - 2*exp(- DRf*Freqf*Delta)*cos(Freqf*delta*sqrt(1-DRf*DRf)) ;
              fil[3] := exp(- 2*Drf*Freqf*Delta) ;
```

```
                fil[4] := 1 + fil[2] + fil[3] ;
                fil[1] := 1 / fil[4] ;
                fil[2] := fil[2]/fil[4] ;
                fil[3] := fil[3]/fil[4] ;
                fil[1] := 1 ;
                fil[2] := 0 ;
                fil[3] := 0 ;
{                repeat until keypressed ;}
                writeln(tab[1],tab[2],tab[3],SQRT(4)) ;
                While m < DataMax do
                      Begin
                      u := Control_UPA ( uu,yy,th,rr,tt,ss,m) ;
                      if u > 4000 then u := 4000 ;
                      if u < 0 then u := 0 ;
                      uu[m] := u ;
                      if m < 50 then uu[m] := Cv - yy[m-1] ;
                      if m = 220 then h := h*0.6 ;
{                      if m = 220 then kkk := 1.6*kkk ;}
                      if m = 250 then ti := 40  ;
                      if m = 300 then ti := 20 ;
                      if m = 350 then flowi := flowi*1.1 ;
                      if m = 400 then flowi := 0.9*flowi ;
                      GenFunConSys ( a,flowi,ti,uu[m-delay],kkk,area,h ) ;
                      yy[m] := a[1] ;
                      uuu[m] := (uu[m] - fil[2]*uuu[m-1] - fil[3]*uuu[m-2])/fil[1] ;
                      yyy[m] := (yy[m] - fil[2]*yyy[m-1] - fil[3]*yyy[m-2])/fil[1] ;
                      perr := yyy[m] ;
                      Sq ( uuu[m-Delay+1],yyy[m],perr,clstate ) ;
                      Writeln ( th[1],yyy[m],uuu[m],'  ',m) ;
                      m := m + 1 ;
                      Case3A(tab,th,r1,tt,ss) ;
                      for i := 1 to n do b1[i] := 0 ;
                      for i := 1 to nu do b1[i] := th[ny+i] ;
                      for i := 1 to delay - 1 do r1[delay+1-i] := r1[delay-i] ;
                      r1[1] := 1 ;
                      Multipole(b1,r1,rr,nu,delay) ;
                      end ;
                end ;
                for i := 1 to 250 do yy[i] := yy[i+200] ;
                savedata(yy,249);
                for i := 1 to 250 do uu[i] := uu[i+200] ;
                savedata(uu,249);
End .
```

โปรแกรมที่ 4

```
Program SimulationTest;
Const
          Norder   =  10  ;
          Nofy     =  2  ;
          Nofu     =  0  ;
          Nofn     =  0  ;
          Nstate   =  1  ;
          Error    =  0.01 ;
          W0       =  0.1 ;{0.1steady state}
          P0       =  2 ;   {5damping CLIMBING}
          T0       =  0.26 ; {0.1dec u & for smooth & sharp response}
          Lamda    =  0.98 ;
          B0       =  1 ;
          Delta    =  6 ;
          Dr       =  0.9 ;
          Freq     =  0.1 ;
          Delay    =  5 ;
          Max      =  10 ;
          DataMax  =  450 ;
          Npar     =  20 ;
          Noff     =  190 ;
          Nsum     =  210 ;
          Value    =  1905 ;{1 = 476.25}
          OneVol   =  476.25 ;
          MaxU     =  4.0 ;
          Cv       =  100 ;
          Sigma    =  1 ;


Type
            Vec1   = Array [1..Npar] of Real ;
            Vec2   = Array [1..Noff] of Real ;
            Vec3   = Array [1..Nsum] of Real ;

         Estpartyp  = Record
            n,na   :  Integer ;
            ny     :  integer ;
            nu     :  Integer ;
            th     :  Vec1 ;
            fi     :  Vec1 ;
            phi    :  Vec1 ;
            diag   :  Vec1 ;
            offdiag : Vec2 ; { like P0 }
            End ;

          Clgroup   =  Record
            n      :  Integer ;
            ny     :  Integer ;
            nu     :  Integer ;
            nn     :  Integer ;
            fi     :  Vec1 ;
            phi    :  Vec1 ;
            th     :  Vec1 ;
```

```
            g      :  Vec1 ;
            s      :  Vec3 ;
            End ;
          Matrix   =  Array [1..4,1..DataMax] of Real ;
          Vector   =  Array [1..Max] of Real ;
          DataVec  =  Array [1..DataMax] of Real ;


function pidcontroller ( u0 : real ; var pid,e : Vec1 ) : real ;
var
        i   :  integer ;
        u   :  real ;
begin
        u := u0 ;
        for i := 1 to 3 do
             u := u + pid[i] * e[i] ;
        pidcontroller := u;
end ;

procedure Newton ( th : vec1 ; var w,hsdelay : real ) ;
var
         i,j   : integer ;
         a,eps  : real    ;
         y : datavec ;
function  fx ( th : vec1 ; w : real ) : real ;
var
         a,b      :  real ;
begin
         a := (1 + th[1]*cos(w) + th[2]*cos(2*w))*sin(hsdelay*w) ;
         b := (th[1]*sin(w) + th[2]*sin(2*w))*cos(hsdelay*w) ;
         fx := a - b ;
end ;
function dfx ( th : vec1 ; w : real ) : real ;
var
         a,b      :  real ;
         c        :  real ;
begin
         a := hsdelay*(1 + th[1]*cos(w) + th[2]*cos(2*w))*cos(hsdelay*w) ;
         b := ( - th[1]*sin(w) - 2*th[2]*sin(2*w))*sin(hsdelay*w) ;
         c := - hsdelay*(th[1]*sin(w) + th[2]*sin(2*w))*sin(hsdelay*w) ;

         dfx := a+b-c   ;
end ;
begin
         eps := 0.000001 ;
{         for i := 1 to 250 do y[i] := fx(th,i/250) ;
         savedata(y,249) ;}
         a := 11/7/delay ;{0.16-0.24,0.48-0.55}
         i := 1 ;
         w := 1 ;
         while abs((a-w)/w) > eps do
               begin
               w := a ;
```

```
                a := w - fx(th,w)/dfx(th,w) ;
{                writeln(a) ;}
                i := i + 1 ;
                if i > 20 then
                    begin
                    a := w ;
                    writeln('        iteration exceed ! ') ;
                    end ;
                end ;
 end ;


procedure Nyquist ( var th : vec1 ; var k,w,hsdelay : real ; ny,nu : integer ) ;
var
        i,j    :  integer ;
        a,b    :  real ;
        c,d    :  real ;
begin
        newton (th,w,hsdelay) ;
{        c := (1 + th[1]*cos(w) + th[2]*cos(2*w))*cos(hsdelay*w) ;
        c := c + (th[1]*sin(w) + th[2]*sin(2*w))*sin(hsdelay*w) ;}

        d := 1 + th[1]*th[1] + th[2]*th[2] + 2*th[1]*(th[2]+1)*cos(w) ;
        d := d + 2*th[2]*cos(2*w) ;
        d := sqrt(d) ;
        k := - d / (th[3]*c) ;

        w := w/delta ;
end ;




procedure tunz ( var th,pid : vec1 ; hsdelay : real ; ny,nu : integer ) ;
var
        i,j     :  integer ;
        k,w     :  real    ;
        gain    :  real    ;
        reset   :  real    ;
        rrate   :  real    ;
        kpid    :  real    ;
        t0      :  real    ;
        a1,a2   :  real    ;
begin
         Nyquist ( th,k,w,hsdelay,ny,nu ) ;
         gain := (k/1.66) ;
         while w > 22/7 do
                w := w - 22/7 ;
         t0 := (2*22/7)/w ;
         writeln(gain,t0,w,k) ;
         reset := 2/t0 ;
```

```
          rrate := t0/8 ;
          kpid := 1 + Delta * reset + rrate/Delta ;
          a1 := (Delta + 2*rrate)/(Delta*kpid) ;
          a2 := rrate/(Delta*kpid) ;
          pid[1] := kpid*gain ;
          pid[2] := - a1*kpid*gain ;
          pid[3] := a2*kpid*gain ;
end ;


Var
          us,uu,yy    :  DataVec ;
          rr,tt,ss : Vec1 ;
          a,b      :  vector ;
          hsdelay : real ;
          pid      :  vec1;
          t,w,u,y  :  Real ;
          m,nn,mm  :  Integer ;
          i,naa,nb :  Integer ;
          clstate  :  Clgroup :
          perr     :  Real :
          da       :  Vec1 :
          iou,ioy  :  Integer ;
          iof,ioc  :  Integer ;
          ub       :  Real    ;
{ee****}  tr,ee,kk :  Vec1 ;
          u0       :  Real :
          pol      :  vec1 ;
          ff,tab   :  Vec1 :
          flowi,ti,kkk,area,h : real ;


Begin
          With clstate Do
               begin
               flowi := 0.05 ;
               ti := 30 ;
               h := 2 ;
               area := 3 ;
               kkk := 0.05 ;
               For i := 1 to Nstate Do
                      a[i] := ti ;
               ny := Nofy ;
               nu := 1 + Nofu ;
               n := ny + nu + Nofn ;   { beware Nofn @ Nofn = 0 }
                       for i := 1 to Npar do
                             begin
                             uu[i] := 0 ;
                             rr[i] := 0 ;
                             tt[i] := 0 ;
                             ss[i] := 0 ;
                             yy[i] := ti ;
                             pid[i] := 0 ;
```

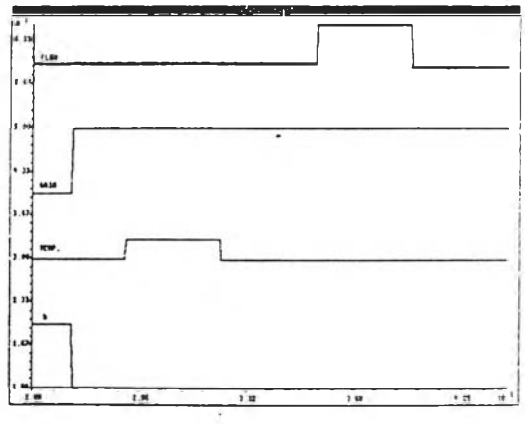
```
                us[i] := 0 ;
                end ;
        t := 0 ;
        w := Cv ;
        y := 0 ;
        m := n + delay ;
        yy[m] := ti ;
        InitialC ( clstate,10000 ) ;
        hsdelay := delay ;
{        th[ny + 1] := 70 ;
        th[1] := -1.5 ;
        th[2] := 0.5 ;
        th[3] := 0.25 ;
        th[4] := 0 ;
        tunz(th,pid,hsdelay,ny,nu) ;
        writeln(n,ny,nu,w) ;}
        While m < DataMax do
                Begin
                ee[3] := ee[2] ;
                ee[2] := ee[1] ;
                ee[1] := Cv - yy[m-1] ;
                uu[m] := ee[1] ;

                if m > 40 then tunz(th,pid,hsdelay,ny,nu) ;
                if m > 50 then uu[m] := pidcontroller ( uu[m-1],pid,ee );
                if uu[m] > 4000 then uu[m] := 4000 ;
                if uu[m] < 0 then uu[m] := 0 ;
                genfunconsys ( a,flowi,ti,uu[m-delay],kkk,area,h ) ;
                if m = 220 then h := 0.6*h ;
{                 if m = 220 then kkk := 1.6*kkk ;}
                if m = 250 then ti := 40  ;
                if m = 300 then ti := 20 ;
                if m = 350 then flowi := 1.1*flowi ;
                if m = 400 then flowi := 0.9*flowi ;
                yy[m] := a[1] ;
                perr := yy[m] ;
                Sq ( uu[m-Delay+1],yy[m],perr,clstate ) ;
                writeln(th[1],th[2],uu[m],yy[m],'  ',m) ;
                m := m + 1 ;
                end ;
        end ;
        for i := 1 to 250 do yy[i] := yy[i+200] ;
        savedata(yy,249);
        for i := 1 to 250 do uu[i] := uu[i+200] ;
        savedata(uu,249);
End .
```

ภาคผนวก ข

แสดงลักษณะของผลตอบของกระบวนการในการจำลอง

ระบบที่ใช้ในการจำลองได้แก่ ระบบควบคุมอุณหภูมิซึ่งได้กล่าวไว้ในบทที่ 3 ในการควบคุมที่จำลองขึ้นดังกล่าวต้องการให้อุณหภูมิของน้ำในกระบวนการมีค่าเท่ากับ 100°C โดยมีเงื่อนไขให้ F_i, T_i เป็นสิ่งรบกวนที่ขึ้นในกระบวนการที่เวลาต่าง ๆ และ K,h เป็นตัวแปรที่ทำให้พารามิเตอร์ของกระบวนการเกิดการเปลี่ยนแปลงที่เวลาต่าง ๆ เช่นกันซึ่งแสดงไว้ในรูปที่ ข.1



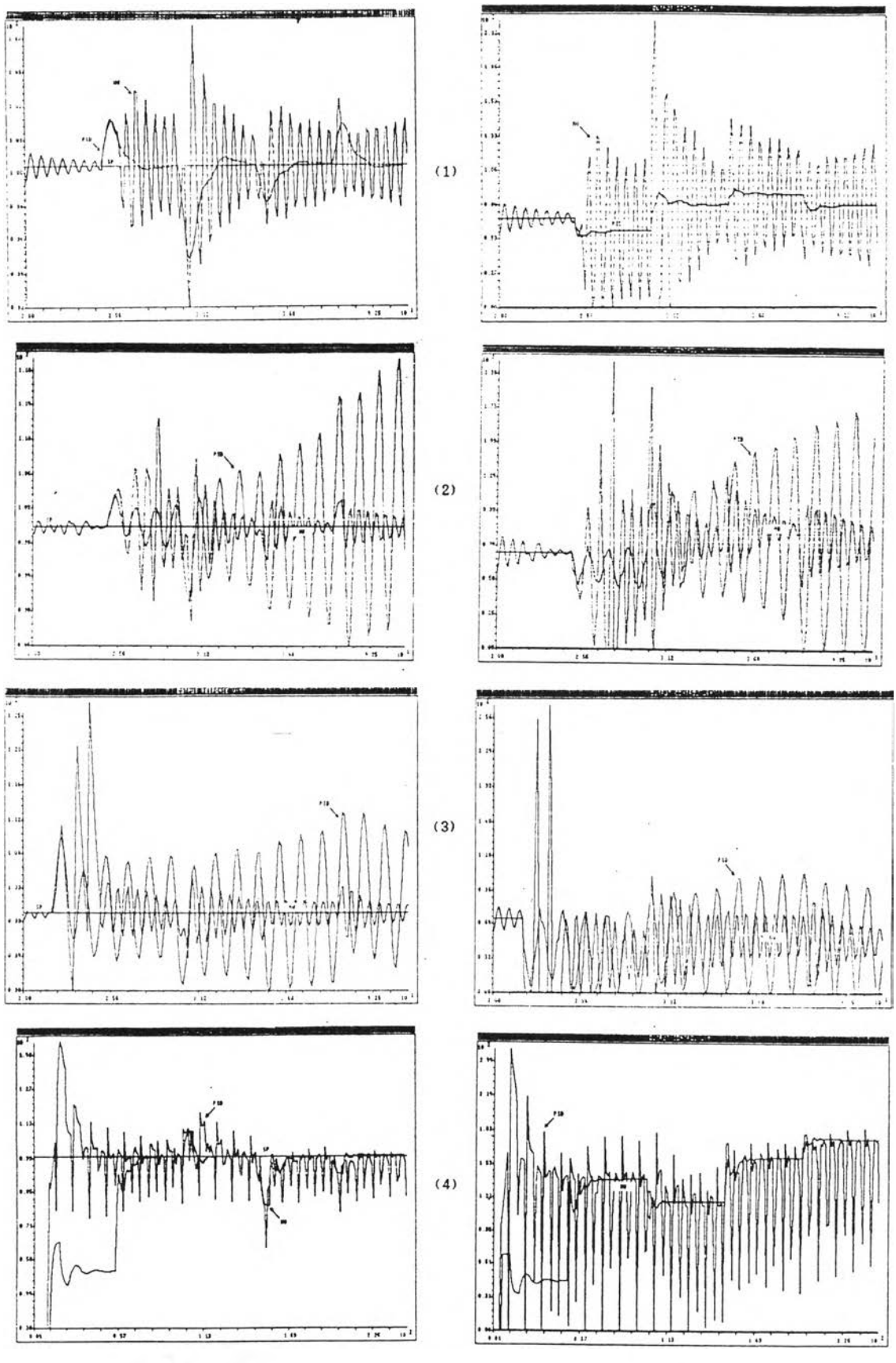
รูปที่ ข.1 แสดงการเปลี่ยนแปลงค่าตัวแปรที่เวลาต่าง ๆ

จากบทที่ 3 จะเห็นว่าเมื่อระบบมีการเปลี่ยนแปลงพารามิเตอร์และสิ่งรบกวนเกิดขึ้นแล้ว การควบคุมอุณหภูมิในกระบวนการที่จำลองด้วยตัวควบคุมแบบ PID ไม่สามารถควบคุมอุณหภูมิให้มีค่าตามต้องการได้ ซึ่งการเปลี่ยนแปลงพารามิเตอร์ของการควบคุมได้แบ่งออกเป็น 2 กรณี

1. การเปลี่ยนแปลงอัตราขยายของกระบวนการ
2. การเปลี่ยนแปลงค่า Time constant ของกระบวนการ

แต่เมื่อใช้กระบวนการควบคุมแบบจูนปรับตัวเองแทนการควบคุมแบบ PID พบว่าการควบคุมแบบจูนปรับตัวเองสามารถควบคุมอุณหภูมิได้ตามข้อกำหนดที่วางไว้ ซึ่งผลตอบของกระบวนการและตัวควบคุมในการจำลองโดยใช้ตัวควบคุมชนิดต่าง ๆ จะแสดงไว้ในรูปต่อไปนี้

<u>หมายเหตุ</u> แกนตั้งของผลตอบของกระบวนการเป็นค่าของอุณหภูมิที่วัดได้มีหน่วยเป็นองศาเซลเซียล แกนนอนเป็นจำนวนครั้งของการสุ่ม ซึ่งคาบของการสุ่มเท่ากับ 6 วินาที



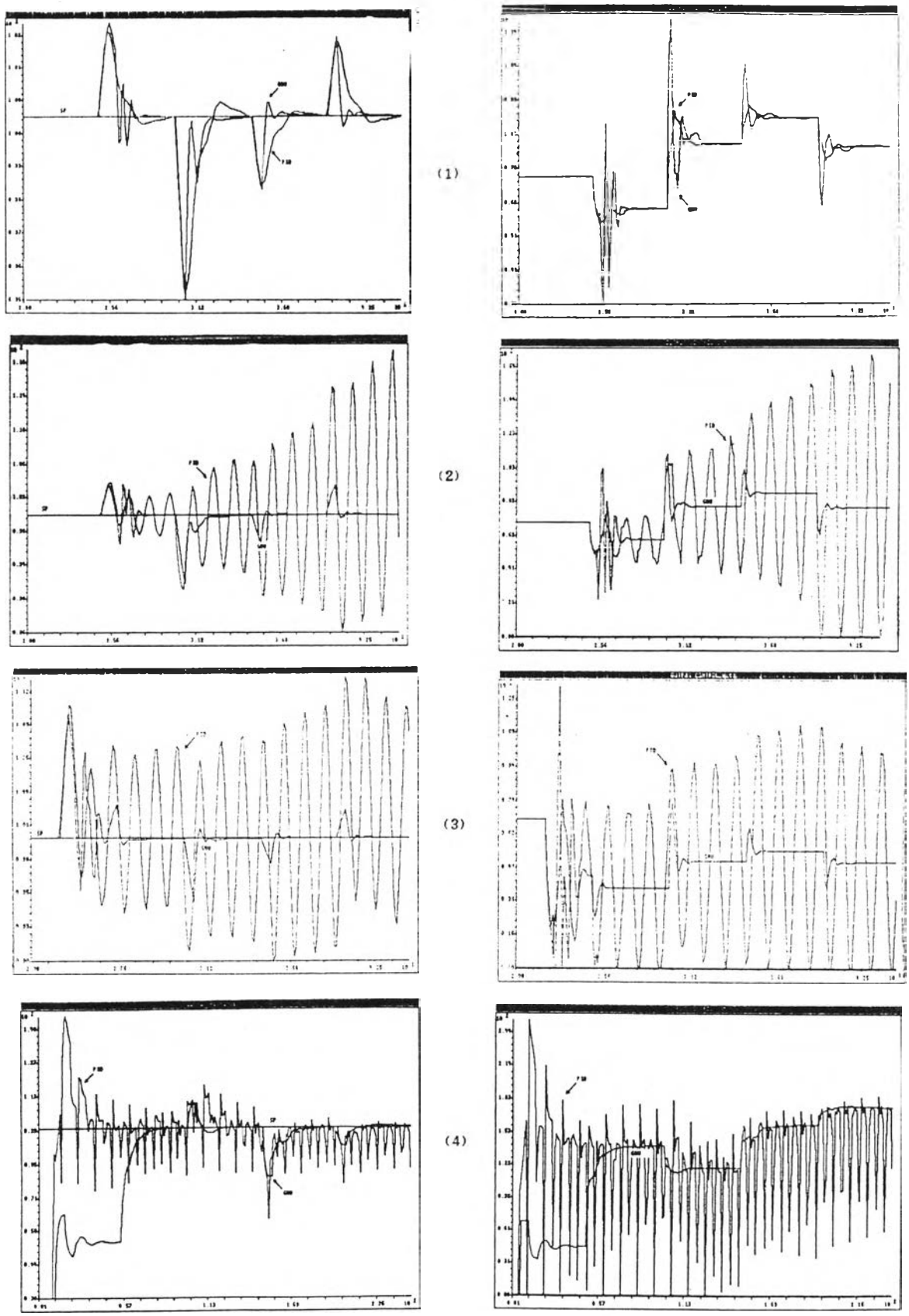
รูปที่ ข.2 แสดงผลตอบของกระบวนการ (ขวา) และผลตอบของตัวควบคุม (ซ้าย)
โดยใช้การควบคุมแบบ Minimum variance control เปรียบเทียบกับ PID

(1) เมื่อกระบวนการมีสิ่งรบกวนเกิดขึ้นอย่างเดียว

(2) เมื่อความสูงลดลง 40% และมีสิ่งรบกวน

(3) เมื่อค่าสัมประสิทธิ์เพิ่มขึ้น 60% และมีสิ่งรบกวน

(4) เมื่ออัตราการไหล F_1 เพิ่มขึ้น 2 เท่า ความสูงลดลง 90% และมีสิ่งรบกวน



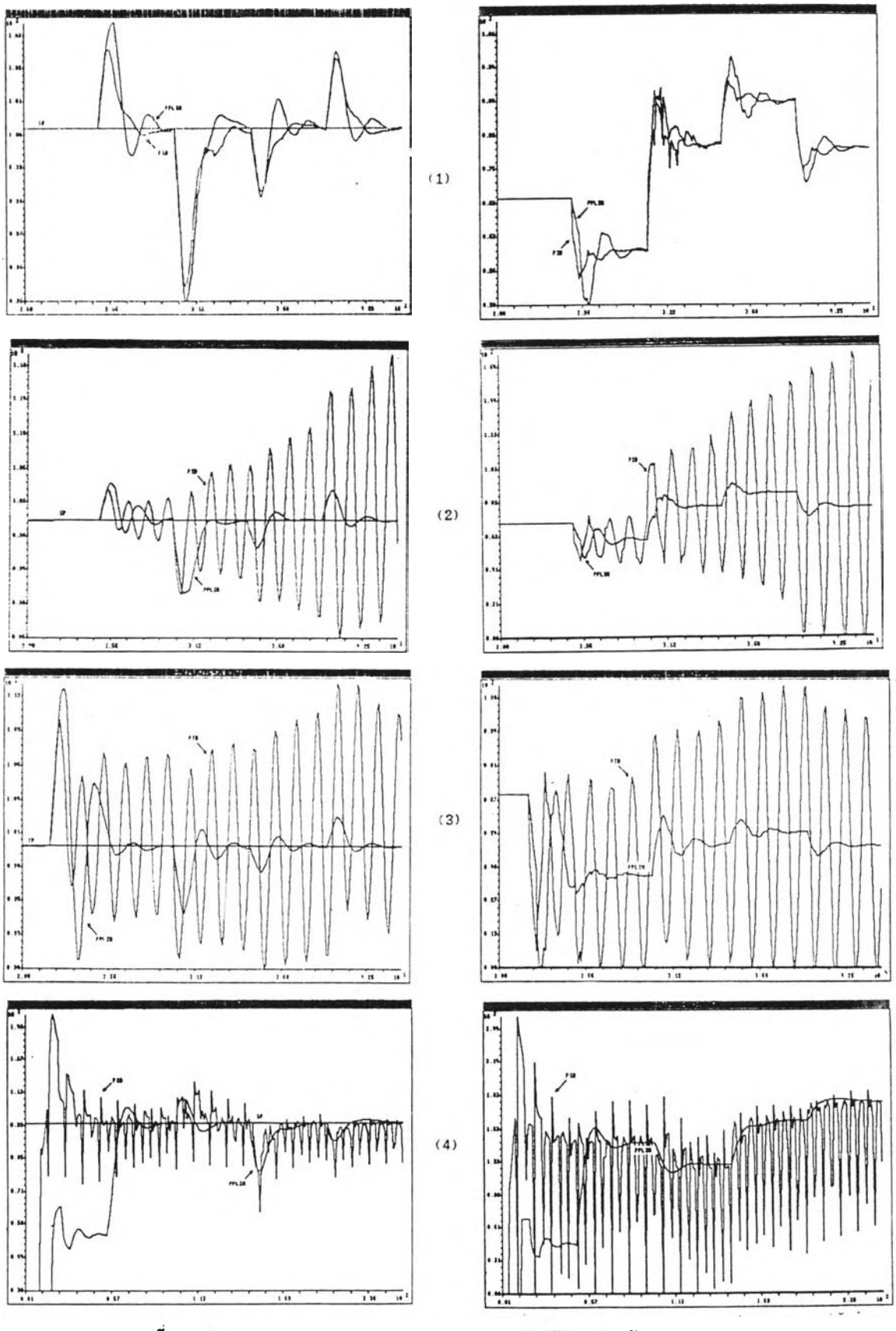
รูปที่ ข.3 แสดงผลตอบของกระบวนการ (ขวา) และผลตอบของตัวควบคุม (ซ้าย)
โดยใช้การควบคุมแบบ Generalized minimum variance เปรียบเทียบกับ PID

(1) เมื่อกระบวนการมีสิ่งรบกวนเกิดขึ้นอย่างเดียว

(2) เมื่อความสูงลดลง 40% และมีสิ่งรบกวน

(3) เมื่อค่าสัมประสิทธิ์เพิ่มขึ้น 60% และมีสิ่งรบกวน

(4) เมื่ออัตราการไหล F_1 เพิ่มขึ้น 2 เท่า ความสูงลดลง 90% และมีสิ่งรบกวน



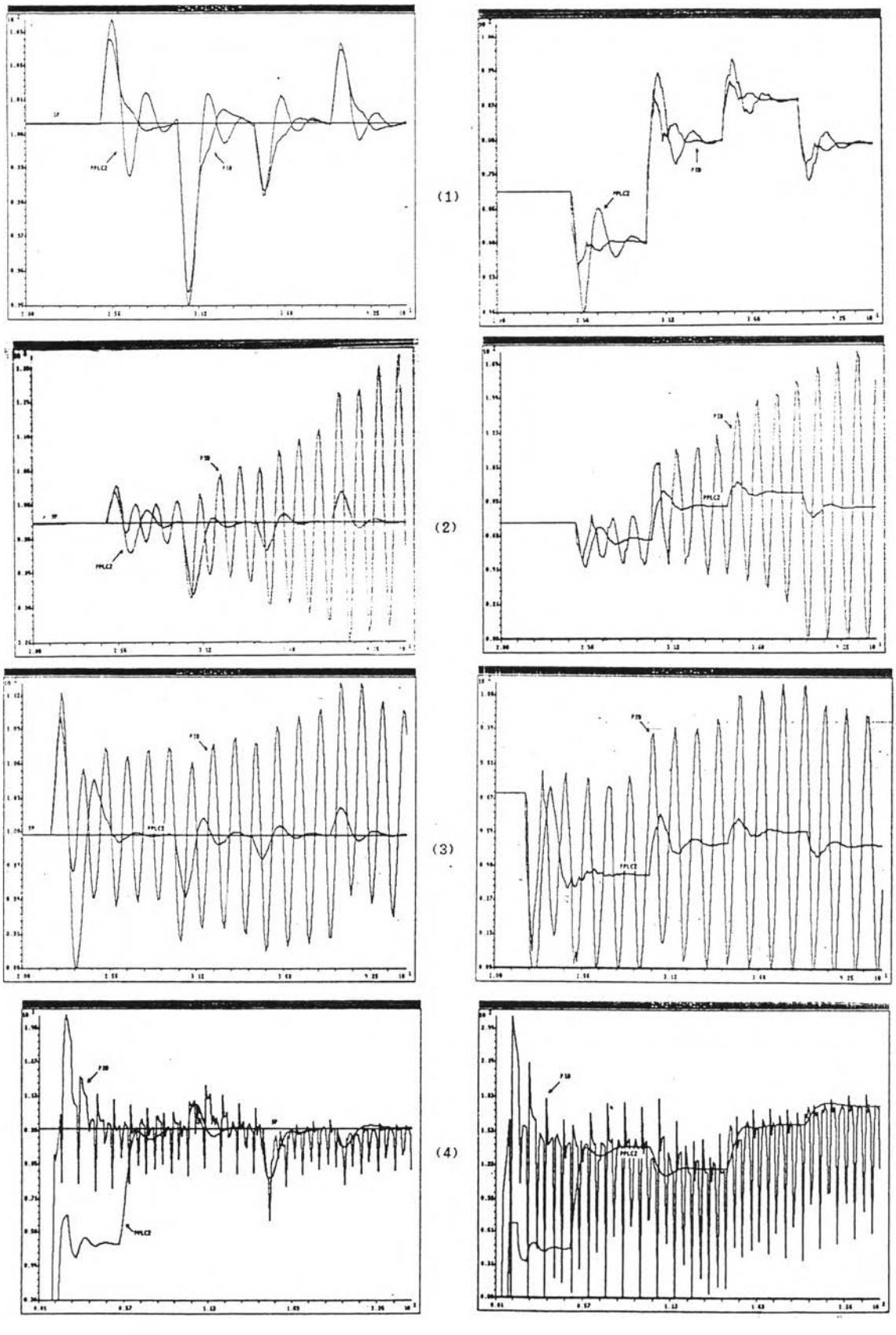
รูปที่ ฃ.4 แสดงผลตอบของกระบวนการ (ขวา) และผลตอบของตัวควบคุม (ซ้าย)
โดยใช้การควบคุมแบบ Pole placement method(Implicit) เปรียบเทียบกับ PID

(1) เมื่อกระบวนการมีสิ่งรบกวนเกิดขึ้นอย่างเดียว

(2) เมื่อความสูงลดลง 40% และมีสิ่งรบกวน

(3) เมื่อค่าสัมประสิทธิ์เพิ่มขึ้น 60% และมีสิ่งรบกวน

(4) เมื่ออัตราการไหล F_1 เพิ่มขึ้น 2 เท่า ความสูงลดลง 90% และมีสิ่งรบกวน



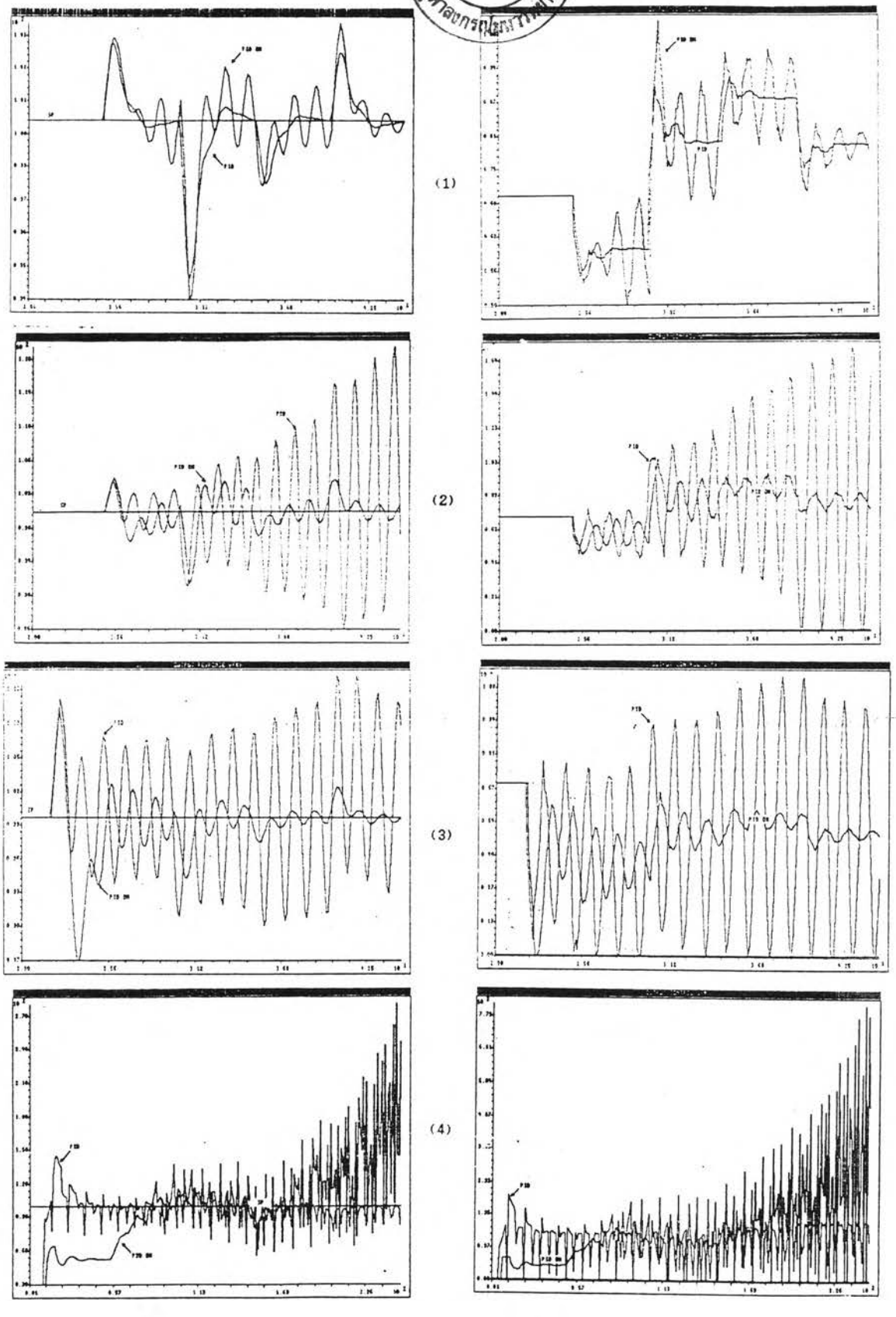
รูปที่ ข.5 แสดงผลตอบของกระบวนการ (ขวา) และผลตอบของตัวควบคุม (ซ้าย)
โดยใช้การควบคุมแบบ Pole placement method (Explicit) เปรียบเทียบกับ PID

(1) เมื่อกระบวนการมีสิ่งรบกวนเกิดขึ้นอย่างเดียว

(2) เมื่อความสูงลดลง 40% และมีสิ่งรบกวน

(3) เมื่อค่าสัมประสิทธิ์เพิ่มขึ้น 60% และมีสิ่งรบกวน

(4) เมื่ออัตราการไหล F_i เพิ่มขึ้น 2 เท่า ความสูงลดลง 90% และมีสิ่งรบกวน



รูปที่ ข.6 แสดงผลตอบของกระบวนการ (ขวา) และผลตอบของตัวควบคุม (ซ้าย)
โดยใช้การควบคุมแบบ PID (on-line) เปรียบเทียบกับ PID

(1) เมื่อกระบวนการมีสิ่งรบกวนเกิดขึ้นอย่างเดียว

(2) เมื่อความสูงลดลง 40% และมีสิ่งรบกวน

(3) เมื่อค่าสัมประสิทธิ์เพิ่มขึ้น 60% และมีสิ่งรบกวน

(4) เมื่ออัตราการไหล F_1 เพิ่มขึ้น 2 เท่า ความสูงลดลง 90% และมีสิ่งรบกวน

ภาคผนวก ค

แสดงลักษณะของผลตอบของการควบคุมอุณหภูมิ

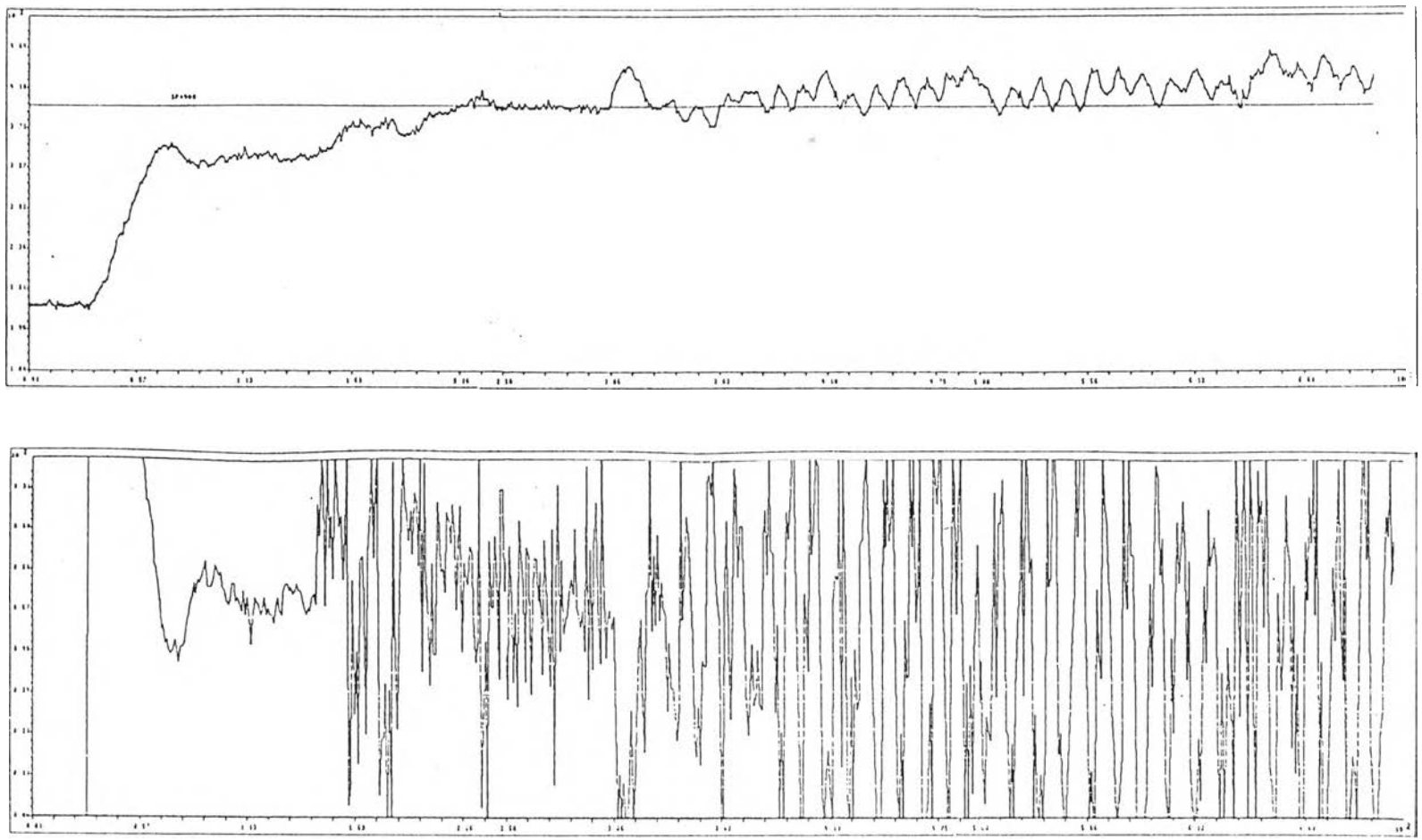
จากบทที่ 4 การควบคุมอุณหภูมิมีข้อกำหนดคือต้องการให้น้ำในถังมีอุณหภูมิประมาณ 37.5°C หรือ 400 หน่วยของค่าในวงจร A/D โดยมีเงื่อนไขว่าระบบควบคุมอุณหภูมิจะต้องเข้าสู่สภาวะคงตัวแล้วจึงทำการเปลี่ยนแปลงระดับความสูงของน้ำในถังจาก 6 เซ็นติเมตร ลดลงมาเป็น 3 เซ็นติเมตร ประมาณการสุ่มที่ 300 และกำหนดให้มีสิ่งรบกวนที่มีลักษณะเป็น Impulse หลังจากการเปลี่ยนแปลงระดับความสูงประมาณการสุ่มที่ 500 ด้วยการเทน้ำเย็นลงในถัง จากการทดลองได้ใช้ตัวควบคุมต่าง ๆ ดังนี้คือ

1. ควบคุมแบบ Conventional PID
2. ควบคุมแบบ Minimum variance control
3. ควบคุมแบบ Generalized minimum variance control
4. ควบคุมแบบ Pole placement method
5. ควบคุมแบบ PI (on-line) using Ziegler & Nichols tuning rules

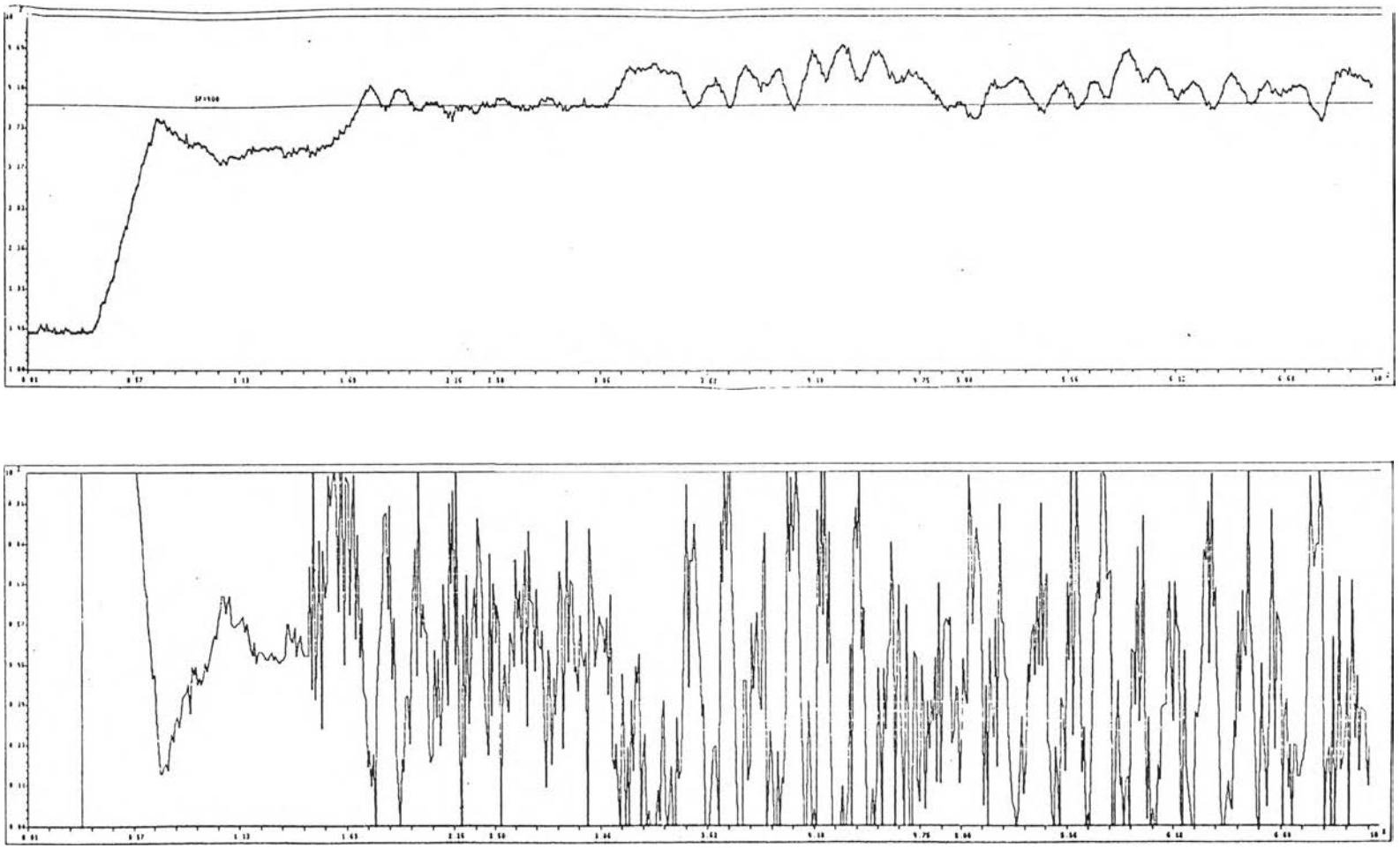
ในการทดลองควบคุมอุณหภูมิได้ทดลองใช้ตัวควบคุมชนิดต่าง ๆ เหล่านี้หลายครั้งเพื่อศึกษาเสถียรภาพและผลของการเปลี่ยนค่าพารามิเตอร์ของตัวควบคุมต่าง ๆ ซึ่งผลการทดลองควบคุมอุณหภูมิแสดงในรูปต่อไปนี้

หมายเหตุ 1. แกนตั้งของผลตอบของกระบวนการเป็นค่าเอาต์พุตของ A/D มีหน่วยเป็นระดับของเอาต์พุตของ A/D (มีค่าตั้งแต่ 0 - 4000 หน่วย ซึ่งค่าที่ 400 หน่วยจะประมาณ 37.5°C)
แกนนอนเป็นแกนเวลามีหน่วยเป็นจำนวนครั้งของการสุ่ม ซึ่งคาบของการสุ่มเท่ากับ 5 วินาที

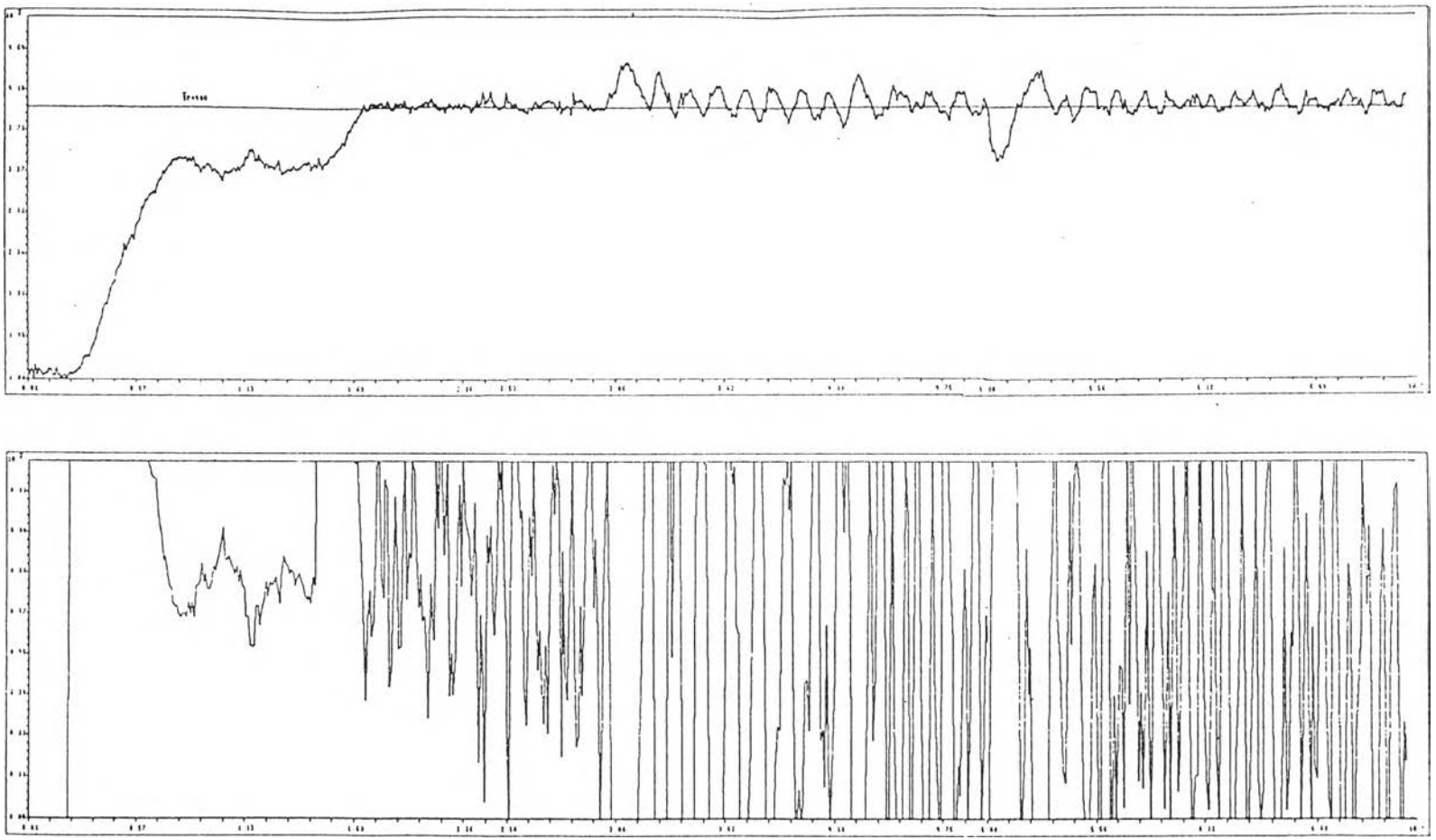
2. แกนตั้งของผลตอบของตัวควบคุมเป็นค่าเอาต์พุตของ D/A มีหน่วยเป็นระดับของเอาต์พุตของ D/A (มีค่าตั้งแต่ 0 - 100 หน่วย มีค่าประมาณ 4-20 mA)
แกนนอนเป็นแกนเวลามีหน่วยเป็นจำนวนครั้งของการสุ่ม ซึ่งคาบของการสุ่มเท่ากับ 5 วินาที



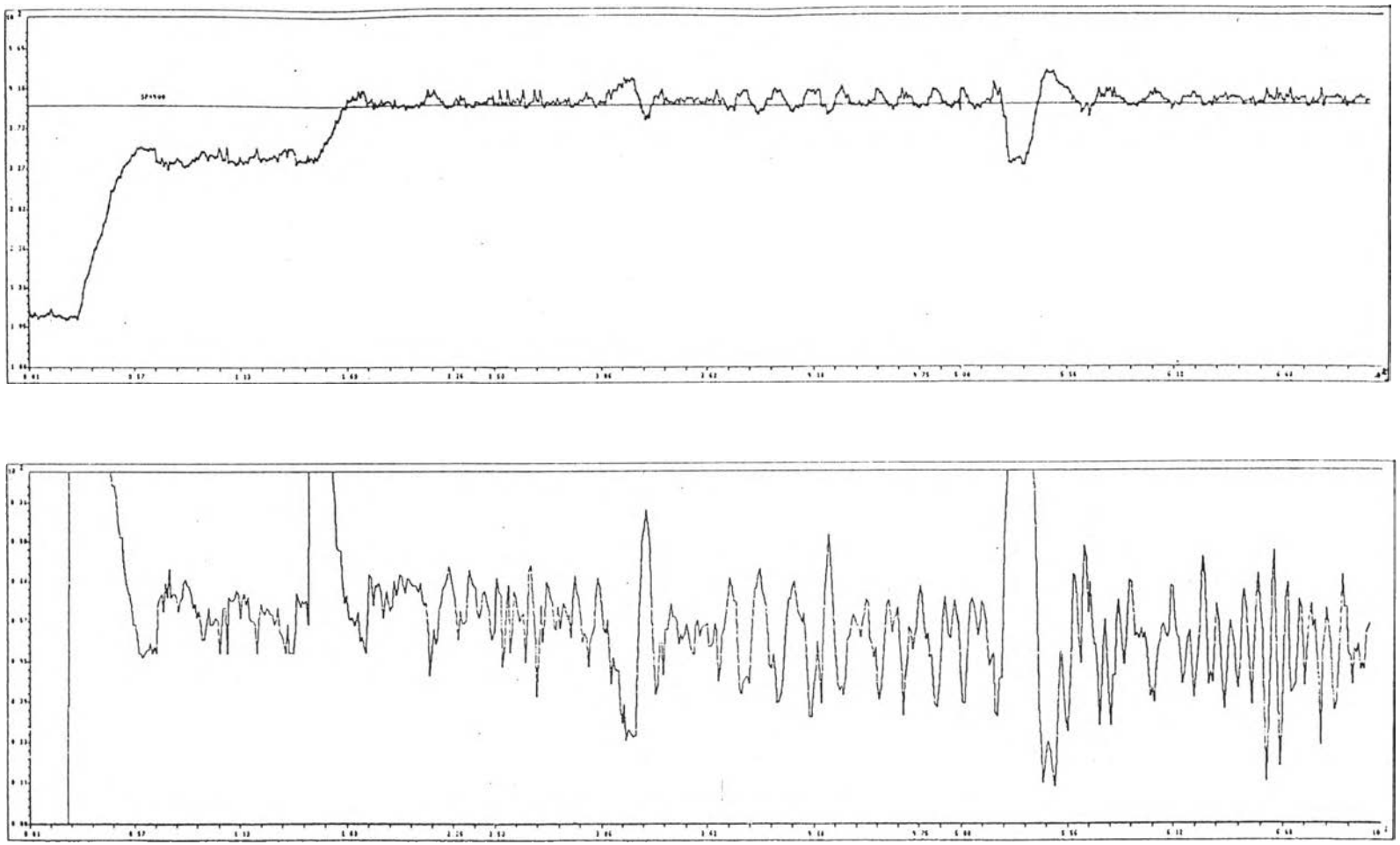
รูปที่ ค.1 แสดงผลตอบของกระบวนการ (บน) และผลตอบของตัวควบคุม (ล่าง)
โดยควบคุมด้วย Conventional PID ครั้งที่ 1



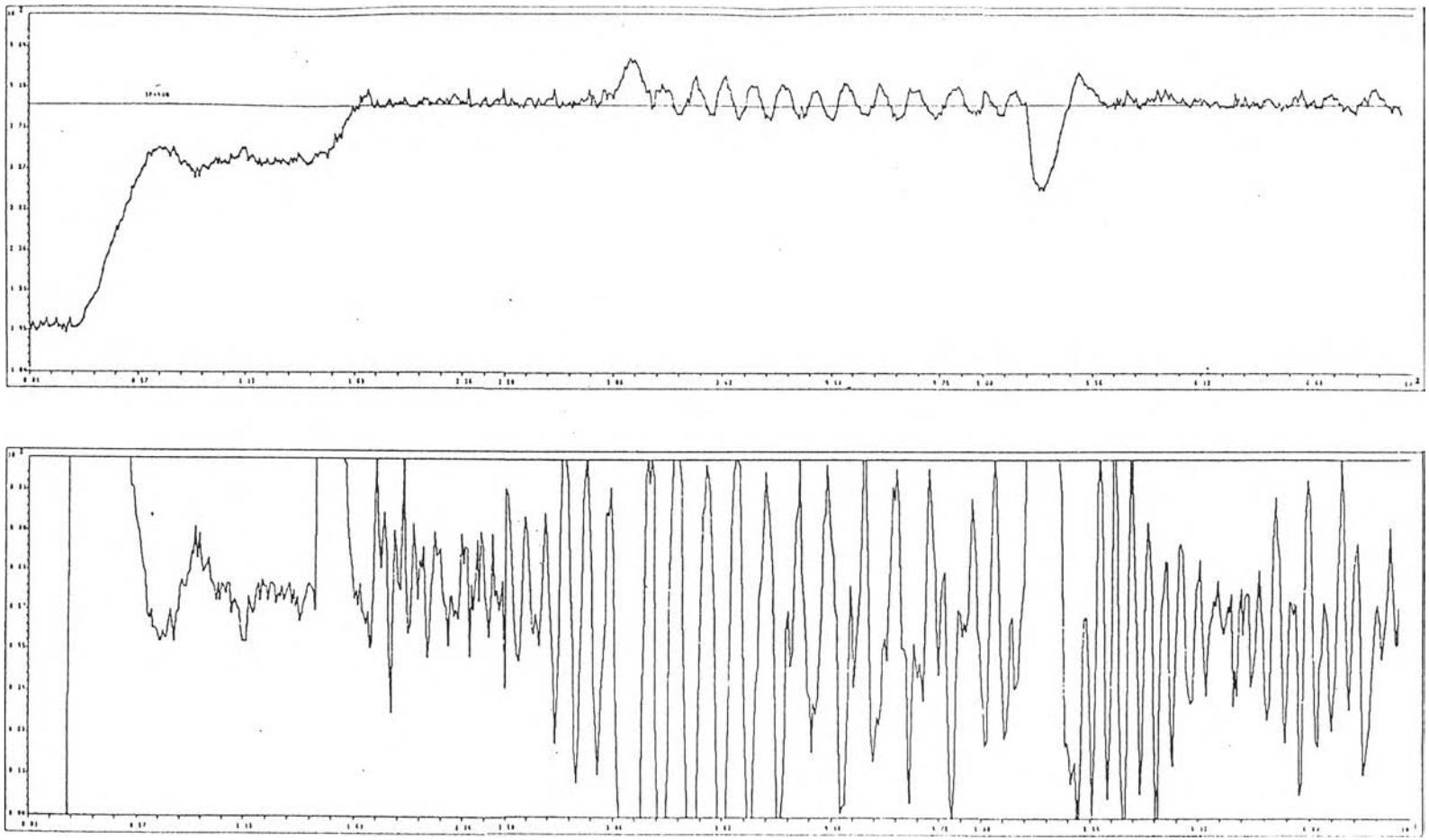
รูปที่ ค.2 แสดงผลตอบของกระบวนการ (บน) และผลตอบของตัวควบคุม (ล่าง)
โดยควบคุมด้วย Conventional PID ครั้งที่ 2



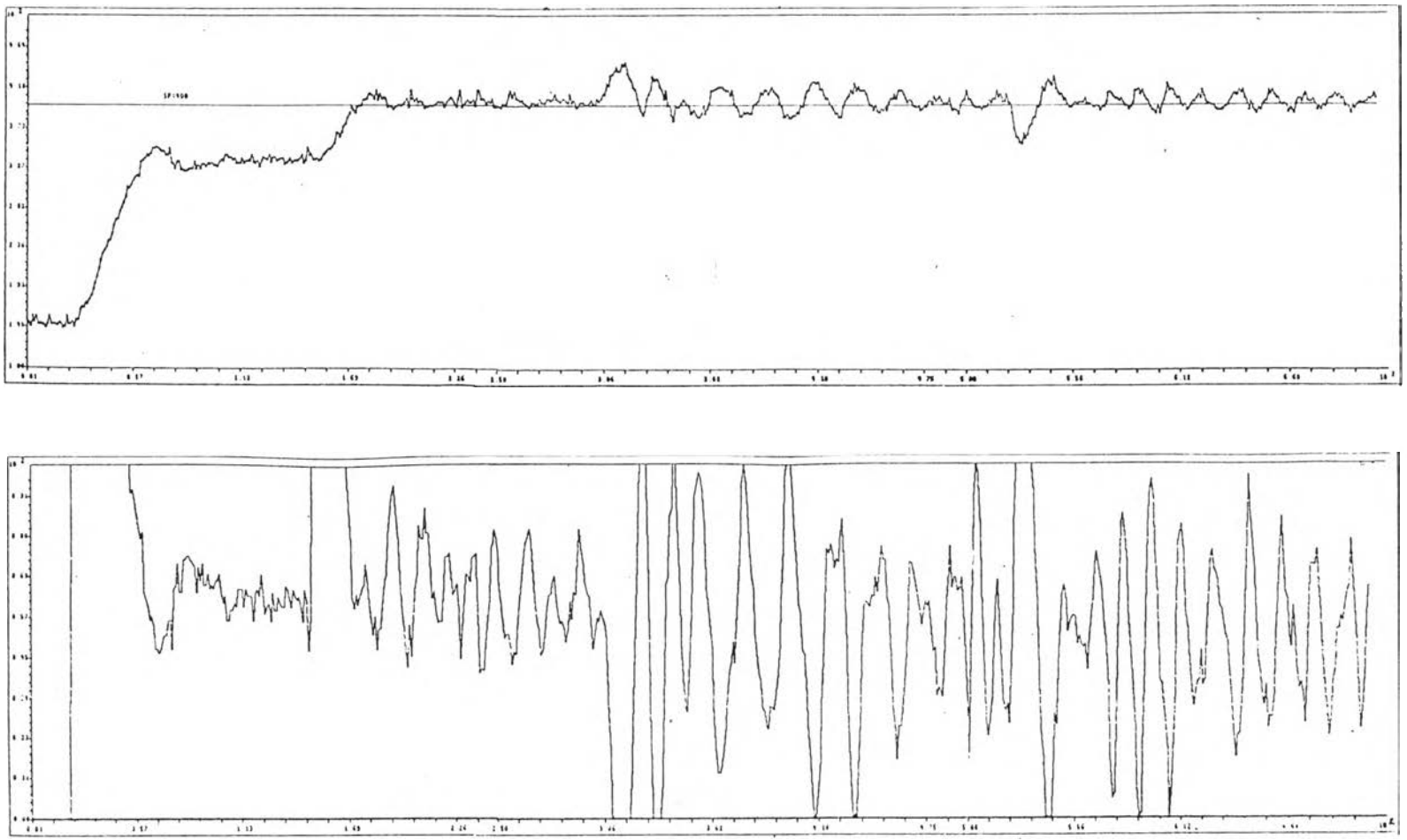
รูปที่ ค.3 แสดงผลตอบของกระบวนการ (บน) และผลตอบของตัวควบคุม (ล่าง)
โดยควบคุมด้วย MV ครั้งที่ 1 กำหนดให้ P(1) = 1 และ Q(1) = 0



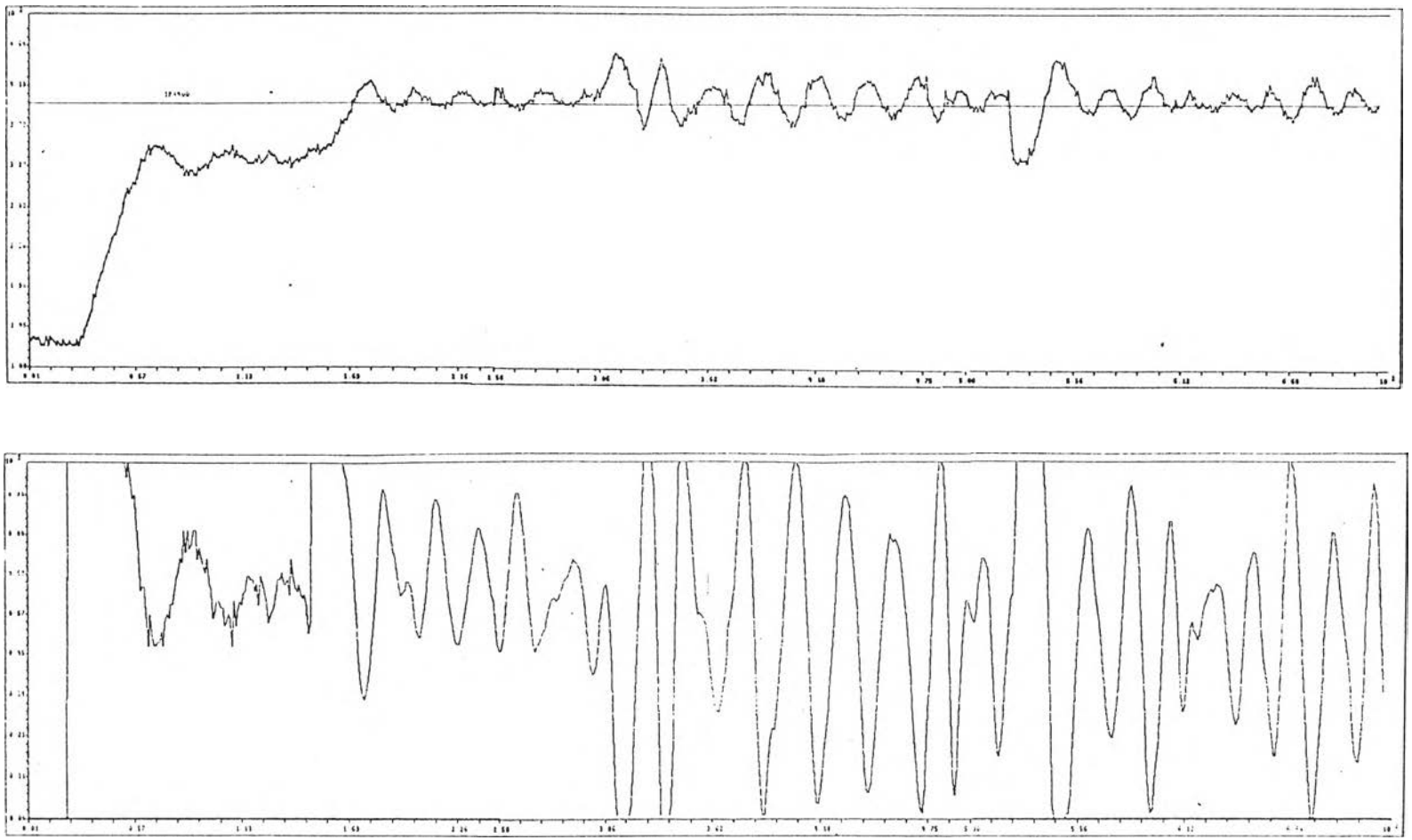
รูปที่ ค.4 แสดงผลตอบของกระบวนการ (บน) และผลตอบของตัวควบคุม (ล่าง)
โดยควบคุมด้วย GMV ครั้งที่ 1 กำหนดให้ $\lambda = 0.001$



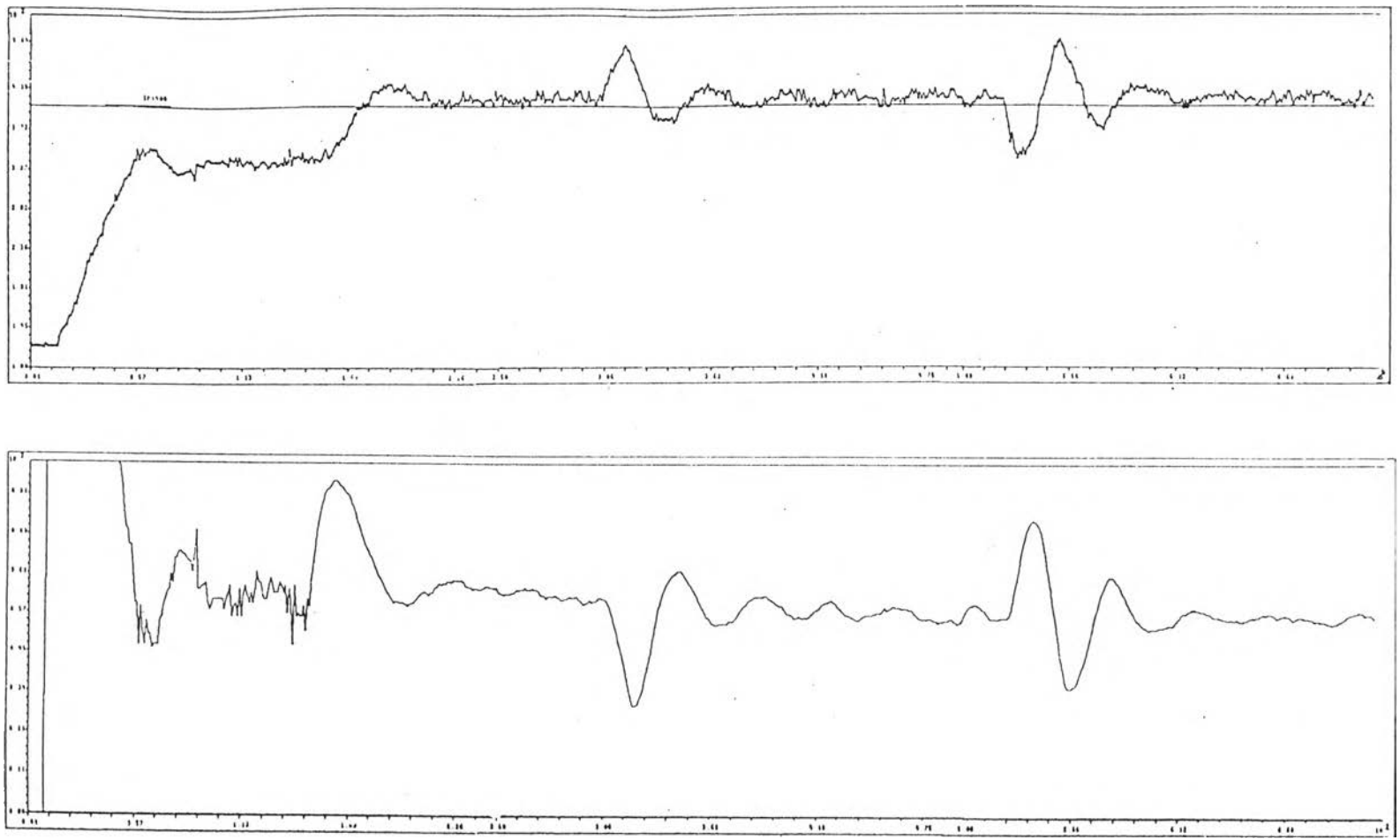
รูปที่ ค.5 แสดงผลตอบของกระบวนการ (บน) และผลตอบของตัวควบคุม (ล่าง)
โดยควบคุมด้วย GMV ครั้งที่ 2 กำหนดให้ $\lambda = 0.01$



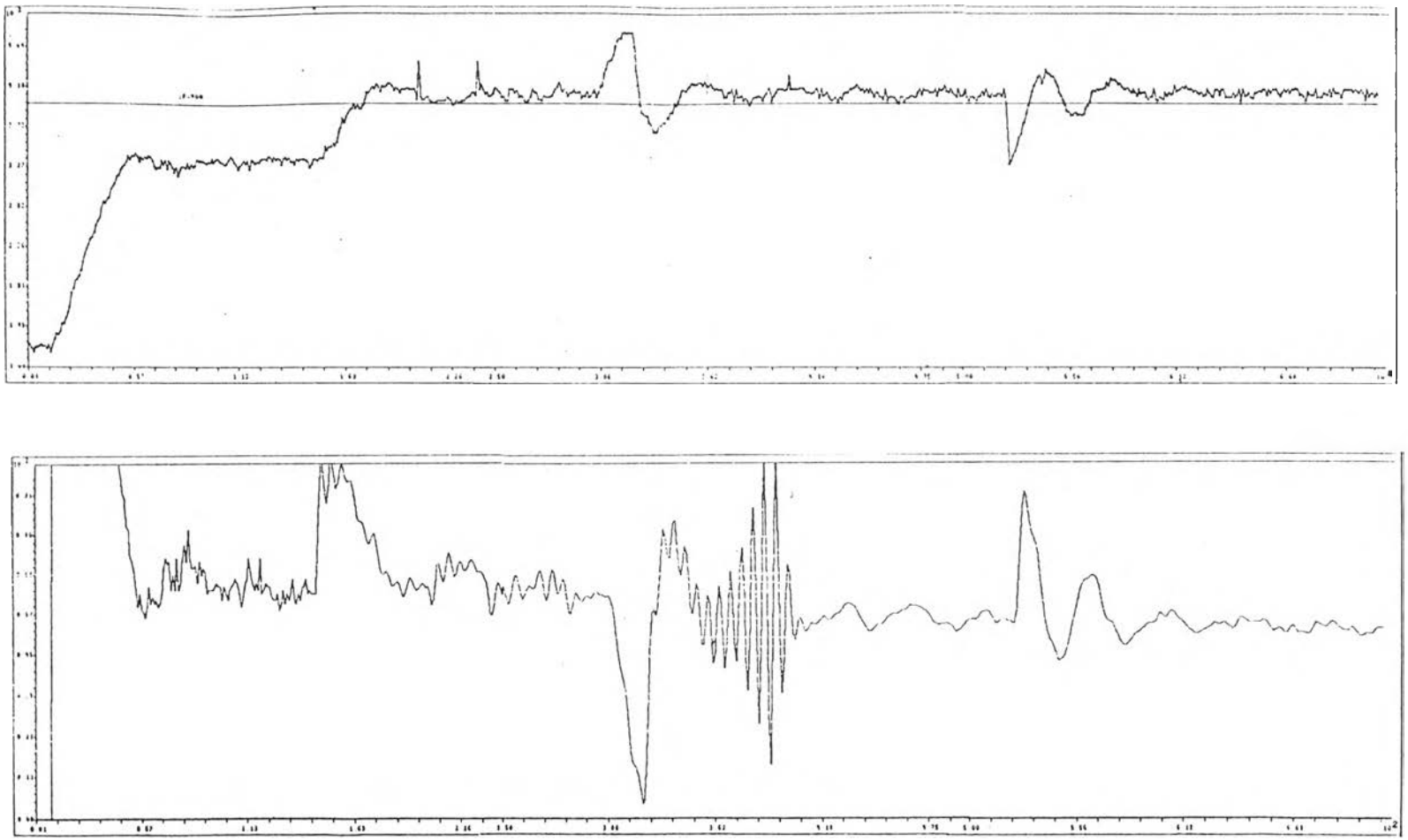
รูปที่ ค.6 แสดงผลตอบของกระบวนการ (บน) และผลตอบของตัวควบคุม (ล่าง)
โดยควบคุมด้วย GMV ครั้งที่ 3 กำหนดให้ $\lambda = 0.1$



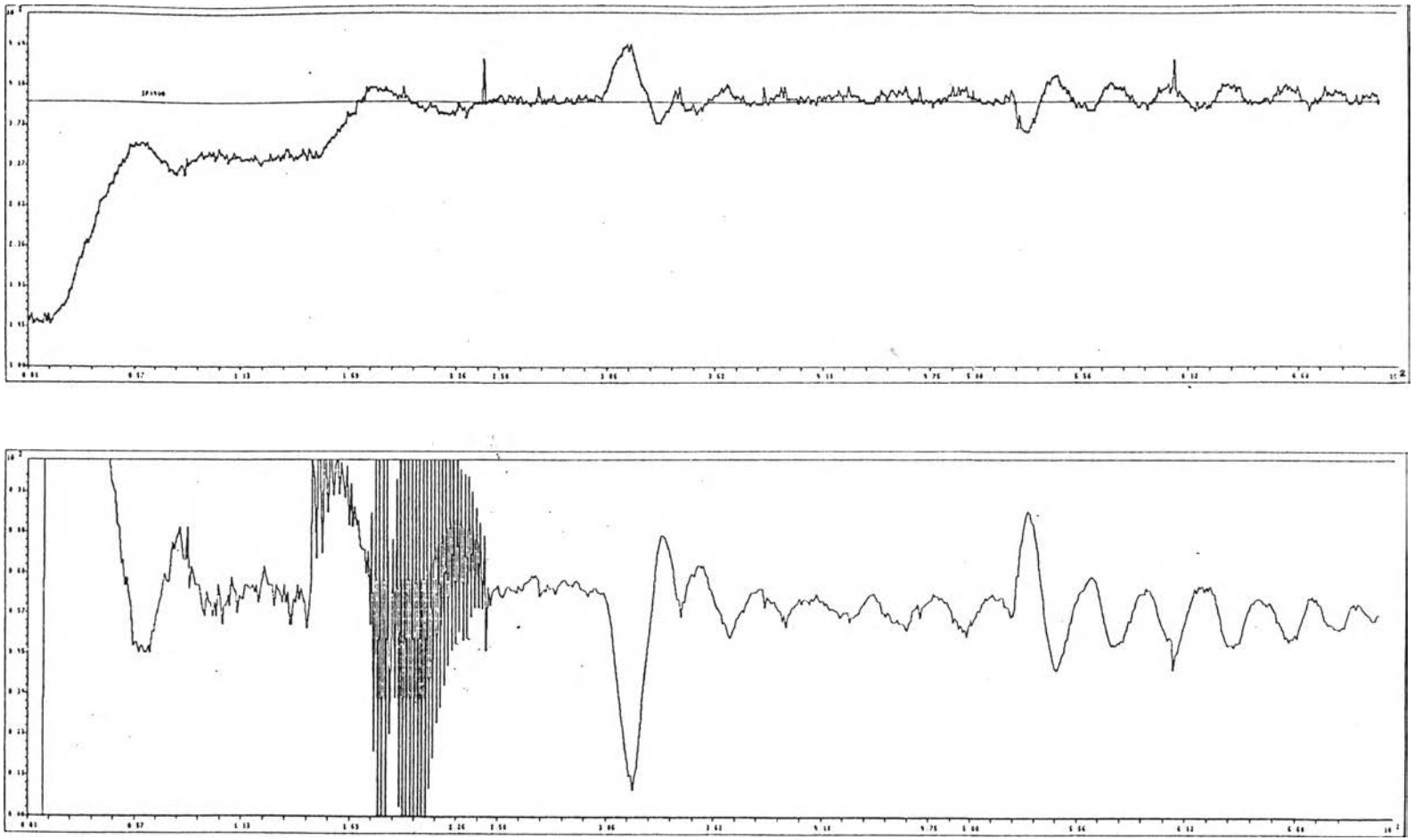
รูปที่ ค.7 แสดงผลตอบของกระบวนการ (บน) และผลตอบของตัวควบคุม (ล่าง)
โดยควบคุมด้วย GMV ครั้งที่ 4 กำหนดให้ $\lambda = 1$



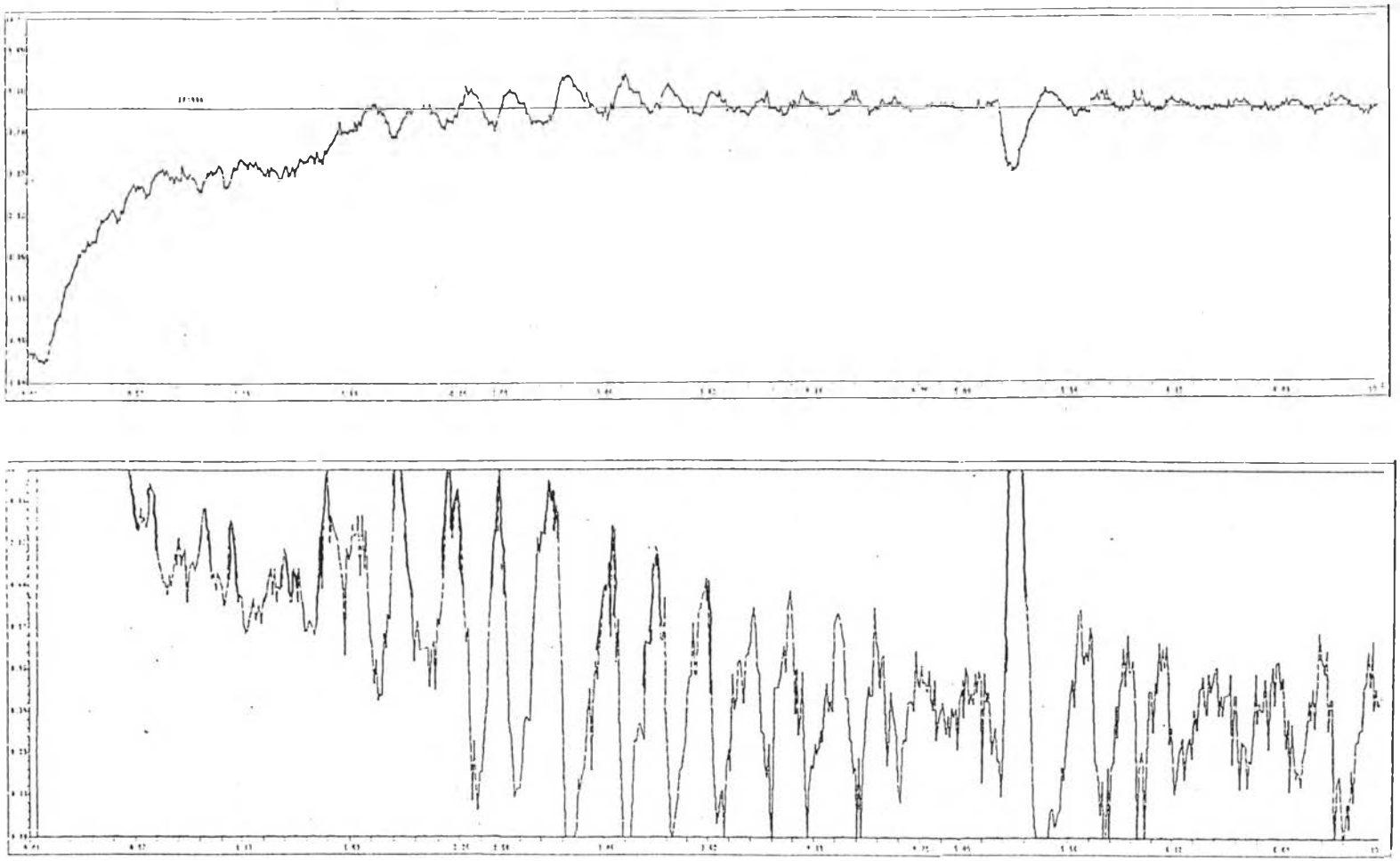
รูปที่ ค.8 แสดงผลตอบของกระบวนการ (บน) และผลตอบของตัวควบคุม (ล่าง)
โดยควบคุมด้วย PPL ครั้งที่ 1 กำหนดให้ ζ = 0.707 และ ω = 0.02 rad/s



รูปที่ ค.9 แสดงผลตอบของกระบวนการ (บน) และผลตอบของตัวควบคุม (ล่าง)
โดยควบคุมด้วย PPL ครั้งที่ 2 กำหนดให้ ζ = 0.707 และ w = 0.025 rad/s



รูปที่ ค.10 แสดงผลตอบของกระบวนการ (บน) และผลตอบของตัวควบคุม (ล่าง)
โดยควบคุมด้วย PPL ครั้งที่ 3 กำหนดให้ ζ = 0.707 และ w = 0.03 rad/s



รูปที่ ค.11 แสดงผลตอบของกระบวนการ (บน) และผลตอบของตัวควบคุม (ล่าง)
โดยควบคุมด้วย PI (on-line) ครั้งที่ 1

ประวัติผู้เขียน

นาย บุญเสริม แจ้งอรุณ เกิดวันที่ 22 กันยายน พ.ศ. 2505 ที่กรุงเทพมหานคร สำเร็จปริญญาวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมไฟฟ้า จากมหาวิทยาลัยเกษตรศาสตร์ เมื่อปี พ.ศ. 2526