เอกสารอ้างอิง

1. สมบูรณ์ จงชัยกิจ, เอกสารประกอบคำบรรยาย เรื่อง การควบคุมแบบอัตโนมัติ และการควบคุมแบบ PID, สมาคมเทคโนโลยี(ไทย-ญี่ปุ่น)

2. C. Zervos, P. R. Belanger, G. A. Dumont, On PID Controller Tuning Using Orthonormal Series Identification, Automatica Vol. 24. No. 2. 1988.

3. Ziegler, J. G. and N. B. Nichols. Optimum setting for PID controller. Trans ASME. 1942:759-768.

4. Astrom, K.J. and T. Hagglund. Automatic tuning of simple regulators for phase and amplitude margins specifications. IFAC Workshop on Adaptive Systems in Control and Signal Processing. SanFrancisco, 1983.

5. Cohen, G.H. and G.A. Coon, Theoretical investigation of retarded control. Trans. ASME. 1953:827

6. Pradeep B. despande, Raymond H. ash. Elements of Computer Process Control with Advanced Control Application, USA:Prentice Hall, 1981

7. Steven C. Chapra, Raymond P. Canale. Numerical Methods for Engineers with Personal Computer Application. 2. USA:McGraw-Hill, 1987

8. John J. D'Azzo, Constantine H. Houpis. Linear Control System Analysis and Design. 6. USA:McGraw-Hill, 1986.

9. Karl J. Astrom and Bjorn Wittenmark, Computer Controlled System Theory and Design. Prentice-Hall,Inc. 1984.

10. Stanley M. Shinners. Modern Control System Theory and Application. Mei Ya Publication, 1972.

11. Himsworth, F. R., Spendley, W. and Hext G. R. The Sequencial application of simplex designs in optimization and evolutionary operation. Techmometrics. Vol 4. 1962:441

12. Smith, C. S. The automatic Computation of maximum likelihood estimates. N.C.B. Scientific Dept., Report No. S.C. 846/MR/40

13. Rosenblock, H. H. An Automatic Method for finding the greatest or least value of a function. The Computer Journal, Vol 3., 1960

14. Spang, H. A. A Review of minimization techniques for nonlinear functions. S.I.A.M. Review, Vol 4. 1962:343.

15. Powell, M.J.D., An efficient method for finding the minimum of a function of several variables without calculating derivatives, Computer Journal, Vol. 7, 1964:155-162

16. Willard I. Zangwill, Minimizing a function without calculating derivatives, Computer Journal, 293-297.

17. Yokogawa Electric Corporation, Technical Information Yew Series 80.

18. James, H. M., N. B. Nichols and R. S. Phillips. Theory of Servomechanisms. McGraw-Hill, 1947.

19. Ohta T., N. Sanomiya, Y. Nishikawa, H. Tanaka and K. Tanaka. A new optimization of PID control parameters for automatic tuning by process computer., Computer Aided Design of Control Systems. (Proceedings of the IFAC Symposium, Zurich, 1979) 133-138.

20. Y. Nishikawa, N. Sanomiya, T. Ohta, H. Tanaka, A Method for Auto-tuning of PID Control Parameter, *Autometica*. Vol 20 No 3, 1984.

21. อำนวย แสงวิโรจน์พัฒน์, *ตัวควบคุมเชิงเลขชนิดโปรแกรมได้สำหรับขบวนการทางอุตสาหกรรมแบบต่อเนื่อง*,วิทยานิพนธ์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย 2532.

22. Salem A. K. Al-Assadi, Lamya A. M. Al-Chalabi, Optimal gain for Proportional-Integral-Derivative Feedback, IEEE Control Systems Magazine, 1987.

23. Foxboro Company, *PID Algorithm with Self-tuning*.

24. Kraus, T.W., Myron T.J., Evaluation and performance of a Self-tuning PID Controller, Foxboro Company.

ภาคผนวก ก.

จากสมการที่ 3.5

$$c(t) = [\ 1 - [(T_1)EXP(-(t-L)/T_1)]/(T_1-T_2)$$

$$+ [(T_2)EXP(-(t-L)/T_2)]/(T_1-T_2)\ ]U(t-L) \text{ -----[3.5]}$$

หาอนุพันธ์ของสมการที่ 3.5 จะได้

$$d/dt\ C(t) = EXP(-(t-L)/T_1)/(T_1-T_2)$$

$$- EXP(-(t-L)/T_2)/(T_1-T_2) \text{ --------------[ก.1]}$$

หาอนุพันธ์ของสมการที่ ก.1 จะได้

$$d^2/dt^2\ C(t) = -\ EXP(-(t-L)/T_1)/(T_1(T_1-T_2))$$

$$+ EXP(-(t-L)/T_2)/(T_2(T_1-T_2)) \text{ --------[ก.2]}$$

ให้สมการที่ ก.2 = 0 และ เอา (T_1-T_2) คูณตลอด จะได้

$$0 = -\ EXP(-(t-L)/T_1)/T_1 + EXP(-(t-L)/T_2)/T_2$$

ย้ายข้างและจัดรูปใหม่จะได้

$$[EXP(-(t-L)/T_1)]/[EXP(-(t-L)/T_2) = T_1/T_2$$

$$EXP[(-(t-L)/T_1) + (-(t-L)/T_2)] = T_1/T_2 \text{ -----------[ก.3]}$$

Take ln สมการที่ ก.3 จะได้

$$(-(t-L)/T_1) + (-(t-L)/T_2)] = \ln (T_1/T_2)$$

$$(t-L)((T_1-T_2)/(T_1*T_2)) = \ln (T_1/T_2)$$

ให้ $a = T_1*T_2/(T_1-T_2)$ และ $n = T_1/T_2$ จะได้

$$T - L = (a)\ln(n)$$

เพราะฉะนั้น จะได้

$$t_m = t = L + (a)\ln (n)$$

โดยที่ t หรือ t_m เป็นเวลาที่เกิดความชันที่มากที่สุด (Maximum Slope)

ภาคผนวก ข.

วิธีแรกของ Powell (The First Powell Procedure)

สำหรับการหาค่าที่เหมาะสมที่สุดโดยวิธีแรกของ Powellนี้ จะเป็นการหาค่าที่เหมาะสมที่สุดของฟังก์ชัน n ตัวแปร โดยก่อนอื่นจะขอกำหนด

ฟังก์ชัน $Y = f(x_1,x_2,\ldots,x_n)$
เริ่มหาค่าที่เหมาะสมที่สุดจากจุดเริ่มต้น p_0
ต้องการหาค่า $x_1,x_2,\ldots,x_n$ ที่ทำให้ค่า Y มีค่าน้อยที่สุด

ก่อนอื่น ขอกำหนด $z_1,z_2,\ldots,z_n$ ให้เป็น Coordinate directions n ทิศทาง และให้ i เป็นตัวแปรแสดงทิศทางของการหาค่าที่เหมาะสมที่สุด

ขั้นตอนที่ 1 :-

เริ่มจากทิศทาง i ที่ i = 1 ไปเรื่อย ๆ จนถึงทิศทางที่ n จะหาค่าของ r_i ที่ทำให้ $f(p_{i-1}+r_iz_i)$ มีค่าน้อยที่สุด และกำหนดจุด $p_i = p_{i-1} + r_iz_i$

ขั้นตอนที่ 2 :-

เริ่มจากทิศทาง i ที่ i = 1 ไปเรื่อย ๆ จนถึงทิศทางที่ n-1 กำหนดทิศทางใหม่ โดยให้ $z_i = z_{i+1}$ และกำหนด z_n เป็นทิศทางที่มุ่งจากจุด p_0 มาจุด p_n หรือ (p_0-p_n)

ขั้นตอนที่ 3 :-

หาค่าของ r ที่ทำให้ $f(p_n + rz_n)$ มีค่าน้อยที่สุดและให้ $p_0 = p_n+rz_n$

หลังจากเสร็จขั้นตอนที่ 3 แล้ว ให้กลับไปเริ่มรอบใหม่ที่ขั้นตอนที่ 1 ใหม่ จนกว่าจะได้จุดที่ให้ค่าที่เหมาะสมที่สุดที่ต้องการ

ตามการหาค่าโดยวิธีที่ 1 ของ Powell นั้น สามารถพิสูจน์ได้ว่าไม่สามารถเข้าสู่จุดที่ให้ค่าที่เหมาะสมที่สุดจริงได้ในทุกฟังก์ชัน โดยจะยกตัวอย่างฟังก์ชัน 3 ตัวแปรดังนี้

$$f(x,y,z) = (x-y+z)^2+(-x+y+z)^2+(x+y-z)^2 \quad \text{------}[4.1]$$

โดยจะเริ่มหาค่าที่เหมาะสมที่สุดจากจุด (0.5,1,0.5)

จากสมการที่ 4.1 จะเห็นได้อย่างชัดเจนว่า ค่าที่เหมาะสมที่สุดสำหรับสมการนี้นั้น มีค่าอยู่ที่จุด (0,0,0) และค่าของ $f(x,y,z) = 0$ แต่จากการใช้วิธีที่ 1 ของ Powell พบว่า เมื่อเสร็จขั้นตอนที่ 1 แล้ว จะได้จุด p_3 ที่ (1/2,1/3,5/18) และเมื่อหาทิศทางใหม่ตามขั้นตอนที่ 2 จะได้ทิศทางใหม่คือ {(0,1,0),(0,0,1),(0,-2/3,-2/9)} ซึ่งจะเห็นได้ว่า ในรอบถัดไปของการหาค่าที่เหมาะสมที่สุดจะไม่มีการหาในทิศทางแกน x ซึ่งทำให้จุดที่ให้ค่าที่เหมาะสมที่สุดที่จะหาได้ไม่ใช่จุดที่ให้ค่าที่เหมาะสมที่สุดที่ $x = 0$ ซึ่งแสดงว่าวิธีนี้ใช้ไม่ได้กับ Strickly Convex Function

วิธีที่ 2 ของ Powell (Simplified Powell's Second Procedure)

ข้อแตกต่างระหว่างวิธีแรกและวิธีที่สองของ Powell ดูเหมือนว่าจะอยู่ที่หลักในการหาทิศทางที่จะหาจุดที่ให้ค่าที่เหมาะสมที่สุดใหม่ และหลักในการเลือกใช้ทิศทางที่เปลี่ยนใหม่นั้น ซึ่งวิธีการหาตามวิธีที่ 2 ของ Powell สามารถอธิบายได้ดังนี้

กำหนด $z_1^1,z_2^1,\ldots,z_n^1$ เป็น Coordinate Direction และมีขนาดเท่ากับ Normallize Unit ดังนั้น $||z_i^1|| = 1$ โดยที่ i เป็นตัวแปรแสดงทิศทางการหาค่าที่เหมาะสมที่สุด มีค่า $i = 1,2,\ldots,n$

กำหนด Scalar e โดยที่ e มีค่าอยู่ระหว่าง 0-1

กำหนด p_0^1 เป็นจุดเริ่มต้นของการหาค่าที่เหมาะสมที่สุด

กำหนด $d^1 = 1$ และกำหนดรอบการทำงาน k โดยเริ่มจาก k = 1

การทำงานรอบที่ k

ขั้นตอนที่ 1

เริ่มจากทิศทาง i ที่ i=1 ไปเรื่อย ๆ จนถึงทิศทางที่ n จะหาค่าของ r_i^k ที่ทำให้ $f(p_{i-1}^k+r_i^kz_i^k)$ มีค่าน้อยที่สุด และกำหนดจุด $p_i^k = p_{i-1}^k + r_i^kz_i^k$

ขั้นตอนที่ 2 :-

กำหนด $a^k = ||p_n^k-p_0^k||$ และ $z_{n+1}^k = (p_n^k-p_0^k)/a^k$ แล้วคำนวนหาค่า r_{n+1}^k ที่ทำให้ $f(p_n^k+r_{n+1}^kz_{n+1}^k)$ มีค่าน้อยที่สุด และกำหนดจุด $p_0^{k+1} = p_{n+1}^k = p_n^k + r_{n+1}^kz_{n+1}^k$

ขั้นตอนที่ 3

หาค่า $r_s^k = \mathrm{Max}\{r_i^k \mid i=1,2,...,n\}$

กรณีที่ 1 ถ้า $r_s^kd^k/a^k >= e$ ให้ $z_i^{k+1} = z_i^k$ สำหรับ i ที่ไม่เท่ากับ s และ ให้ $z_s^{k+1} = z_{n+1}^{k+1}$ สำหรับ i ที่เท่ากับ s และให้ $d^{k+1} = r_s^kd^k/a^k$

กรณีที่ 2 ถ้า $r_s^kd^k/a^k < e$ ให้ $z_i^{k+1} = z_i^k$, i=1,2,...,n และให้ $d^{k+1} = d^k$

เริ่มทำรอบการทำงานที่ k โดยแทน k ด้วย k+1 จนกระทั่งถึงจุดที่ให้ค่าที่เหมาะสมที่สุด

Powell ได้พิสูจน์ให้เห็นว่าการหาค่าที่เหมาะสมที่สุดตามวิธีที่ 2 นั้น สามารถแก้ปัญหาที่เกิดจากการหาค่าตามวิธีที่ 1 ซึ่งทำให้วิธีที่ 2 นี้ สามารถหาค่าได้กับฟังก์ชันที่มีลักษณะของ Strickly Convex Function แต่อย่างไรก็ตาม วิธีที่ 2 ของ Powell ถึงแม้จะทำการ Simplify แล้ว ก็ยังเห็นได้ว่าเป็นวิธีที่ยุ่งยากในการหาค่าที่เหมาะสมที่สุดอยู่ Willard I. Zangwill จึงได้พัฒนาวิธีที่ 2 ของ Powell ขึ้นมาใหม่ ซึ่งสามารถพิสูจน์ได้ว่า วิธีที่พัฒนานี้

เป็นวิธีที่ลู่เข้าสู่จุดที่ให้ค่าที่เหมาะสมที่สุด ใน Finite number of Iteration และสามารถพิสูจน์ได้ว่า วิธีนี้จะลู่เข้าแม้กับฟังก์ชันในลักษณะของ Strickly Convex function และสามารถหาหาค่าที่เหมาะสมที่สุดได้ทุกคำตอบ

ภาคผนวก ค.

A/D, D/A Specification

1. INTRODUCTION TO A/D, D/A CONVERTER CARD

1.1. GENERAL DESCRIPTION

A/D, D/A Converter card, Deer Mountain Inc. part No. DM-P005B, is designed for IBM PC, XT, AT expansion slot, DM-P005B including:

1) 4 indepedent CH.(CH0-CH3) of 12-bit D/A converter.
2) Choose channel 0 through "Continuous approximate compare mode" offer 8 CH. A/D converter.
3) CH0-CH3 through "voltage to current converting CKT", offer 4-20mA standard industrial control signal output, its convenient for remote control signal transmission.

1.2. FEATURES

(1) D/A converter voltage O/P

* Resolution: 12 bits.
* Current setting time: 500nS to 0.05% of full scale.
* Full scale O/P voltage range: 0Vdc-10Vdc.
* Nonlinear distortion: 0.05% of full scale.
* O/P voltage precision: 0.3% of FSR.
* O/P voltage precision temperature coefficient: 20 PPM of FSR/°C.
* Power supply rejection: ±0.05% of FSR/%.

(2) D/A converter current O/P

* Resolution: 12 bits.
* Current setting time: 800nS to 0.05% of full scale.
* Full scale O/P current range: 4mA-20mA.
* Nonlinear distortion: 0.05% of full scale.
* O/P current temperature coefficent: 30 PPM of FSR/°C.
* Power supply rejection: 0.005% of FSR/%.
* Including O/P short protection device.
* Max loading resistor RL(max): 600 OHM ±5%.

(3) A/D converter voltage input
- * 8 channel, single end I/P mode.
- * Resolution: 12 bits.
- * Conversion time: 1mS/channel.
- * I/P voltage range: 0-5Vdc.
- * I/P impedance: 200K OHM ±5%.
- * Nonlinear distortion: 0.1% of full scale.
- * Precision: 0.5% of FSR ±1 LSB.
- * Power supply rejection: ±0.005% of FSR%.
- * Including I/P protection circuit.

1.3. SPECIFICATIONS

(1) Electrical Specifications
- * Capacity: 4 channels D/A or 3 channels D/A, 8 channels A/D.
- * Input signal: 0-5Vdc.
- * Output signal: 0-10Vdc and 4-20mA.
- * Can be used with multiple cards in parallel.
- * I/O address selection: 380-3FFH.
- * Interface: all address, data and control signals are TTL compatible.
- * Power requirements: ±5Vdc/200mA, ±12Vdc/200mA.

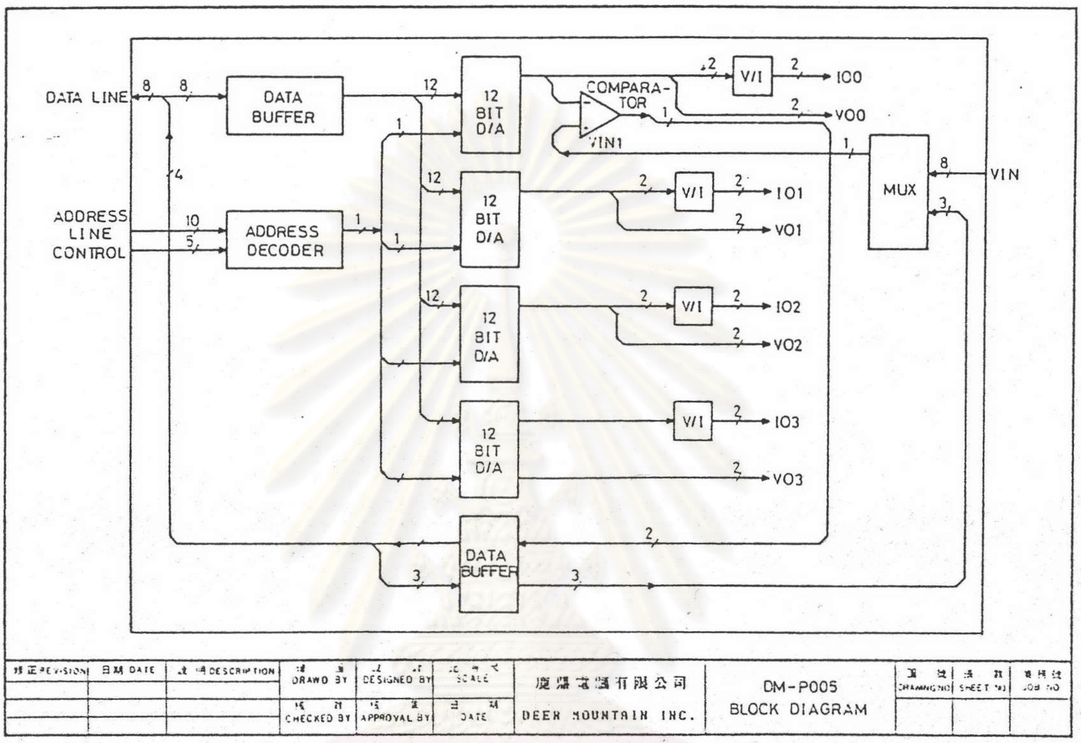
(2) Enviromental Specifications
- * Operating temperature: 0°C to 55°C.
- * Relative humidity: 0% to 90%.

(3) Mechanical Specifications
- * Meets general mechanical specifications of the IBM PC , XT, AT.
- * P.C.B. size:

  Length: 33.6cm
  Width : 10.4cm
  Height: 1.34cm
- * P.C.B. material: 1.6mm thickness FR-4.
- * P.C.B. gold finger thickness: 20u.
- * I/O connector: D-TYPE 37 pin, 1 set.
- * The locations of IC use circular hole socket, its gold inner contact/tin outer sleeve.

2. DESCRIPTION OF CIRCUIT

2.1. SCHEMATIC DIAGRAM



2.2. PRINCIPLES OF CKT

The DM-P005B has two functions as follows:

1) 4 channels D/A
   D/A output mode:
   * Voltage range is 0-10V.
   * Current range is 4-20mA.
2) A/D part: The user can select CH0 convert to 8 channels A/D by continous compare mode, A/D input is 0-5V.

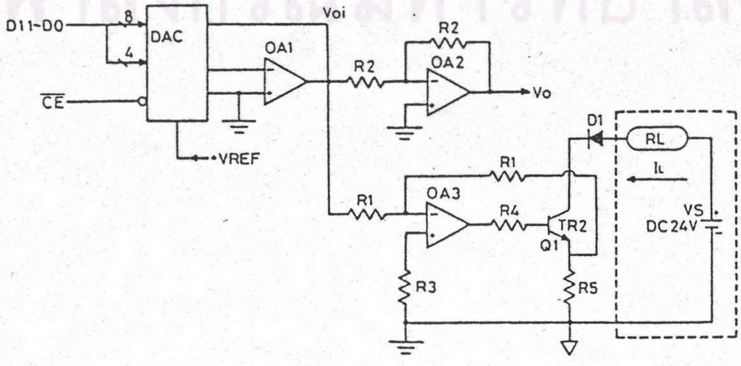


Fig 2-1

(1) 4-Channel D/A

DM-P005B use current multipling 12-bit DAC as D/A converter part.

The D/A converter includes three parts

1) D/A part: consist of OA1, DAC
OUTPUT VALTAGE Voi: 0-10V,
when input
D11-D0 is 0000H, Voi=-0.000V.
D11-D0 is 0FFFH, Voi=-9.996V.

2) Voltage output part: consist of OA2, R2.
Vo=-Voi by OA2, R2, output range is +0.000V-9.996V.

3) Current output part: consist of OA3, Q1, R5, D1, current output range: 4-20mA

Voi=1V	Il=4mA
Voi=5V	Il=20mA

the relation of output current Il and voltage output Voi :

Il=voi/Rl

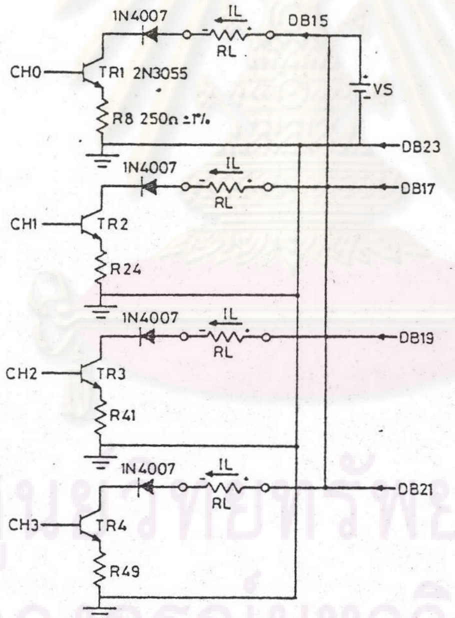


Fig 2-2

Vs min=24Vdc	Vs max=35Vdc
Rl min=0 OHM	Rl max=600 OHM

Remarks: Vs & Rl enable Q1 on, then occur current Il, user can use Il as remote control signal.

4) D/A channel port address arrangement:

	Low byte (D7-D0)	High byte (D11-D8)
CH0	380	381
CH1	382	383
CH2	384	385
CH3	386	387

D/A trasfer coefficient=Output Voltage(full scale)/2
=10V/4096
=2.241mV/bit

(2) A/D Part

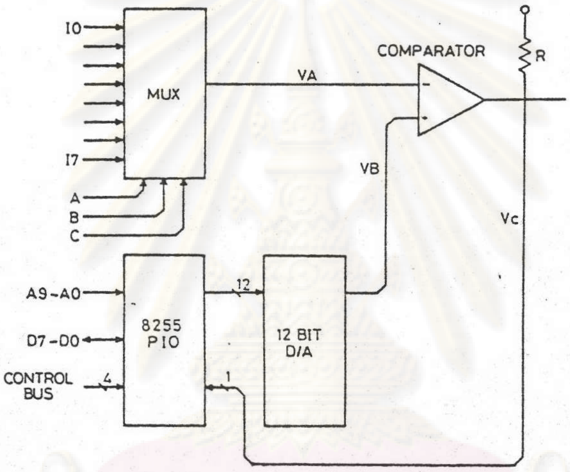


Fig 2-3

Pin A, B, C, of multiplex for channel selection. Vb of D/A output, it compare with Va, and the comparator output Vc is to 8255 of IBM PC, XT, AT main board. If Va>Vb, then Vc=0, software counter increases not stop until Vc="1", the contain of software counter is the binary of Va.
The CKT can be also named continuous compare converter.

2.3. ADDRESS DECODING CKT

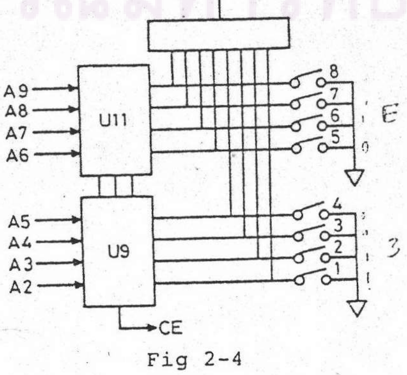


Fig 2-4

In Dip-Switch, pin OFF is logic "1", pin ON is logic "0". When address signal is equal with Dip-Switch setting $\overline{CE}$ sends a high pulse to D/A converter, pulse width is same as clock of IBM PC.

3. OPERATION DESCRIPTION

3.1. ADDRESS SELECTION

The board address can be set by Dip-Switch from 380-3FFH, each P.C.B. needs 10 address port, selecting mode as follows:

1) The relation of Dip-switch & address line as table 3-1

DIP-SW PIN NO.	8	7	6	5	4	3	2	1
ADDRESS LINE	A9	A8	A7	A6	A5	A4	A3	A2

Table 3-1

When Dip-Switch set "ON", represents the corresponding address line as ligic "0", otherwise as logic "1".

2) Example: DM-P005B address had been set 390-399H at our factory, then Dip-Switch setting as table 3-2

DIP-SW PIN NO.	8	7	6	5	4	3	2	1
DIP-SW STATUS	OFF	OFF	OFF	ON	ON	ON	OFF	OFF

Table 3-2

3.2. ANALOG I/O &DB-25 CONNECTOR ASSIGNMENT TABLE

DM-P005 use D-TYPE 25 pin connector as analog signal input & output, it define as table 3-3.

PIN NO	DESCRIPTION
1	A/D CH1 I/P Terminal
2	A/D CH2 I/P Terminal
3	A/D CH3 I/P Terminal
4	A/D CH4 I/P Terminal
5	A/D CH5 I/P Terminal
6	A/D CH6 I/P Terminal
7	A/D CH7 I/P Terminal
8	A/D CH8 I/P Terminal
9	D/A CH1 Voltage O/P Terminal
10	D/A CH2 Voltage O/P Terminal
11	D/A CH3 Voltage O/P Terminal
12	D/A CH4 Voltage O/P Terminal
13	NC
14	NC
15	Positive Terminal of CH1 Current O/P Channel
16	NC
17	Positive Terminal of CH2 Current O/P Channel
18	NC
19	Positive Terminal of CH3 Current O/P Channel
20	NC
21	Positive Terminal of CH4 Current O/P Channel
22	A/D, D/A Common Ground
23	A/D, D/A Common Ground
24	A/D, D/A Common Ground
25	A/D, D/A Common Ground

Notes:

1) DM-P005B choose CH. 0 through "continuous approximate compare mode" offer 8CH A/D, so D/A & A/D have mutual relation in voltage precision.
2) The precision of adjustment for D/A & A/D is VR2.

ภาคผนวก ง.

[ง.1] การประมาณค่าเวลาประวิง (Delay Time)

การประมาณค่าโดยวิธีการของ Pade

การประมาณค่าโดยวิธีการของ Pade นี้เป็นวิธีการประมาณค่า Differentiable Power Series ฟังก์ชันในรูปของ

$$d(x) = c_0 + c_1x + c_2x^2 + c_3x^3 + \ldots..$$

ให้อยู่ในรูปของฟังก์ชันตรรกยะ (Rational fraction)

$$F(x) = A_m(x)/B_m(x)$$

โดยที่

$$A_m(x) = a_0 + a_1x + a_2x^2 + \ldots + a_mx^m \text{ -----------[a]}$$

$$B_m(x) = b_0 + b_1x + b_2x^2 + \ldots + b_mx^m \text{ -----------[b]}$$

ค่า Coefficient ของ $A_m(x)$ และ $B_m(x)$ สามารถเขียนอยู่ในรูป matrix ได้ดังนี้

$$a = \begin{bmatrix} a_0 \\ a_1 \\ . \\ a_m \end{bmatrix} \qquad b = \begin{bmatrix} b_0 \\ b_1 \\ . \\ b_m \end{bmatrix}$$

กำหนด matrix $c_{p,q}$

$$c_{p,q} = \begin{bmatrix} c_q & c_{q-1} & \cdots & c_{q-p} \\ c_{q+1} & c_q & \cdots & c_{q-p+1} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ c_{p+q-1} & c_{q+p-2} & \cdots & c_{q-1} \end{bmatrix}$$

ค่า Coefficient ของสมการ [a] และ [b] หาได้จากสมการ matrix

$c_{m,m+1}b = 0$ และสมการ $c_{m+1,0}b = a$

การประมาณค่าอันดับหนึ่งของเวลาประวิง

กระจาย e^x ในรูปของ MacLaurin Series จะได้

$$e^x = 1 + x + x^2/2! + x^3/3! + \ldots$$

$m=1,\ c_0=1,\ c_1=1,\ c_2=1/2,\ c_3=1/6$

จะได้ $c_{1,2}b = \begin{vmatrix} 1/2 & 1 \end{vmatrix} \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = 0$ จะได้ $b_1 = -b_0/2$

$$c_{2,0}b = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \quad \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}$$

จะได้ $a_0 = b_0$ และ $b_0 + b_1 = a_1$

ให้ $a_0 = b_0 = 1$ เพราะฉะนั้นจะได้ $b_1 = 1/2$, $a_1= 1/2$

เพราะฉะนั้น จะได้

$$F(x) = (1 + x/2)/(1 - x/2) = d(x)$$

แทน x ด้วย $-Ls$ จะได้

$$EXP(-Ls) = (1- Ls/2)/(1 + Ls/2)$$

$$= (2 - Ls)/(2 + Ls)$$

การประมาณค่าอันดับสองของเวลาประวิง

$$F(x) = (a_0 + a_1x + a_2x^2)/(b_0 + b_1x + b_2x^2)$$

จะได้ $m=2$, $c_0=1$, $c_1=1$, $c_2=1/2$, $c_3=1/6$, $c_4=1/24$

$$c_{2,3} = \begin{bmatrix} c_3 & c_2 & c_1 \\ c_4 & c_3 & c_2 \end{bmatrix} \begin{bmatrix} 1/6 & 1/2 & 1 \\ 1/24 & 1/6 & 1/2 \end{bmatrix}$$

$$c_{3,0} = \begin{bmatrix} c_0 & 0 & 0 \\ c_1 & c_0 & 0 \\ c_2 & c_1 & c_0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1/2 & 1 & 1 \end{bmatrix}$$

เพราะว่า $c_{2,3}b = 0$

จะได้ $b_0/6 + b_1/2 + b_2 = 0$

$b_0/24 + b_1/6 + b_2/2 = 0$

เพราะว่า $c_{3,0}b = a$

จะได้ $a_0 = b_0$

$b_0 + b_1 = a_1$

$b_0/2 + b_1 + b_2 = a_2$

ให้ $a_0 = b_0 = 1$ จะได้ $a_1 = 1/2$, $a_2 = b_2 = 1/12$, $b_1 = -1/2$

เพราะฉะนั้นจะได้ $F(x) = (12 + 6x + x^2)/(12 - 6x + x^2)$

แทนค่า x ด้วย $-Ls$

$$e^{-Ls} = (12 - 6Ls + L^2s^2)/(12 + 6Ls + L^2s^2)$$

[ง.2] การหา Recursive Form ของเวลาประวิง

จาก $e^{-Ls} = (12 - 6Ls + L^2s^2)/(12 + 6Ls + L^2s^2)$

$$= 1 - 12Ls/(L^2s^2 + 6Ls + 12)$$

$$= 1 - (12s/L)/[(s+3/L)^2 + 3/L^2]$$

เปลี่ยนให้อยู่ในรูป Z-Transform จะได้

$$e^{-Ls} = (z-1)/z\ Z[\ 1/s - (12s/L)/[(s+3/L)^2 + 3/L^2]\ ]$$

$$= \frac{(z-1)}{z}\,\frac{z}{z-1} - \frac{12}{L}\,\frac{L}{3}\,\frac{ze^{-3T/L}\,\mathrm{Sin}(\ 3\ T/L)}{z^2 - 2z\mathrm{Cos}(\ 3\ T/L)e^{-3T/L} + e^{-6T/L}}$$

$$= 1 - \frac{4\ 3\ (z-1)\mathrm{Sin}(\ 3\ T/L)e^{-3T/L}}{z^2 - 2z\mathrm{Cos}(\ 3\ T/L)e^{-3T/L} + e^{-6T/L}}$$

$$= \frac{z^2 - 2\mathrm{Cos}(\ 3\ T/L)ze^{-3T/L} + e^{-6T/L} - 4\ 3\ \mathrm{Sin}(\ 3\ T/L)ze^{-3T/L} + 4\ 3\ \mathrm{Sin}(\ 3\ T/L)e^{-3T/L}}{z^2 - 2z\mathrm{Cos}(\ 3\ T/L)e^{-3T/L} + e^{-6T/L}}$$

$$= \frac{1 - (\ 2\mathrm{Cos}(\ 3\ T/L) + 4\ 3\ \mathrm{Sin}(\ 3\ T/L))e^{-3T/L}z^{-1} + (\ 4/\ 3\ \mathrm{Sin}(\ 3\ T/L)e^{-3T/L} + e^{-6T/L})z^{-2}}{1 - 2\mathrm{Cos}(\ 3\ T/L)e^{-3T/L}z^{-1} + e^{-6T/L}z^{-2}}$$

ทำ z^{-1}-Transform แล้วจัดรูปใหม่จะได้

$$c(k) = a[0]u[k] + a[1]u[k-1] + a[2]u[k-2] + b[0]c[k-1] + b[1]c[k-2]$$

โดยที่

$$a[0] = 1$$

$$a[1] = -[2\mathrm{Cos}(\ 3\ T/L) + 4\ 3\ \mathrm{Sin}(\ 3\ T/L)]e^{-3T/L}$$

$$a[2] = 4/\ 3\ \mathrm{Sin}(\ 3\ T/L)e^{-3T/L} + e^{-6T/L}$$

$$b[0] = 2\ \mathrm{Cos}(\ 3\ T/L)e^{-3T/L}$$

$$b[1] = e^{-6T/L}$$

[ง.3] การหา Recursive Form ของระบบ

- พิจารณาระบบหน่วงเกิน

$$C(s) = K/[s(s+T_1)(s+T_2)]$$

เปลี่ยนให้อยู่ในรูป Z-Transform จะได้

$$C(z) = (z-1)/z\ Z[\ K/[s(s+T_1)(s+T_2)]\ ] \quad \text{-------------}[*]$$

ใช้ Partial Fraction จะได้

$$K/[s(s+T_1)(s+T_2)] = A/s + B/(s+T_1) + C/(s+T_2)$$

โดยที่

$$A = K/T_1T_2$$

$$B = K/T_1(T_1-T_2)$$

$$C = -K/T_2(T_1-T_2)$$

จากสมการ [*] จะได้

$$C(z) = A + B(z-1)/(z-e^{-T_1T}) - C(z-1)/(z-e^{-T_2T})$$

ให้ $D = e^{-T_1T}$, $E = e^{-T_2T}$

$$C(z) = A + B(z-1)/(z-D) - C(z-1)/(z-E)$$

$$= \frac{(A+B+C)z^2 + (CD+C-AD-AE-BE-B)z + (ADE+BE-CD)}{z^2 - (D+E)z + DE}$$

$$= \frac{(A+B+C) + (CD+C-AD-AE-BE-B)z^{-1} + (ADE+BE-CD)z^{-2}}{1 - (D+E)z^{-1} + DEz^{-2}}$$

หา z^{-1}-Transform แล้วจัดรูปใหม่จะได้

$$c(k) = a[0]u[k] + a[1]u[k-1] + a[2]u[k-2] + b[0]c[k-1]$$
$$+ b[1]c[k-2]$$

โดยที่

$a[0] = A + B - C$

$a[1] = CD + C - AD - AE - BE - B$

$a[2] = ADE + BE - CD$

$b[0] = D + E$

$b[1] = DE$

$A = K/T_1T_2$

$B = K/T_1(T_1-T_2)$

$C = K/T_2(T_1-T_2)$

$D = e^{-T_1T}$

$E = e^{-T_2T}$

- พิจารณาระบบหน่วงวิกฤต

$C(s) = K/[s(s+T_1)^2]$

เปลี่ยนให้อยู่ในรูป Z-Transform จะได้

$C(z) = (z-1)/z\ Z[\ K/[s(s+T_1)^2]\ ]$ ------------------[**]

ใช้ Partial Fraction จะได้

$K/[s(s+T_1)^2] = A/s + B/(s+T_1) + C/(s+T_1)^2$

โดยที่

$A = K/T_1{}^2$

$B = -K/T_1{}^2$

$C = -K/T_1$

จากสมการ [**] จะได้

$$C(z) = K/T_1{}^2 - K/T_1{}^2(z-1)/(z-e^{-T_1T})$$
$$- Te^{-T_1T}(z-1)/(z-e^{-T_1T})^2$$

$$= [\ -2Ke^{-T_1T}/T_1{}^2 + K/T_1{}^2(1+e^{-T_1T})-Te^{-T_1T}]z^{-1}$$

$$1 - 2e^{-T_1T}z^{-1} + e^{-2T_1T}z^{-2}$$

$$+ [Ke^{-2T_1T}/T_1{}^2 - Ke^{-T_1T}/T_1{}^2 + Te^{-T_1T}]z^{-2}$$

ทำ z^{-1}-Transform แล้วจัดรูปใหม่จะได้

$$c(k) = a[0]u[k] + a[1]u[k-1] + a[2]u[k-2] + b[0]c[k-1] + b[1]c[k-2]$$

โดยที่

$a[0] = 0$

$a[1] = -Ke^{-T_1T}/T_1{}^2 + K/T_1{}^2 - Te^{-T_1T}$

$a[2] = Ke^{-2T_1T}/T_1{}^2 - Ke^{-T_1T}/T_1{}^2 + Te^{-T_1T}$

$b[0] = 2e^{-T_1T}$

$b[1] = e^{-2T_1T}$

- พิจารณาระบบหน่วงขาด

$C(s) = K/[s(s^2+As+B)]$

เปลี่ยนให้อยู่ในรูป Z-Transform จะได้

$C(z) = (z-1)/z\ Z[\ K/[s(s^2+As+B)]\ ]$ -------------[***]

ใช้ Partial Fraction จะได้

$K/[s(s^2+As+B)] = p/s + (xs+y)/(s^2+As+B)$

โดยที่

$p = K/B$

$x = -K/B$

$y = -AK/B$

จากสมการ [***] จะได้

$C(z) = K/B\ [\ 1 - (z-1)(z-p)/((z^2-qz+r)\ ]$

โดยที่

$p = [\mathrm{Cos}(BT) + A\ \mathrm{Sin}(BT)/2B]\ e^{-AT/2}$

$q = 2\ \mathrm{Cos}(BT)e^{-AT/2}$

$r = e^{-AT/2}$

$$c(z) = K/B[[(p-q+1)z^{-1}+(r-p)z^{-2}]/[z^{2}-qz+r]]$$

ทำ z^{-1}-Transform แล้วจัดรูปใหม่จะได้

$$c(k) = a[0]u[k] + a[1]u[k-1] + a[2]u[k-2] + b[0]c[k-1] + b[1]c[k-2]$$

โดยที่

$a[0] = 0$

$a[1] = K/B(p-q+1)$

$a[2] = K(r-p)/B$

$b[0] = q$

$b[1] = -r$

ภาคผนวก จ.

Program Listing

```
#include    <stdio.h>
#include    <io.h>
#include    <dos.h>
#include    <math.h>
#include    <conio.h>
#include    <stdarg.h>
#include    <stdlib.h>
#include    <attrib.h>

int         clr();
int         writemem();
int         cattrib();
int         gotorc();
int         getkey();
int         steady();
int         testmag(int out,int data,int setvalue,int stopvalue,
            int maxtime);
double      pow(double x,double y);
double      sqrt(double x);
double      fabs(double x);
void        stepmag(int outchno,double *magnitude,int *error);
void        restoredata(int outchno,double magnitude,int *error);
void        slopetest(void);
void        identify(double magnitude);
void        clrscr(void);
double      acos(double x);
double      sqrt(double x);
double      chord(double l,int *opt);
double      fabs(double x);
int         steady();
void        stepmag(int outchno,double *magnitude,int *error,
            int action);
void        restoredata(int outchno,double magnitude,int *error);
void        slopetest(void);
void        identify(double magnitude,double *gn,double *delay,
            double *para1,double *para2,int *state);
double      func1(double r,double n);
double      func2(double r,double n);
double      log(double x);
double      acos(double x);
double      exp(double x);
```

```
extern  int     chanal=2;
extern  int     *point=0;
extern  int     otimes=0;
extern  int     otimem=0;
extern  int     ntimes=0;
extern  int     ntimem=0;
extern  int     num=16440;
extern  int     number=65534;
extern  int     interval=20;
extern  int     errorcode=0;
extern  int     opt=0;
extern  int     lenght=0;
extern  double  maxdelaytime=0,maxsteadytime=0,presentsv=0;
extern  double  step=0.2,mvc=0,pvc=0,anc=0,bnc=0,cnc=0;
extern  double  ts=0.216;
extern  double  a[3]={0,0,0};
extern  double  b[3]={0,0,0};
extern  double  c[3]={0,0,0};
extern  double  d[3]={0,0,0};
extern  double  hilim=100;
extern  double  lowlim=0;



main()

{

FILE            *in;
int             i,j,ii,p,row,col,outchno,inchno,index;
int             samdata[8220];
double          k,l,proc1,proc2;
char            msg[11];

/******************     LOGO     ******************/

hidecur();
logo();

/****************** INITIALIZE #1 ******************/

point = &samdata[0];

/******************    DRAW BOX    ******************/

drawbox();

/****************** READ DATA #1  ******************/

movup(5,13,15,64,8,Normal);
writemem(0,5,23,Highlight,"Please choose data come from :-  ");
writemem(0,8,16,Normal," [ 1 ]  From data file. ");
writemem(0,10,16,Normal," [ 2 ]  From Process Interface. [Identify
System]");
```

```
writemem(0,20,6,Reverse,"  Input Requirement Data  ");
row = 8;
col = 16;
cattrib(0,row,col,50,Reverse);
index = 1;
do {
    p = getkey();
    switch(p)
      {
        case HOME_KEY:
        case PGUP_KEY:  {
                index = 1;
                cattrib(0,row,col,50,Normal);
                row = 8;
                col = 16;
                cattrib(0,row,col,50,Reverse);
                break;
                }
        case END_KEY:
        case PGDN_KEY:  {
                index = 2;
                cattrib(0,row,col,50,Normal);
                col = 16;
                row = 10;
                cattrib(0,row,col,50,Reverse);
                break;
                }
        case DOWN_KEY:  {
                cattrib(0,row,col,50,Normal);
                index++;
                row = 10;
                if(index > 2)


                  {
                    index = 1;
                    row = 8;
                    col = 16;
                  }
                cattrib(0,row,col,50,Reverse);
                break;
                }
        case UP_KEY:    {
                cattrib(0,row,col,50,Normal);
                index--;
                row = 8;
                if(index < 1)
                  {
                    index = 2;
                    row = 10;
                    col = 16;
                  }
                cattrib(0,row,col,50,Reverse);
```

```
                  break;
                  }
          case '\r':
          case '\n':  {
                  row = 0;
                  break;
                  }
          default:
                  break;
        }
} while (row != 0);

/**************  READ DATA & IDEN FROM FILE  ***************/

if (index == 1)
  {
    movup(5,10,15,66,6,Normal);
    presentsv = 50.0;
    fiden(&k,&l,&proc1,&proc2,&opt);
  }

/**************  READ DATA & IDEN FROM PROCESS  *************/

  else if (index == 2)
    {
     movup(5,10,15,66,6,Normal);
     pciden(&k,&l,&proc1,&proc2,&opt);
    }

/********************* SHOW VALUE ***********************/

movup(4,15,3,75,12,Normal);
writemem(0,5,18,Highlight,"  Identify System Transfer Function is
:-  ");
movup(20,22,3,37,3,Normal);
writemem(0,21,4,Reverse,"  Response of Identify System  ");
writemem(0,8,20,Normal,"  G(s) = ");
writeva(0,7,29,Normal,"            %+.4lf*S",(-1.0*l));
writeva(0,8,30,Underline,"     %.4lf*e          ",k);
if(opt==2)
  {
    writemem(0,9,29,Normal,"    2   ");
    writeva(0,10,29,Normal,"    S %+.4lf*S %+.4lf",proc1,proc2);
  }



 else if (opt==0¦¦opt==1)
    writeva(0,10,29,Normal," (S %+.4lf)(S %+.4lf)",proc1,proc2);
writemem(0,14,21,Highlight,"  Optimum                        Quit ");
cattrib(0,14,21,11,Reverse);
row = 14;
col = 21;
ii = 11;
index = 1;
```

```
do {
    p = getkey();
    switch(p)
      {
        case LEFT_KEY:
        case RIGHT_KEY: {
                cattrib(0,row,col,ii,Highlight);
                if (index == 1)
                  {
                    col = 47;
                    ii = 10;
                    index = 2;
                  }
                  else {
                    col = 21;
                    ii = 11;
                    index = 1;
                    }
                cattrib(0,row,col,ii,Reverse);
                break;
                }
        case '\r':
        case '\n':  {
                row = 0;
                break;
                }
        default:  break;
       }
 } while (row != 0);


 /********************** OPTIMIZE  VALUE *************************/

 if (index == 1)
   {
   optimize(k,l,proc1,proc2,opt);
   }
 gotorc(1,0);
 showcur();
 writemem(0,0,0,Reverse,"  Bye-Bye  and have a nice DOS.  ");
 sound(1000);
 delay(200);
 sound(1600);
 delay(500);
 nosound();

 }
```

```
fiden(double *gn,double *delay,double *para1,double *para2,int *opt)

{

FILE      *in;
extern    double  maxsteadytime,maxdelaytime;



char      name[15];
double    time[3],inp[3],out[5000],slope[2];
double    ts,ti,area,maxslope,ai,ao,lamda,tm;
double    c,t1,t2,ceta,neta,k,gain,r,q,w;
double    e = 0.3678794412,eps = 0.000001;
long int  l,m,n;
int       i,j,state;

area = 0.0;
maxslope = 0.0;
i = 1;
do {
    state = 1;
    writemem(0,7,30,Highlight,"Enter File's Name :-  ");
    writemem(0,10,34,Normal,"              ");
    cattrib(0,10,34,12,Reverse);
    showcur();
    gotorc(10,34);
    scanf("%s",&name);
    writemem(0,20,45,Normal,"                             ");
    writemem(0,21,45,Normal,"                             ");
    if((in = fopen(name,"r"))==NULL)
      {
        writemem(0,20,45,Reverse,"  Can not open input file.  ");
        writemem(0,21,45,ReverseBlink,"         TRY AGAIN !     ");
        state = 0;
      }
   } while (state == 0);
state = 1;
hidecur();
l = fileno(in);
m = filelength(l);
for ( j=0;j<=1;j++)
  {
    fscanf(in,"%lf",&time[j]);
    fseek(in,1,SEEK_CUR);
    fscanf(in,"%lf",&inp[j]);
    fseek(in,1,SEEK_CUR);
    fscanf(in,"%lf",&out[j]);
    fseek(in,1,SEEK_CUR);
  }
ts = time[1]-time[0];                    /*find sampling interval*/
do
  {
    i = i + 1;
    fscanf(in,"%lf",&time[2]);
    fseek(in,1,SEEK_CUR);
    fscanf(in,"%lf",&inp[2]);
    fseek(in,1,SEEK_CUR);
    fscanf(in,"%lf",&out[i]);
    fseek(in,1,SEEK_CUR);
    n = ftell(in);
    if ( ( i >= 500 ) && ( fabs(out[i] - out[i-1]) < eps) )
      {
        state = 0;
        c = out[i];
      }
    if ( i == 4999 )
      {
        writemem(0,20,45,Reverse,"  Can't store all of data!  ");
        writemem(0,20,45,ReverseBlink,"      Program Terminated    ");
        return(1);
      }


  } while ( n <= m && state == 1 );
fcloseall();
maxsteadytime = ts*i;
gain = c/inp[2];
k = 1.0/c;
for (j=0;j<=i;j++)
    out[j] = k*out[j];
for ( j=1;j<=i;j++ )
  {
    ao = ( out[j-1] + out[j] )/2.0;
    area = area + ( 1.0 - ao )*ts;
    slope[1] = ( out[j] - out[j-1] )/ts;
    if ( slope[1] < slope[0] )
        {
          if ( maxslope < slope[0] )
            {
              maxslope = slope[0];
              ti = j;
            }
        }
      else
        {
          if ( maxslope < slope[1] )
            {
              maxslope = slope[1];
              ti = j;
            }
        }
    slope[0] = slope[1];
  }
tm = ( 2.0 - out[ti] - out[(ti-1)] + maxslope*(((ti-1)*ts)+(ti*ts)))
/(2*maxslope);
lamda = maxslope*(tm-area);
```

```
neta = chord(lamda,&i);
*opt = i;
if ( *opt == 0 )
  {
   t1 = t2 = 1.0/(maxslope*e);
   ceta = area - 2.0*t1;
   k = gain/(t1*t2);
   r = 1.0/t1;
   q = 1.0/t2;
  }
 else if (*opt == 1)
   {
    ai = 1.0/(1.0-neta);
    ao = pow(neta,ai);
    t2 = ao/maxslope;
    ai = neta/(1.0-neta);
    ao = pow(neta,ai);
    t1 = ao/maxslope;
    ceta = area - t2*(neta+1.0)/neta;
    k = gain/(t1*t2);
    r = 1.0/t1;
    q = 1.0/t2;
   }
  else
   {
    ai = acos(neta);
    ao = 1.0-pow(neta,2.0);
    ao = sqrt(ao);
    w = ai/(ao*(tm-area));
    ceta = area - 2*neta/w;



    k = gain*w*w;
    r = 2.0*neta*w;
    q = w*w;
   }
*gn = k;
*delay = maxdelaytime = ceta;
*para1 = r;
*para2 = q;

}

void    pciden(double *k,double *l,double *t1,double *t2,int *state)

{

extern  int     chanal;
extern  int     *point;
extern  int     number;
extern  int     opt;
extern  int     lenght;
extern  double  maxdelaytime,maxsteadytime,presentsv;
```

```
FILE                 *out;
int                  i,ii,p,row,col,outchno,inchno,index;
int                  error,action;
double               magnitude;
char                 msg[11];
double               yr;

/*************  READ DATA FROM PROCESS  *************/

writemem(0,5,32,Highlight,"Approximate Data");
writemem(0,7,16,Normal,"Maximum Delay time           :-
second");
writemem(0,8,16,Normal,"Maximum Steady state time   :-
second");
writemem(0,9,16,Normal,"Present Process Value       :-
%     ");
writemem(0,10,16,Normal,"Final Element Action        :-
[0-Direct,1-Reverse]");
writemem(0,12,34,Highlight,"    Ready    ");
writemem(0,20,6,Reverse,"  Input Requirement Data  ");
row = 7;
col = 46;
cattrib(0,row,col,11,Reverse);
index = 1;
gotorc(row,col);
do {
    if(index != 5)
        showcur();
      else
        hidecur();
    p = getkey();
    writemem(0,20,49,Normal,"                        ");
    writemem(0,21,49,Normal,"                        ");
    if ( (p >= 0x30 && p<= 0x39) ¦¦ (p==0x2E))
      {
        i = 0;
        do {
            if ( (p >= 0x30 && p<= 0x39) ¦¦ (p==0x2E))
              {

                  putch(p);
                  cattrib(0,row,col,11,Reverse);

                  msg[i] = p;
                  i++;
                  if(i>10)
                    {
                      i=0;
                      gotorc(row,col);
                    }
                }
              p = getkey();
              if(p == 0x08)
                {
```

```
            ii = col + i - 1;
            if(i!=0)
              {
                i--;
                gotorc(row,ii);
                putch(' ');
                cattrib(0,row,col,11,Reverse);
                gotorc(row,ii);
              }
          }
       } while( p != 0x0D);
    msg[i] = '\0';
    if (index == 1)
        maxdelaytime = atof(msg);
      else if(index == 2)
          maxsteadytime = atof(msg);
        else if(index == 3)
            presentsv = atof(msg);
          else if(index == 4)
             action = atoi(msg);
    for(ii=i;ii<=10;ii++)
        putch(' ');
  }
switch(p)
  {
    case HOME_KEY:
    case PGUP_KEY:  {
            index = 1;
            cattrib(0,row,col,11,Highlight);
            row = 7;
            col = 46;
            cattrib(0,row,col,11,Reverse);
                gotorc(row,col);
            break;
            }
    case END_KEY:
    case PGDN_KEY: {
            index = 5;
            cattrib(0,row,col,11,Highlight);
            col = 34;
            row = 12;
            cattrib(0,row,col,11,Reverse);
            gotorc(row,col);
            break;
            }
     case DOWN_KEY:  {
            cattrib(0,row,col,11,Highlight);
            index++;
            if(index > 5)
               {
                 index = 1;
                 row = 7;
```

```
                    col = 46;
                  }
                  else if (index == 5)
                    {
                    row = 12;
                    col = 34;
                    }
                    else
                    row++;
                cattrib(0,row,col,11,Reverse);
                     gotorc(row,col);
                break;
                }
        case UP_KEY:     {
                cattrib(0,row,col,11,Highlight);
                index--;
                if(index < 1)
                  {
                    index = 5;
                    row = 12;
                    col = 34;
                  }
                  else if (index == 4)
                    {
                    row = 10;
                    col = 46;
                    }
                    else
                      {
                    row--;
                      }
                 cattrib(0,row,col,11,Reverse);
                     gotorc(row,col);
                 break;
                 }
         case '\r':
         case '\n': {
                 if (index==5)
                     index = 100;
                   else
                     {
                        cattrib(0,row,col,11,Highlight);
                        index++;
                         if(index > 5)
                         {
                            index = 1;
                            row = 7;
                            col = 46;
                         }
                         else if (index == 5)
                           {
                             row = 12;
                             col = 34;
```

```
                                  }
                                  else
                                    {
                                        row++;
                                    }
                               cattrib(0,row,col,11,Reverse);
                               gotorc(row,col);
                            }
                        break;
                        }


                default:  {
                            writemem(0,20,49,Reverse,"   Keypress Error ! ");
                            writemem(0,21,49,ReverseBlink,"   TRY  AGAIN !");
                            break;
                        }
              }
        } while (index != 100);
        yr = maxsteadytime/86.4;
        lenght = ceil(yr);

        /*****************  READ DATA #2  *****************/

        writemem(0,5,29,Highlight,"Input/Output Port Data");
        writemem(0,7,16,Normal,"        Output Port (0-3) :-              ");
        writemem(0,8,16,Normal,"        Input Port  (0-7) :-              ");
        writemem(0,9,16,Normal,"                                          ");
        writemem(0,10,16,Normal,"
");
        writemem(0,12,34,Highlight,"           ");
        writemem(0,11,34,Highlight,"   Ready   ");
        row = 7;
        col = 45;
        cattrib(0,row,col,11,Reverse);
        index = 1;
        gotorc(row,col);
        do {
            if ( index !=3 )
                showcur();
              else
                hidecur();
            p = getkey();
            writemem(0,20,49,Normal,"                      ");
            writemem(0,21,49,Normal,"                      ");
            if ( (index == 1) && (p >= 0x30) && (p<= 0x33) )
              {
                do {
                    if ( p>=0x30 && p<=0x33  )
                      {
                        putch(p);
                        cattrib(0,row,col,1,Reverse);
                        gotorc(row,col);
                        msg[0] = p;
```

```
          }
        p = getkey();
        if(p == 0x08)
          {
            gotorc(row,col);
            putch(' ');
            cattrib(0,row,col,1,Reverse);
            gotorc(row,col);
            msg[0] = ' ';
          }
       } while( (p != 0x0D) );
    msg[1] = '\0';
    outchno = atoi(msg);
  }
if ( (index == 2) && (p >= 0x30) && (p<= 0x37) )
  {
    do {
        if ( p>=0x30 && p<=0x37 )
          {
            putch(p);
            cattrib(0,row,col,1,Reverse);
            gotorc(row,col);


            msg[0] = p;
          }
        p = getkey();
        if(p == 0x08)
          {
            gotorc(row,col);
            putch(' ');
            cattrib(0,row,col,1,Reverse);
            gotorc(row,col);
            msg[0] = ' ';
          }
       } while( (p != 0x0D) );
    msg[1] = '\0';
    inchno = atoi(msg);
  }
switch(p)
  {
    case HOME_KEY:
    case PGUP_KEY:  {
            index = 1;
            cattrib(0,row,col,11,Highlight);
            row = 7;
            col = 45;
            cattrib(0,row,col,11,Reverse);
                gotorc(row,col);
            break;
            }
    case END_KEY:
    case PGDN_KEY:  {
            index = 3;
        cattrib(0,row,col,11,Highlight);
        col = 34;
        row = 11;
        cattrib(0,row,col,11,Reverse);
            gotorc(row,col);
        break;
        }
case DOWN_KEY:  {
        cattrib(0,row,col,11,Highlight);
        index++;
        if(index > 3)
          {
            index = 1;
            row = 7;
            col = 45;
          }
          else if (index == 3)
            {
            row = 11;
            col = 34;
            }
            else
            row++;
        cattrib(0,row,col,11,Reverse);
            gotorc(row,col);
        break;
        }
case UP_KEY:    {
        cattrib(0,row,col,11,Highlight);
        index--;
        if(index < 1)
          {
            index = 3;


            row = 11;
            col = 34;
          }
          else if (index == 2)
            {
            row = 8;
            col = 45;
            }
            else
              {
            row--;
              }
        cattrib(0,row,col,11,Reverse);
            gotorc(row,col);
        break;
        }
case '\b':  {
        gotorc(row,col);
        putch(' ');
```

```
              gotorc(row,col);
              msg[0] = ' ';
              break;
              }
      case '\r':
      case '\n':  {
              if (index==3)
                  index = 100;
                else
                  {
                     cattrib(0,row,col,11,Highlight);
                     index++;
                     if(index > 3)
                      {
                          index = 1;
                          row = 7;
                          col = 45;
                      }
                      else if (index == 3)
                         {
                           row = 11;
                           col = 34;
                         }
                         else
                           {
                               row++;
                           }
                      cattrib(0,row,col,11,Reverse);
                      gotorc(row,col);
                   }
               break;
               }
       default:  {
                   writemem(0,20,49,Reverse,"   Keypress Error ! ");
                   writemem(0,21,49,ReverseBlink,"   TRY  AGAIN !");
                   break;
               }
      }
} while (index != 100);

/****************** OUT PRESENT SV *****************/

writemem(0,20,2,Reverse,"  Find Step Disturbance Magnitude.  ");
magnitude = 403.2 + presentsv*16.128;



p = floor(magnitude);
output(outchno,p);
magnitude = 0;

/******************  INITIALIZE#2  *****************/

chanal = inchno;
```

```
/****************** FIND STEP MAG  *****************/

error = 0;
number = -2;
opt = 0;
stepmag(outchno,&magnitude,&error,action);
if (error == 1)
  {
    writemem(0,20,46,Reverse,"    System GAIN too low    ");
    writemem(0,21,46,Reverse,"  Can't used this method!  ");
    writemem(0,22,46,ReverseBlink," 10 s.  Program Terminated ");
    delay(10000);
    clr();
    gotorc(0,0);
    showcur();
    exit(0);
  }

/******************  RESTORE DATA  *****************/

number = -2;
opt = 0;
if ( action == 1)
    magnitude = -1.0*magnitude;
writemem(0,20,2,Normal,"                                                  ");
writemem(0,20,10,Reverse,"     Restore Data     ");
restoredata(outchno,magnitude,&error);
if ( action == 1)
    magnitude = -1.0*magnitude;
magnitude = fabs(magnitude);

/***************** IDENTIFY SYSTEM *****************/

writemem(0,20,10,Reverse,"    Identify System    ");
identify(magnitude,k,l,t1,t2,state);
writemem(0,20,10,Normal,"                       ");

}

identify(double magnitude,double *gn,double *delay,double *para1,
         double *para2,int *state)

{

extern  double  ts;
extern  int     number;
extern  int     *point;
extern  int     lenght;
extern  double   presentsv;
double           k,l,t1,t2;
double           input,t;
double           area,maxslope,slopen,ai,ao,lamda,tm;
double           c,ceta,neta,gain,r,q,w;
double           e = 0.3678794412;
double           sumx,sumy,sumxx,sumxy,x,y;
```

```
int              i,j,n,sv,offset,opt;

/***********************     INITIALIZE     ***********************/

area = 0.0;
maxslope = 0.0;
slopen = 0.0;
n = number/2.0;
input = 1.0;
ceta = 403.2 + presentsv*16.128;
neta = magnitude*16.128;
w = *point;
sv = floor(ceta);
offset = ceta - w;
j = floor(neta);

/*********************  FIND GAIN & PARAMETER *********************/

c = (*(point+n)-sv+offset)/neta;
gain = c/input;
k = 1.0/c;
for (i=1;i<=n;i++)
  {
    r = (*(point + i - 1) - sv + offset)*k/neta;
    q = (*(point + i) - sv + offset)*k/neta;
    ao = ( r + q )/2.0;
    area = area + (1.0-ao)*ts;
   }

/*********************   FIND MAXIMUM SLOPE   *********************/

sumx = sumy = sumxy = sumxx = 0.0;
for(j=lenght*20;j<=n;j++)
  {
    t = ts*j;
     for(i=0;i<lenght*20;i=i+lenght)
      {
        x = t - (i*ts);
        y = ((*(point+j-i)-sv+offset)*k)/neta;
          sumx = sumx + x;
          sumy = sumy + y;
          sumxy = sumxy + x*y;
          sumxx = sumxx + x*x;
        }
    slopen = ((20*sumxy)-(sumx*sumy))/((20*sumxx)-(sumx*sumx));
    if( slopen > maxslope )
      {
       maxslope = slopen;
       ao = sumy/20.0 - slopen*sumx/20.0;
      }
  }
```

```
tm = (input - ao)/maxslope;
lamda = fabs(maxslope*(tm-area));
neta = chord(lamda,&opt);
if ( opt == 0 )
   {
     t1 = t2 = 1.0/(maxslope*e);
     ceta = area - 2.0*t1;
     k = gain/(t1*t2);
     r = 1.0/t1;
     q = 1.0/t2;
   }


  else if (opt == 1)
    {
       ai = 1.0/(1.0-neta);
       ao = pow(neta,ai);
       t2 = ao/maxslope;
       ai = neta/(1.0-neta);
       ao = pow(neta,ai);
       t1 = ao/maxslope;
       ceta = area - t2*(neta+1.0)/neta;
       k = gain/(t1*t2);
       r = 1.0/t1;
       q = 1.0/t2;
     }
    else
      {
       ai = acos(neta);
       ao = 1.0-pow(neta,2.0);
       ao = sqrt(ao);
       w = ai/(ao*(tm-area));
       ceta = area - 2*neta/w;
       k = gain*w*w;
       r = 2.0*neta*w;
       q = w*w;
      }
 *gn = k;
 *delay = ceta;
 *state = opt;
 *para1 = r;
 *para2 = q;
 }

 double       chord(double lamda,int *opt)

 {

 double       start,stop,x,fstart,fstop,fx;
 double       crit = 0.3678794412,eps  = 0.0005;
 int          i;

 i = 0;
 start = 0.0000000001;
 stop = 0.9999999999;
 if ( fabs(lamda - crit) < eps )
     {
       *opt = 0;
       x = 1.0;
       return(x);
     }
   else if ( lamda < crit )
          {
             *opt = 1;
             fstart = func1(lamda,start);
             fstop = func1(lamda,stop);
          }
        else
            {
              *opt = 2;
              fstart = func2(lamda,start);
              fstop = func2(lamda,stop);
            }
 if ( fabs(fstart) < eps )
    return(start);
  else if ( fabs(fstop) < eps )


          return(stop);
do
  {
    x = ( start*fstop - stop*fstart )/( fstop - fstart );
    if ( *opt == 1 )
        fx = func1(lamda,x);
      else if ( *opt == 2 )
        fx = func2(lamda,x);
    if ( fabs(fx) < eps )
        return(x);
    if ( fx*fstart < 0 )
        {
          stop = x;
          fstop = fx;
        }
      else
        {
          start = x;
          fstart = fx;
        }
  } while ( i == 0 );
}

double      func1(double r,double n)

{

double      a,b;
```

```
a = 1.0/(1.0-n);
b = pow(n,a);
a = b/(n-1);
b = log(n);
return((a*b)-r);

}

double       func2(double r,double n)

{

double       a,b,c,d;

a = acos(n);
c = 1.0-pow(n,2.0);
b = sqrt(c);
c = -1.0*a*n/b;
d = exp(c);
c = a*d/b;
return(c-r);

}

void         stepmag(int out,double *magnitude,int *error,int action)

{

extern  int     *point;
extern  int     opt;
extern  int     errorcode;
extern  int     number;
extern  double  maxdelaytime;
extern  double  maxsteadytime;



extern  double  presentsv;
extern  double  ts;

int             sv,data,time,cond;
double          r,mvalue,value,dbias,onepercent,fivepercent,spstep;

dbias = 403.2;
onepercent = 16.128;
spstep = 0;
fivepercent = onepercent*5;
value = dbias + presentsv*onepercent;
sv = floor(value);
r = (maxdelaytime + maxsteadytime)/ts;
time = 2*floor(r);
opt = 0;
while ( opt == 0 && spstep <= 20*onepercent )
  {
      spstep = spstep + fivepercent;
      if ( action == 1 )
             value = value - fivepercent;
       else if (action==0)
             value = value + fivepercent;
      mvalue = dbias + presentsv*onepercent + 0.6*spstep;
     data = floor(value);
     cond = floor(mvalue);
     number = -2;
     steady();
     opt = 0;
     number = -2;
     testmag(out,data,sv,cond,time);
     if (opt == 1)
         *magnitude = spstep/onepercent;
  }
if (opt == 0 && spstep> 20*onepercent)
    *error = 1;
}

void        restoredata(int out,double magnitude,int error)

{

extern  int      *point;
extern  int      opt;
extern  int      errorcode;
extern  int      number;
extern  double   maxdelaytime;
extern  double   maxsteadytime;
extern  double   presentsv;
extern  double   ts;

int              sv,data,time,cond;
double           r,mvalue,value,dbias,onepercent,fivepercent,spstep;

ts = 0.216;
dbias = 403.2;
onepercent = 16.128;
error = 0;
steady();
value = dbias + presentsv*onepercent;
sv = floor(value);
spstep = value + magnitude*onepercent;
data = floor(spstep);
opt = 0;



number = -2;
keepdata(out,data,sv,error);

}
```

```
void        slopetes()

{

extern  int     *point;
extern  int     number;
extern  int     opt,lenght;
extern  int     otimes,otimem;
extern  double  maxdelaytime,maxsteadytime;
double          slope,sumx,sumy,sumxy,sumxx;
double          intv,t,x,y,eps;
int             pt,i,j;

eps = 0.00001;
intv =0.216;
pt = number/2;
t = pt*intv;
if( t > maxdelaytime && pt > lenght*20 )
  {
    sumx = sumy = sumxy = sumxx = 0.0;
    for(i=0;i<lenght*20;i=i+lenght)
      {
        x = t - (i*intv);
        y = ((*(point+pt-i)*0.0024801573)-1.0)*25.0;
        sumx = sumx + x;
        sumy = sumy + y;
        sumxy = sumxy + x*y;
        sumxx = sumxx + x*x;
      }
    slope = ((20*sumxy)-(sumx*sumy))/((20*sumxx)-(sumx*sumx));
    if(slope < 0)
        slope = slope*(-1);
    if( (slope < eps) || (t >= maxsteadytime) )
        opt = 1;
  }
}

optimize(double gain,double tdelay,double para1,double para2,
         int status)

{

extern  double  ts,mvc,pvc;
extern  double  a[],b[],c[],d[],hilim,lowlim;

double  pid[3],o,l,q[14];
double  dx,eps,result;
int     i,ii,p,row,col;
int     pi,index,max;
char        mass[10];

drawbox();
```

```
/*-------------- Initialize Data ---------------------*/

pid[1] = 100;            /* Proportional Band [PB] */
pid[0] = 10000;          /*   Integral Time [Ti]   */
pid[2] = 0;              /*  Derivative Time [Td]  */



hilim = 100;
lowlim = 0;

/*------------ Read Controller Data ------------------*/

writemem(0,1,37,Highlight,"INPUT");
movup(5,13,15,75,9,Normal);
movup(20,22,3,37,3,Normal);
writemem(0,5,17,Reverse,"    Input start point of Controller Parameter
   ");
writemem(0,8,20,Normal,"Proportional Band (PB) := ");
writemem(0,9,20,Normal,"Integral time (Ti)     := ");
writemem(0,10,20,Normal,"Derivetive time (Td)   := ");
writemem(0,11,20,Normal,"High Limit (%)         := ");
writemem(0,12,20,Normal,"Low Limit (%)          := ");
writemem(0,14,37,Highlight,"Ready");
writemem(0,21,6,Reverse,"  Input Start Parameter  ");
cattrib(0,8,47,11,Reverse);
row = 8;
col = 47;
index = 1;
gotorc(row,col);
do {
    if(index != 6)
        showcur();
      else
        hidecur();
    p = getKey();
    writemem(0,20,49,Normal,"                       ");
    writemem(0,21,49,Normal,"                       ");
    if ( (p >= 0x30 && p<= 0x39) || (p==0x2E))
      {
        i = 0;
        do {
            if ( (p >= 0x30 && p<= 0x39) || (p==0x2e))
              {
                putch(p);
                cattrib(0,row,col,11,Reverse);
                mass[i] = p;
                i++;
                if(i>10)
                  {
                    i=0;
                    gotorc(row,col);
                  }
              }
```

```
       p = getkey();
       if (p == 0x08)
         {
           ii = col + i - 1;
           if (i!=0)
             {
               i--;
               gotorc(row,ii);
               putch(' ');
               cattrib(0,row,col,11,Reverse);
               gotorc(row,ii);
             }
         }
      } while( p != 0x0D);
   mass[i] = '\0';
   switch (index)
     {


       case 1 :
           {
           pid[1] = atof(mass);
           break;
           }
       case 2 :
           {
           pid[0] = atof(mass);
           break;
           }
       case 3 :
           {
           pid[2] = atof(mass);
           break;
           }
       case 4 :
           {
           hilim = atof(mass);
           hilim = hilim/100.0;
           break;
           }
       case 5 :
           {
           lowlim = atof(mass);
           lowlim = lowlim/100.0;
           break;
           }
      }
  }
switch(p)
  {
    case HOME_KEY:
    case PGUP_KEY:  {
            index = 1;
```

```
        cattrib(0,row,col,11,Highlight);
        row = 8;
        col = 47;
        cattrib(0,row,col,11,Reverse);
            gotorc(row,col);
        break;
        }
case END_KEY:
case PGDN_KEY:  {
        index = 6;
        cattrib(0,row,col,11,Highlight);
        col = 34;
        row = 14;
        cattrib(0,row,col,11,Reverse);
            gotorc(row,col);
        break;
        }
case DOWN_KEY:  {
        cattrib(0,row,col,11,Highlight);
        index++;
        if(index > 6)
          {
            index = 1;
            row = 8;
            col = 47;
          }
          else if (index == 6)
            {
            row = 14;


            col = 34;
            }
            else
            row++;
        cattrib(0,row,col,11,Reverse);
            gotorc(row,col);
        break;
        }
case UP_KEY:    {
        cattrib(0,row,col,11,Highlight);
        index--;
        if(index < 1)
          {
            index = 6;
            row = 14;
            col = 34;
          }
          else if (index == 5)
            {
                row = 12;
                col = 47;
            }
            else
```

```
                    {
                  row--;
                    }
              cattrib(0,row,col,11,Reverse);
                  gotorc(row,col);
              break;
              }
      case '\r':
      case '\n':  {
              if (index==6)
                  index = 100;
                else
                  {
                     cattrib(0,row,col,11,Highlight);
                     index++;
                     if(index > 6)
                      {
                          index = 1;
                          row = 8;
                          col = 47;
                      }
                      else if (index == 6)
                        {
                           row = 14;
                           col = 34;
                        }
                        else
                          {
                               row++;
                          }
                      cattrib(0,row,col,11,Reverse);
                      gotorc(row,col);
                   }
               break;
               }
       default:  {
                   writemem(0,20,49,Reverse,"  Keypress Error ! ");
                   writemem(0,21,49,ReverseBlink,"    TRY  AGAIN !");
                   break;
               }


      }
} while (index != 100);

/*---------------- Choose Performance Index ------------------*/

movup(5,14,15,75,10,Normal);
movup(20,22,3,37,3,Normal);
writemem(0,5,20,Reverse,"   Choose Performace Index for Optimum    ");
writemem(0,8,26,Highlight," 1.  For No Overshoot ");
writemem(0,9,26,Highlight," 2.  ISE. Performance Index ");
writemem(0,10,26,Highlight," 3.  IAE. Performance Index ");
writemem(0,11,26,Highlight," 4.  ITAE. Performance Index ");
writemem(0,12,26,Highlight," 5.  ISE. PI. Hurwitz Method ");
writemem(0,21,5,Reverse,"  Choose Performance Index  ");
pi = selectmenu(8,23,35,1,5);

/*--------------- Read Optimal Process Data ------------------*/

movup(5,14,15,75,10,Normal);
movup(20,22,3,37,3,Normal);
writemem(0,5,26,Reverse,"   Optimal Criteria Data  ");
writemem(0,8,20,Highlight," Search Step Size [0.2]   := ");
writemem(0,9,20,Highlight," Precition Index  [0.001] := ");
writemem(0,10,20,Highlight," Max Iteration    [25]    := ");
writemem(0,12,37,Highlight,"Ready");
writemem(0,21,6,Reverse,"  Input Optimal Parameter  ");
cattrib(0,8,49,11,Reverse);
row = 8;
col = 49;
index = 1;
gotorc(row,col);
do {
    if(index != 4)
        showcur();
      else
        hidecur();
    p = getkey();
    writemem(0,20,49,Normal,"                        ");
    writemem(0,21,49,Normal,"                        ");
    if ( (p >= 0x30 && p<= 0x39) || (p==0x2E))
      {
        i = 0;
        do {
            if ( (p >= 0x30 && p<= 0x39) || (p==0x2E))
              {
                putch(p);
                cattrib(0,row,col,11,Reverse);
                mass[i] = p;
                i++;
                if(i>10)
                  {
                    i=0;
                    gotorc(row,col);
                  }
              }
            p = getkey();
            if (p == 0x08)
              {
                ii = col + i - 1;
                if (i!=0)
                  {
                    i--;
                    gotorc(row,ii);
```

```
                 putch(' ');
                 cattrib(0,row,col,11,Reverse);
                 gotorc(row,ii);
               }
           }
        } while( p != 0x0D);
    mass[i] = '\0';
    switch (index)
      {
        case 1 :
            {
            dx = atof(mass);
            break;
            }
        case 2 :
            {
            eps = atof(mass);
            break;
            }
        case 3 :
            {
            max = atoi(mass);
            break;
            }
      }
  }
switch(p)
  {
    case HOME_KEY:
    case PGUP_KEY:  {
            index = 1;
            cattrib(0,row,col,11,Highlight);
            row = 8;
            col = 49;
            cattrib(0,row,col,11,Reverse);
                gotorc(row,col);
            break;
            }
    case END_KEY:
    case PGDN_KEY:  {
            index = 4;
            cattrib(0,row,col,11,Highlight);
            col = 34;
            row = 12;
            cattrib(0,row,col,11,Reverse);
                gotorc(row,col);
            break;
            }
    case DOWN_KEY:  {
            cattrib(0,row,col,11,Highlight);
            index++;
            if(index > 4)
              {
                index = 1;
```

```
             row = 8;
             col = 49;
           }
           else if (index == 4)
             {
             row = 12;
             col = 34;
             }
             else



             row++;
        cattrib(0,row,col,11,Reverse);
            gotorc(row,col);
        break;
        }
case UP_KEY:    {
        cattrib(0,row,col,11,Highlight);
        index--;
        if(index < 1)
          {
            index = 4;
            row = 12;
            col = 34;
          }
          else if (index == 3)
            {
                row = 10;
                col = 49;
            }
            else
              {
            row--;
              }
        cattrib(0,row,col,11,Reverse);
            gotorc(row,col);
        break;
        }
case '\r':
case '\n':  {
        if (index==4)
            index = 100;
          else
            {
               cattrib(0,row,col,11,Highlight);
               index++;
               if(index > 4)
                {
                   index = 1;
                   row = 8;
                   col = 49;
                }
                else if (index == 4)
                  {
                    row = 12;
```

```
                            col = 34;
                          }
                          else
                            {
                                row++;
                            }
                       cattrib(0,row,col,11,Reverse);
                       gotorc(row,col);
                    }
                break;
                }
        default:  {
                    writemem(0,20,49,Reverse,"   Keypress Error ! ");
                    writemem(0,21,49,ReverseBlink,"  TRY  AGAIN ! ");
                    break;
                }
      }
} while (index != 100);
if (pi != 5)



 {
    if( (status == 0) || (status == 1) )
        damp1(gain,tdelay,para1,para2);
    if( status == 2 )
        damp2(gain,tdelay,para1,para2);
    mvc = pconst(pid[0],pid[1],pid[2],eps);
 }
 else
   {
     if (status == 0 || status ==1 )
       {
          o = para1+para2;
          l = para1*para2;
          para1 = o;
          para2 = l;
          hconst(gain,tdelay,para1,para2,q);
       }
   }
 writemem(0,1,36,Highlight,"OUTPUT");
 movup(5,13,15,75,9,Normal);
 movup(20,22,3,37,3,Normal);
 writemem(0,5,25,Highlight,"   Parameter of Controller   ");
 if ( pi ==1 )
     writemem(0,7,23,Highlight," No Overshoot Performance Index ");
   else if (pi == 2)
       writemem(0,7,27,Highlight," ISE. Performance Index ");
     else if (pi == 3)
         writemem(0,7,27,Highlight," IAE. Performance Index ");
       else if (pi == 4)
           writemem(0,7,26,Highlight," ITAE. Performance Index ");
           else if (pi == 5)
                  writemem(0,7,26,Highlight," ISE. PI. Hurtz Method ");
 writemem(0,8,30,Highlight,"Iteration #  ");
 writemem(0,10,20,Normal,"Proportional Band (PB)  := ");
writemem(0,11,20,Normal,"Integral time (Ti)        := ");
writemem(0,12,20,Normal,"Derivetive time (Td)      := ");
writemem(0,21,4,Reverse,"  Output Controller Parameter  ");
writemem(0,8,42,Reverse,"  0  ");
writeva(0,10,47,Reverse," %7.3f  ",pid[1]);
writeva(0,11,47,Reverse," %7.3f  ",pid[0]);
writeva(0,12,47,Reverse," %7.3f  ",pid[2]);
writemem(0,14,20,Highlight,"Performance Index Value := ");
powell(pi,pid,&result,dx,eps,max,q);
movup(20,22,3,37,3,Normal);
writemem(0,5,25,Highlight,"      Optimum Parameters        ");
writemem(0,8,30,Normal,"                    ");
writemem(0,10,20,Normal,"Proportional Band (PB)  := ");
writemem(0,11,20,Normal,"Integral time (Ti)      := ");
writemem(0,12,20,Normal,"Derivetive time (Td)    := ");
writemem(0,21,4,Reverse,"  Optimum Controller Parameter  ");
writeva(0,10,47,Reverse," %7.3lf  ",pid[1]);
writeva(0,11,47,Reverse," %7.3lf  ",pid[0]);
writeva(0,12,47,Reverse," %7.3lf  ",pid[2]);
writemem(0,14,20,Highlight,"Performance Index Value := ");
writeva(0,14,47,Reverse,"%7.3lf   ",result);
sound(500);
delay(200);
sound(2000);
delay(500);
nosound();
delay(1000);
}


void        damp1(double k,double l,double p1,double p2)

{

extern  double  ts,a[],b[],c[],d[];

double      p,q,r,a1,a2,ap,aq,ar;
double      x,y,m,n,e;
double      ceta1,expo1,aa,j;
int         i;

ceta1 = sqrt(3.0)*ts/l;
expo1 = exp(-3.0*ts/l);
p = 6.9282*expo1*sin(ceta1);
q = 2.0*expo1*cos(ceta1);
r = exp(-6.0*ts/l);
x = k/(p1*p2);
y = k/(p1*(p1-p2));
m = k/(p2*(p1-p2));
n = exp(-1.0*p1*ts);
e = exp(-1.0*p2*ts);
a[0] = 1.0;
a[1] = -1.0*(p+q);
```

```
a[2] = p+r;
b[0] = 0.0;
b[1] = q;
b[2] = -r;
c[0] = x+y-m;
c[1] = m*(n+1)-x*(n+e)-y*(e+1);
c[2] = x*n*e + y*e - m*n;
d[0] = 0.0;
d[1] = n+e;
d[2] = -1.0*n*e;

}

void        damp2(double k,double l,double p1,double p2)

{

extern  double  ts,a[],b[],c[],d[];

double      p,q,r,a1,a2,ap,aq,ar;
double      ceta1,expo1,aa,j;
int         i;

ceta1 = sqrt(3.0)*ts/l;
expo1 = exp(-3.0*ts/l);
p = 6.9282*expo1*sin(ceta1);
q = 2.0*expo1*cos(ceta1);
r = exp(-6.0*ts/l);
ceta1 = -1.0*p1*ts/2.0;
expo1 = exp(ceta1);
ceta1 = p2 - p1*p1/4.0;
aa = sqrt(ceta1);
ceta1 = ts*aa;
ap = expo1*(cos(ceta1)+p1*sin(ceta1)/(2.0*aa));
aq = 2.0*expo1*cos(ceta1);
ar = exp(-1.0*p1*ts);
a1 = k*(ap-aq+1)/p2;
a2 = k*(ar-ap)/p2;
a[0] = 1.0;
a[1] = -1.0*(p+q);


a[2] = (p+r);
b[0] = 0.0;
b[1] =  q;
b[2] = -1.0*r;
c[0] = 0;
c[1] = a1;
c[2] = a2;
d[0] = 0;
d[1] = aq;
d[2] = -1.0*ar;

}
```

```
double      pconst(double ti,double pb,double td,double eps)

{

extern  double  ts,step,mvc,pvc,anc,bnc,cnc;
extern  double  maxdelaytime,maxsteadytime,presentsv;
extern  double  a[],b[],c[],d[],hilim,lowlim;
double          u[4],v[4],mv[10],o[10],en[10];
double          an[2],bn[2],cn[2];
double          kp,tf,t,bias,x,y,n,epps;
double          slope,sumx,sumy,sumxy,sumxx;
int             count,i,status,num;

/**************** initialize **********************/

t = 0;
status = 0;
count = 0;
bias = 0.5;
num = 10;
epps = 0.0001;
kp = 100/pb;
n = 8;
tf = td/n;
for(i=0;i<=1;i++)
  {
    o[i] = u[i] = mv[i] = v[i] = bias;
    an[i] = bn[i] = cn[i] = en[i] = 0.0;
  }
for(i=2;i<=3;i++)
  {
    o[i] = u[i] = mv[i] = v[i] = bias;
    en[i] = 0.0;
  }
for(i=4;i<=9;i++)
  {
    o[i] = mv[i] = bias;
    en[i] = 0.0;
  }

/*********** Find steady state of MV[] ************/

do {
    t = t + ts;
    count = count + 1;
    en[0] = u[0] - o[0];
    /******** Stop Cond. ********/
    if( t > maxdelaytime && count > 10 )
      {
        sumx = sumy = sumxy = sumxx = 0.0;
  for( i=0;i<=9;i++)
    {
      x = t - (i*ts);
      y = o[i];
      sumx = sumx + x;
      sumy = sumy + y;
      sumxy = sumxy + x*y;
      sumxx = sumxx + pow(x,2);
    }
  slope = (num*sumxy - sumx*sumy)/(num*sumxx - pow(sumx,2.0));
  if( fabs(slope) < epps )
    {
      if ( fabs(en[0]) > eps )
        {
          if ( t >= 2*maxsteadytime )
              status = 1;
        }
        else
          status = 1;
    }
    else
      if ( t > 3*maxsteadytime )
          status = 1;
 }
/******** Controller ********/
an[0] = Kp*en[0];
bn[0] = bn[1] + an[0]*ts/ti;
cn[0] = (tf*cn[1] + kp*td*(o[0]-o[3]-3.0*o[2]+3.0*o[1])
    /(6.0*(tf+ts)));
mv[0] = an[0]+bn[0]-cn[0]+bias;
if(mv[0]>hilim)
    mv[0]=hilim;
if(mv[0]<lowlim)
    mv[0]=lowlim;
/********   Process   ********/
v[0] = mv[0] + a[1]*mv[1] + a[2]*mv[2] + b[1]*v[1] + b[2]*v[2];
o[0] = c[0]*v[0] + c[1]*v[1] + c[2]*v[2] + d[1]*o[1] + d[2]*o[2];
/******** Iteration  ********/
an[1] = an[0];
bn[1] = bn[0];
cn[1] = cn[0];
 for(i=8;i>=3;i--)
   {
     mv[i+1] = mv[i];
     o[i+1] = o[i];
     en[i+1] = en[i];
   }
 for(i=2;i>=0;i--)
   {
     mv[i+1] = mv[i];
     o[i+1] = o[i];
     en[i+1] = en[i];
     u[i+1] = u[i];
     v[i+1] = v[i];
```

```
      }
    } while (status==0);
anc = an[0];
bnc = bn[0];
cnc = cn[0];
pvc = o[0];
return(mv[0]);
}



void        powell(pi,kid,v,dx,eps,max,q)

double      kid[],*v,dx,eps,q[];
int         pi,max;

{

extern  double  step;
double      xn[4][3],s[3][3];
double      x[3],yn[4],f[4],del[4],z[3];
double      y,g,h;
int         i,j,l,count;

/* initialize */

for ( i=0;i<=2;i++ )
  {
    xn[0][i] = x[i] = kid[i];
    for ( j=0;j<=2;j++ )
      {
        if ( i==j )
            s[i][j] = 1;
          else
            s[i][j] = 0;
      }
  }
switch (pi)
  {
    case 1 :
      {
        y = yn[0] = f[0] = nonshoot(xn[0][0],xn[0][1],xn[0][2],eps);
        for(i=0;i<=2;i++)
         {
           g = s[1][i];
           s[1][i]=s[0][i];
           s[0][i]=g;
         }
        break;
      }
    case 2 :
      {
        y = yn[0] = f[0] = ise(xn[0][0],xn[0][1],xn[0][2],eps);
        break;
      }
```

```
   case 3 :
     {
       y = yn[0] = f[0] = iae(xn[0][0],xn[0][1],xn[0][2],eps);
       break;
     }
   case 4 :
     {
       y = yn[0] = f[0] = itae(xn[0][0],xn[0][1],xn[0][2],eps);
         break;
     }
   case 5 :
     {
       isehurz(q,xn[0][1],xn[0][0],xn[0][2],&y);
       yn[0] = f[0] = y;
       break;
     }
   default :
       break;
 }


writeva(0,14,47,Reverse,"  %7.3lf ",f[0]);
count = 0;

/* END initialize */

do
  {
    count = count + 1;
    /* step #1 */
    for ( i=0;i<=2;i++ )
      {
        dscp(pi,x,&y,dx,s[i],eps,max,q);
        f[i+1] = y;
      }
    writemem(0,8,42,Normal,"       ");
    writemem(0,10,46,Normal,"            ");
    writemem(0,11,46,Normal,"            ");
    writemem(0,12,46,Normal,"            ");
    writemem(0,14,46,Normal,"            ");
    writeva(0,8,42,Reverse," %d ",count);
    writeva(0,10,47,Reverse," %7.3f  ",x[1]);
    writeva(0,11,47,Reverse," %7.3f  ",x[0]);
    writeva(0,12,47,Reverse," %7.3f  ",x[2]);
    writeva(0,14,47,Reverse," %7.3lf ",y);
    sound(1600);
    delay(500);
    nosound();

    /* test for stop */
    g = 0;
    for ( i=0;i<=2;i++ )
      {
        h = (x[i]-xn[0][i])*(x[i]-xn[0][i]);
    g = g + h;
  }
g = sqrt(g);
if ( g <= eps )
  {
    for ( i=0;i<=2;i++ )
        kid[i] = x[i];
    *v = y;
    break;
  }
/* step #2 */
for(i=0;i<=2;i++)
  {
    xn[1][i] = x[i];
    xn[2][i] = 2.0*xn[1][i] - xn[0][i];
    if ( xn[2][i] <= 0.0 )
       xn[2][i] = 0.00001;
    if ( xn[2][i] > 9999.0)
      xn[2][i] = 9999.0;
  }
yn[1] = f[3];
switch (pi)
  {
    case 1 :
      {
        yn[2] = nonshoot(xn[2][0],xn[2][1],xn[2][2],eps);
        break;
      }
    case 2 :
      {



        yn[2] = ise(xn[2][0],xn[2][1],xn[2][2],eps);
        break;
      }
    case 3 :
      {
        yn[2] = iae(xn[2][0],xn[2][1],xn[2][2],eps);
        break;
      }
    case 4 :
      {
        yn[2] = itae(xn[2][0],xn[2][1],xn[2][2],eps);
        break;
      }
    case 5 :
      {
        isehurz(q,xn[0][1],xn[0][0],xn[0][2],&yn[2]);
        break;
      }
    default :
        break;
  }
/* step #3 */
for(i=1;i<=3;i++)
```

```
  {
    del[i] = fabs(f[i-1] - f[i]);
  }
if ( del[1] < del [2] )
  {
    del[0] = del[2];
    for ( j=0;j<=2;j++ )
      {
        z[j] = s[1][j];
      }
    l = 1;
  }
 else
  {
    del[0] = del[1];
    for ( j=0;j<=2;j++ )
      {
        z[j] = s[0][j];
      }
    l = 0;
  }
if ( del[3] > del [0] )
  {
    del[0] = del[3];
    for ( j=0;j<=2;j++ )
      {
        z[j] = s[2][j];
      }
    l = 2;
  }
g = yn[0]-yn[1]-del[0];
h = pow(g,2);
g = yn[0] - yn[2];
g = pow(g,2);
if ( (yn[2] >= yn[0]) ||
   ( (yn[0]-2.0*yn[1]+yn[2])*h >= 0.5*del[0]*g ))
  {
    if ( yn[1] <= yn[2] )
      {
        for ( i=0;i<=2;i++ )


            xn[0][i] = x[i] = xn[1][i];
        y = yn[1];
      }
      else
        {
        for ( i=0;i<=2;i++ )
            xn[0][i] = x[i] = xn[2][i];
        y = yn[2];
        }
   }
  else
   {
     g = 0;
```

```
       for ( i=0;i<=2;i++ )
         {
           h = (xn[1][i]-xn[0][i])*(xn[1][i]-xn[0][i]);
           g = g + h;
         }
       g = sqrt(g);
       for ( i=0;i<=2;i++ )
         {
           z[i] = xn[1][i] - xn[0][i];
           z[i] = z[i]/g;
         }
       for ( i=1;i<=2;i++ )
         {
           if (i == 2)
               for ( j=0;j<=2;j++ )
                   s[i][j] = z[j];
             else
               for ( j=0;j<=2;j++ )
                   s[i][j] = s[i+1][j];
         }
        if ( yn[1] <= yn[2] )
           {
            for ( i=0;i<=2;i++ )
                xn[0][i] = x[i] = xn[1][i];
            y = yn[1];
           }
          else
            {
            for ( i=0;i<=2;i++ )
               xn[0][i] = x[i] = xn[2][i];
            y = yn[2];
            }
      }
   } while ( count < max );
if ( count == max )
  {
    if ( yn[1] <= yn[2] )
      {
        for ( i=0;i<=2;i++ )
            kid[i] = xn[1][i];
        *v = yn[1];
      }
     else
       {
        for ( i=0;i<=2;i++ )
            kid[i] = xn[2][i];
        *v = yn[2];
       }
   }
 }
```

```
void        dscp(pi,xo,yo,dx,di,eps,max,pq)

double      xo[],*yo,dx,di[],eps,pq[];
int         pi,max;

{

extern  double  step;
double          x[4][3],y[4],xz[3],a[3],b[3],c[3];
double          yz,z,q[2],w;
int             i,j,k,l,count,state;

state = 0;
for(i=0;i<=2;i++)
    x[0][i] = xo[i];
y[0] = *yo;
for(i=0;i<=2;i++)
  {
     x[1][i] = x[0][i] + dx*di[i];
     if( x[1][i] < 0 )
    x[1][i] = 0.00001;
     if(x[1][i] > 9999.0)
    x[1][i] = 9999.0;
  }
switch (pi)
  {
    case 1 :
      {
        y[1] = nonshoot(x[1][0],x[1][1],x[1][2],eps);
        break;
      }
    case 2 :
      {
        y[1] = ise(x[1][0],x[1][1],x[1][2],eps);
        break;
      }
    case 3 :
      {
        y[1] = iae(x[1][0],x[1][1],x[1][2],eps);
        break;
      }
    case 4 :
      {
        y[1] = itae(x[1][0],x[1][1],x[1][2],eps);
        break;
      }
    case 5 :
      {
        isehurz(pq,x[1][1],x[1][0],x[1][2],&y[1]);
        break;
      }
    default :
        break;
  }
```

```
if ( y[1] >= y[0] )
  {
    dx = -2.0*dx;
    for(i=0;i<=2;i++)
      {
        a[0] = x[0][i];
        x[0][i] = x[1][i];
        x[1][i] = a[0];



      }
    a[0] = y[0];
    y[0] = y[1];
    y[1] = a[0];
  }
 else
    dx = 2.0*dx;
for(i=0;i<=2;i++)
  {
    x[2][i] = x[1][i] + dx*di[i];
    if( x[2][i] < 0 )
    x[2][i] = 0.00001;
    if(x[2][i] > 9999.0)
    x[2][i] = 9999.0;
  }
switch (pi)
  {
    case 1 :
      {
        y[2] = nonshoot(x[2][0],x[2][1],x[2][2],eps);
        break;
      }
    case 2 :
      {
        y[2] = ise(x[2][0],x[2][1],x[2][2],eps);
        break;
      }
    case 3 :
      {
        y[2] = iae(x[2][0],x[2][1],x[2][2],eps);
        break;
      }
    case 4 :
      {
        y[2] = itae(x[2][0],x[2][1],x[2][2],eps);
        break;
      }
    case 5 :
      {
        isehurz(pq,x[2][1],x[2][0],x[2][2],&y[2]);
        break;
      }
    default :
        break;
  }
q[0] = 0;
q[1] = 0;
while ( y[2] <= y[1] && fabs(y[1]-y[2]) > 0.00001 && state == 0 )
  {
    for (i=0;i<=1;i++ )
       {
        y[i] = y[i+1];
        for(k=0;k<=2;k++)
            x[i][k] = x[i+1][k];
       }
    dx = 2.0*dx;
    for(i=0;i<=2;i++)
      {
        x[2][i] = x[1][i] + dx*di[i];
        if ( x[2][i] <= 0 )
        x[2][i] = 0.00001;
        if ( x[2][i] > 9999.0)
        x[2][i] = 9999.0;



      }
    for(i=0;i<=2;i++)
      {
        q[1] = q[1] + x[2][i];
      }
    if( q[1] == q[0])
        state = 1;
      else
        {
        q[0] = q[1];
        q[1] = 0;
        switch (pi)
         {
           case 1 :
            {
                y[2] = nonshoot(x[2][0],x[2][1],x[2][2],eps);
                break;
            }
           case 2 :
            {
                y[2] = ise(x[2][0],x[2][1],x[2][2],eps);
                break;
            }
           case 3 :
            {
                y[2] = iae(x[2][0],x[2][1],x[2][2],eps);
                break;
            }
           case 4 :
            {
                y[2] = itae(x[2][0],x[2][1],x[2][2],eps);
                break;
            }
            case 5 :
             {
                isehurz(pq,x[2][1],x[2][0],x[2][2],&y[2]);
                break;
             }
           default :
                break;
         }
        }
  }
for(i=0;i<=2;i++)
    x[3][i] = x[2][i];
y[3] = y[2];
for(i=0;i<=2;i++)
    x[2][i] = x[3][i] - 0.5*dx*di[i];
switch (pi)
  {
    case 1 :
      {
        y[2] = nonshoot(x[2][0],x[2][1],x[2][2],eps);
        break;
      }
    case 2 :
      {
        y[2] = ise(x[2][0],x[2][1],x[2][2],eps);
        break;
      }
    case 3 :
      {
        y[2] = iae(x[2][0],x[2][1],x[2][2],eps);


        break;
      }
    case 4 :
      {
        y[2] = itae(x[2][0],x[2][1],x[2][2],eps);
        break;
      }
    case 5 :
      {
        isehurz(pq,x[2][1],x[2][0],x[2][2],&y[2]);
        break;
      }
    default :
        break;
  }
 if ( y[1] >= y[2] )
     for (i=0;i<=2;i++)
       {
         for(k=0;k<=2;k++)
```

```
            x[i][k] = x[i+1][k];
        y[i] = y[i+1];
      }
if ( y[0] <= y[1] )
  {
    for(i=0;i<=2;i++)
        xz[i] = x[0][i];
    yz = y[0];
  }
 else
  {
    for(i=0;i<=2;i++)
        xz[i] = x[1][i];
    yz = y[1];
  }
if( yz >= y[2] )
  {
    for(i=0;i<=2;i++)
        xz[i] = x[2][i];
    yz = y[2];
  }
count = 0;
j = 1;
q[0] = 0;
q[1] = 0;
do
{
    count = count + 1;
    for(i=0;i<=2;i++)
      {
        if ( di[i] != 0 )
          {
            a[i] = pow(x[0][i],2.0);
            b[i] = pow(x[1][i],2.0);
            c[i] = pow(x[2][i],2.0);
            w = ((x[1][i]-x[2][i])*y[0] + (x[2][i]-x[0][i])*y[1] +
                (x[0][i]-x[1][i])*y[2] );
            if (w == 0)
                w = 0.00001;
            x[3][i] = 0.5*( (b[i]-c[i])*y[0] + (c[i]-a[i])*y[1] +
                      (a[i]-b[i])*y[2] )/w;
            if ( x[3][i]<=0 )
               x[3][i] = 0.00001;
            if (x[3][i] > 9999.0)



               x[3][i] = 9999.0;
           }
       }
     for(i=0;i<=2;i++)
       {
         q[1] = q[1] + x[3][i];
       }
```

```
if( q[1] == q[0])
  count = max;
 else
  {
  q[0] = q[1];
  q[1] = 0;
      switch (pi)
   {
     case 1 :
     {
        y[3] = nonshoot(x[3][0],x[3][1],x[3][2],eps);
        break;
     }
     case 2 :
     {
        y[3] = ise(x[3][0],x[3][1],x[3][2],eps);
        break;
     }
     case 3 :
     {
        y[3] = iae(x[3][0],x[3][1],x[3][2],eps);
        break;
     }
     case 4 :
     {
        y[3] = itae(x[3][0],x[3][1],x[3][2],eps);
        break;
     }
     case 5 :
      {
        isehurz(pq,x[3][1],x[3][0],x[3][2],&y[3]);
        break;
      }
     default :
    break;
   }
  }
 if ( fabs(yz - y[3]) < eps )
  {
   if ( yz < y[3] )
    {
     j = 0;
     for(i=0;i<=2;i++)
       xo[i]= xz[i];
     *yo = yz;
    }
   else
    {
     j = 0;
     for(i=0;i<=2;i++)
       xo[i]= x[3][i];
     *yo = y[3];
    }
  }
```

```
     else
      {


        for (i=0;i<=2;i++)
          {
            for (k=0;k<=2;k++)
              {
                if ( y[k] > y[k+1] )
                  {
                    z = y[k];
                    y[k] = y[k+1];
                    y[k+1] = z;
                    for(l=0;l<=2;l++)
                      {
                        z = x[k][l];
                        x[k][l] = x[k+1][l];
                        x[k+1][l] = z;
                      }
                  }
              }
          }
        for (i=0;i<=2;i++)
            xz[i] = x[0][i];
        yz = y[0];
      }
} while ( (j!=0) && (count!=max) );
if ( count == max )
  {
    for(i=0;i<=2;i++)
        xo[i]= x[0][i];
    *yo = y[0];
  }
}

double      iae(double ti,double pb,double td,double eps)

{

extern  double  ts,step,mvc,pvc,anc,bnc,cnc;
extern  double  maxdelaytime,maxsteadytime,presentsv;
extern  double  a[],b[],c[],d[],hilim,lowlim;
double          u[4],mv[10],v[4],o[10],en[10];
double          an[2],bn[2],cn[2];
double          kp,tf,t,er,inp,bias,x,y,n,epps;
double          slope,sumx,sumy,sumxy,sumxx;
int             count,i,status,num;

/**************** initialize *********************/

t = 0;
epps = 0.0001;
status = 0;
count = 0;
```

```
bias = 0.5;
er = 0.0;
num = 10;
kp = 100/pb;
n = 8.0;
tf = td/n;
for(i=0;i<=1;i++)
  {
    mv[i] = v[i] = mvc;
    o[i] = pvc;
    u[i] = presentsv/100;
    an[i] = anc;
    bn[i] = bnc;



    cn[i] = cnc;
    en[i] = 0.0;
  }
 for(i=2;i<=3;i++)
  {
    mv[i] = v[i] = mvc;
    o[i] = pvc;
    u[i] = presentsv/100;
    en[i] = 0.0;
  }
 for(i=4;i<=9;i++)
  {
    mv[i] = mvc;
    o[i] = pvc;
    en[i] = 0.0;
  }

 inp = u[0] + step;

 /*********** Find steady state of MV[] ************/

 do {
     t = t + ts;
     count = count + 1;
     en[0] = u[0] - o[0];
     /******** Stop Cond. ********/
     if( count > 10 && t > 3*maxdelaytime )
       {
         sumx = sumy = sumxy = sumxx = 0.0;
         for( i=0;i<=9;i++)
           {
             x = t-(i*ts);
             y = o[i];
             sumx = sumx + x;
             sumy = sumy + y;
             sumxy = sumxy + x*y;
             sumxx = sumxx + pow(x,2);
           }
```

```
        slope = (num*sumxy - sumx*sumy)/(num*sumxx - pow(sumx,2.0));
        if( fabs(slope) < epps )
           {
              if ( fabs(en[0]) > eps )
                 {
                    if ( t >= 2*maxsteadytime )
                          status = 1;
                 }
                 else
                    status = 1;
           }
           else
              if ( t > 3*maxsteadytime )
                    status = 1;
      }
   /******** Controller ********/
   an[0] = kp*en[0];
   bn[0] = bn[1] + an[0]*ts/ti;
   cn[0] = (tf*cn[1] + kp*td*(o[0]-o[3]-3.0*o[2]+3.0*o[1])/
               (6.0*(tf+ts)));
   mv[0] = an[0]+bn[0]+cn[0]+bias;
   if(mv[0]>hilim)
         mv[0]=hilim;
   if(mv[0]<lowlim)
         mv[0]=lowlim;



    /********   Process   ********/
    v[0] = mv[0] + a[1]*mv[1] + a[2]*mv[2] + b[1]*v[1] + b[2]*v[2];
    o[0] = c[0]*v[0] + c[1]*v[1] + c[2]*v[2] + d[1]*o[1] + d[2]*o[2];
    /********  Iteration  ********/
    an[1] = an[0];
    bn[1] = bn[0];
    cn[1] = cn[0];
    for(i=8;i>=3;i--)
       {
          mv[i+1] = mv[i];
          o[i+1] = o[i];
          en[i+1] = en[i];
       }
    for(i=2;i>=0;i--)
       {
          mv[i+1] = mv[i];
          o[i+1] = o[i];
          en[i+1] = en[i];
          u[i+1] = u[i];
          v[i+1] = v[i];
       }
    } while (status==0);

/***************    Loop    **********************/

pvc = o[0];
mvc = mv[0];
```

```
t = 0;
count = 0;
status = 0;
do {
    t = t + ts;
    count = count + 1;
    u[0] = inp;
    en[0] = u[0] - o[0];
    /********    IAE     ********/
    er = er + fabs(en[0]);
    /******** Stop Cond. ********/
    if( t > maxdelaytime && count > 10 )
      {
        sumx = sumy = sumxy = sumxx = 0.0;
        for( i=0;i<=9;i++)
          {
            x = t-(i*ts);
            y = o[i];
            sumx = sumx + x;
            sumy = sumy + y;
            sumxy = sumxy + x*y;
            sumxx = sumxx + pow(x,2);
          }
        slope = (num*sumxy - sumx*sumy)/(num*sumxx - pow(sumx,2.0));
        if( fabs(slope) < epps )
          {
            if ( fabs(en[0]) > eps )
              {
                if ( t >= 2*maxsteadytime )
                    status = 1;
              }
              else
                status = 1;
          }
          else
            if ( t > 3*maxsteadytime )


                status = 1;
      }
     /******** Controller ********/
     an[0] = kp*en[0];
     bn[0] = bn[1] + an[0]*ts/ti;
     cn[0] = (tf*cn[1] + kp*td*(o[0]-o[3]-3.0*o[2]+3.0*o[1])
             /(6.0*(tf+ts)));
     mv[0] = an[0]+bn[0]+cn[0]+bias;
     if(mv[0]>hilim)
         mv[0]=hilim;
     if(mv[0]<lowlim)
         mv[0]=lowlim;
     /********   Process   ********/
     v[0] = mv[0] + a[1]*mv[1] + a[2]*mv[2] + b[1]*v[1] + b[2]*v[2];
     o[0] = c[0]*v[0] + c[1]*v[1] + c[2]*v[2] + d[1]*o[1] + d[2]*o[2];
     /********  Iteration  ********/
    an[1] = an[0];
    bn[1] = bn[0];
    cn[1] = cn[0];
    for(i=8;i>=3;i--)
      {
        mv[i+1] = mv[i];
        o[i+1] = o[i];
        en[i+1] = en[i];
      }
    for(i=2;i>=0;i--)
      {
        mv[i+1] = mv[i];
        o[i+1] = o[i];
        en[i+1] = en[i];
        u[i+1] = u[i];
        v[i+1] = v[i];
      }
   } while (status==0);
er = er*ts;
return(er);
}

double      ise(double ti,double pb,double td,double eps)

{

extern  double  ts,step,mvc,pvc,anc,bnc,cnc;
extern  double  maxdelaytime,maxsteadytime,presentsv;
extern  double  a[],b[],c[],d[],hilim,lowlim;
double          u[4],mv[10],v[4],o[10],en[10];
double          an[2],bn[2],cn[2];
double          kp,tf,t,er,inp,bias,x,y,n,epps,err;
double          slope,sumx,sumy,sumxy,sumxx;
int             count,i,status,num;

/**************** initialize **********************/

t = 0;
epps = 0.0001;
status = 0;
count = 0;
bias = 0.5;
er = 0.0;
num = 10;
kp = 100/pb;
n = 8.0;
tf = td/n;



for(i=0;i<=1;i++)
  {
    mv[i] = v[i] = mvc;
    o[i] = pvc;
    u[i] = presentsv/100;
    an[i] = anc;
    bn[i] = bnc;
    cn[i] = cnc;
    en[i] = 0.0;
  }
for(i=2;i<=3;i++)
  {
    mv[i] = v[i] = mvc;
    o[i] = pvc;
    u[i] = presentsv/100;
    en[i] = 0.0;
  }
for(i=4;i<=9;i++)
  {
    mv[i] = mvc;
    o[i] = pvc;
    en[i] = 0.0;
  }

inp = u[0] + step;

/*********** Find steady state of MV[] ************/

do {
    t = t + ts;
    count = count + 1;
    en[0] = u[0] - o[0];
    /******** Stop Cond. ********/
    if( count > 10 && t > 3*maxdelaytime )
      {
        sumx = sumy = sumxy = sumxx = 0.0;
        for( i=0;i<=9;i++)
          {
            x = t-(i*ts);
            y = o[i];
            sumx = sumx + x;
            sumy = sumy + y;
            sumxy = sumxy + x*y;
            sumxx = sumxx + pow(x,2);
          }
        slope = (num*sumxy - sumx*sumy)/(num*sumxx - pow(sumx,2.0));
        if( fabs(slope) < epps )
          {
            if ( fabs(en[0]) > eps )
              {
                if ( t >= 2*maxsteadytime )
                    status = 1;
              }
              else
                status = 1;
          }
          else
            if ( t > 3*maxsteadytime )
                status = 1;
      }
```

```
     /******** Controller ********/
     an[0] = kp*en[0];
     bn[0] = bn[1] + an[0]*ts/ti;



     cn[0] = (tf*cn[1] + kp*td*(o[0]-o[3]-3.0*o[2]+3.0*o[1])
             /(6.0*(tf+ts)));
     mv[0] = an[0]+bn[0]+cn[0]+bias;
     if(mv[0]>hilim)
         mv[0]=hilim;
     if(mv[0]<lowlim)
         mv[0]=lowlim;
     /********   Process   ********/
     v[0] = mv[0] + a[1]*mv[1] + a[2]*mv[2] + b[1]*v[1] + b[2]*v[2];
     o[0] = c[0]*v[0] + c[1]*v[1] + c[2]*v[2] + d[1]*o[1] + d[2]*o[2];
     /********  Iteration  ********/
     an[1] = an[0];
     bn[1] = bn[0];
     cn[1] = cn[0];
     for(i=8;i>=3;i--)
       {
         mv[i+1] = mv[i];
         o[i+1] = o[i];
         en[i+1] = en[i];
       }
     for(i=2;i>=0;i--)
       {
         mv[i+1] = mv[i];
         o[i+1] = o[i];
         en[i+1] = en[i];
         u[i+1] = u[i];
         v[i+1] = v[i];
       }
    } while (status==0);

/****************      Loop      **********************/

pvc = o[0];
mvc = mv[0];
t = 0;
count = 0;
status = 0;
do {
    t = t + ts;
    count = count + 1;
    u[0] = inp;
    en[0] = u[0] - o[0];
    err = en[0]*100.0;
    /********    ISE      ********/
    er = er + err*err;
    /******** Stop Cond. ********/
    if( t > maxdelaytime && count > 10 )
      {
        sumx = sumy = sumxy = sumxx = 0.0;
```

```
  for( i=0;i<=9;i++)
   {
    x = t-(i*ts);
    y = o[i];
    sumx = sumx + x;
    sumy = sumy + y;
    sumxy = sumxy + x*y;
    sumxx = sumxx + pow(x,2);
   }
  slope = (num*sumxy - sumx*sumy)/(num*sumxx - pow(sumx,2.0));
  if( fabs(slope) < epps )
   {
    if ( fabs(en[0]) > eps )
     {


      if ( t >= 2*maxsteadytime )
        status = 1;
     }
     else
      status = 1;
   }
   else
    if ( t > 3*maxsteadytime )
      status = 1;
 }
/******** Controller ********/
an[0] = kp*en[0];
bn[0] = bn[1] + an[0]*ts/ti;
cn[0] = (tf*cn[1] + kp*td*(o[0]-o[3]-3.0*o[2]+3.0*o[1])
    /(6.0*(tf+ts)));
mv[0] = an[0]+bn[0]+cn[0]+bias;
if(mv[0]>hilim)
  mv[0]=hilim;
if(mv[0]<lowlim)
  mv[0]=lowlim;
/********  Process  ********/
v[0] = mv[0] + a[1]*mv[1] + a[2]*mv[2] + b[1]*v[1] + b[2]*v[2];
o[0] = c[0]*v[0] + c[1]*v[1] + c[2]*v[2] + d[1]*o[1] + d[2]*o[2];
/********  Iteration  ********/
an[1] = an[0];
bn[1] = bn[0];
cn[1] = cn[0];
for(i=8;i>=3;i--)
 {
  mv[i+1] = mv[i];
  o[i+1] = o[i];
  en[i+1] = en[i];
 }
for(i=2;i>=0;i--)
 {
  mv[i+1] = mv[i];
  o[i+1] = o[i];
```

```
        en[i+1] = en[i];
        u[i+1] = u[i];
        v[i+1] = v[i];
      }
   } while (status==0);
er = er*ts;
return(er);
}

double      itae(double ti,double pb,double td,double eps)

{

extern  double  ts,step,mvc,pvc,anc,bnc,cnc;
extern  double  maxdelaytime,maxsteadytime,presentsv;
extern  double  a[],b[],c[],d[],hilim,lowlim;
double          u[4],mv[10],v[4],o[10],en[10];
double          an[2],bn[2],cn[2];
double          kp,tf,t,er,inp,bias,x,y,n,epps;
double          slope,sumx,sumy,sumxy,sumxx;
int             count,i,status,num;

/**************** initialize *********************/

t = 0;
epps = 0.0001;



status = 0;
count = 0;
bias = 0.5;
er = 0.0;
num = 10;
kp = 100/pb;
n = 8.0;
tf = td/n;
for(i=0;i<=1;i++)
   {
     mv[i] = v[i] = mvc;
     o[i] = pvc;
     u[i] = presentsv/100;
     an[i] = anc;
     bn[i] = bnc;
     cn[i] = cnc;
     en[i] = 0.0;
   }
 for(i=2;i<=3;i++)
   {
     mv[i] = v[i] = mvc;
     o[i] = pvc;
     u[i] = presentsv/100;
     en[i] = 0.0;
   }
```

```
for(i=4;i<=9;i++)
  {
    mv[i] = mvc;
    o[i] = pvc;
    en[i] = 0.0;
  }

inp = u[0] + step;

/*********** Find steady state of MV[] ************/

do {
    t = t + ts;
    count = count + 1;
    en[0] = u[0] - o[0];
    /******** Stop Cond. ********/
    if( count > 10 && t > 3*maxdelaytime )
      {
        sumx = sumy = sumxy = sumxx = 0.0;
        for( i=0;i<=9;i++)
          {
            x = t-(i*ts);
            y = o[i];
            sumx = sumx + x;
            sumy = sumy + y;
            sumxy = sumxy + x*y;
            sumxx = sumxx + pow(x,2);
          }
        slope = (num*sumxy - sumx*sumy)/(num*sumxx - pow(sumx,2.0));
        if( fabs(slope) < epps )
          {
            if ( fabs(en[0]) > eps )
              {
                if ( t >= 2*maxsteadytime )
                    status = 1;
              }
              else
                status = 1;



          }
          else
            if ( t > 3*maxsteadytime )
                status = 1;
      }
      /******** Controller ********/
      an[0] = kp*en[0];
      bn[0] = bn[1] + an[0]*ts/ti;
      cn[0] = (tf*cn[1] + kp*td*(o[0]-o[3]-3.0*o[2]+3.0*o[1])
              /(6.0*(tf+ts)));
      mv[0] = an[0]+bn[0]+cn[0]+bias;
      if(mv[0]>hilim)
          mv[0]=hilim;
      if(mv[0]<lowlim)
          mv[0]=lowlim;
    /********   Process  ********/
    v[0] = mv[0] + a[1]*mv[1] + a[2]*mv[2] + b[1]*v[1] + b[2]*v[2];
    o[0] = c[0]*v[0] + c[1]*v[1] + c[2]*v[2] + d[1]*o[1] + d[2]*o[2];
    /********  Iteration  ********/
    an[1] = an[0];
    bn[1] = bn[0];
    cn[1] = cn[0];
    for(i=8;i>=3;i--)
      {
        mv[i+1] = mv[i];
        o[i+1] = o[i];
        en[i+1] = en[i];
      }
    for(i=2;i>=0;i--)
      {
        mv[i+1] = mv[i];
        o[i+1] = o[i];
        en[i+1] = en[i];
        u[i+1] = u[i];
        v[i+1] = v[i];
      }
   } while (status==0);

/****************    Loop    **********************/

pvc = o[0];
mvc = mv[0];
t = 0;
count = 0;
status = 0;
do {
    t = t + ts;
    count = count + 1;
    u[0] = inp;
    en[0] = u[0] - o[0];
    /********     ITAE     ********/
    er = er + t*fabs(en[0]);
    /******** Stop Cond. ********/
    if( t > maxdelaytime && count > 10 )
      {
        sumx = sumy = sumxy = sumxx = 0.0;
        for( i=0;i<=9;i++)
          {
            x = t-(i*ts);
            y = o[i];
            sumx = sumx + x;
            sumy = sumy + y;
            sumxy = sumxy + x*y;


            sumxx = sumxx + pow(x,2);
          }
        slope = (num*sumxy - sumx*sumy)/(num*sumxx - pow(sumx,2.0));
   if( fabs(slope) < epps )
    {
     if ( fabs(en[0]) > eps )
      {
       if ( t >= 2*maxsteadytime )
         status = 1;
      }
      else
       status = 1;
    }
    else
     if ( t > 3*maxsteadytime )
       status = 1;
  }
 /******** Controller ********/
 an[0] = kp*en[0];
 bn[0] = bn[1] + an[0]*ts/ti;
 cn[0] = (tf*cn[1] + kp*td*(o[0]-o[3]-3.0*o[2]+3.0*o[1])
     /(6.0*(tf+ts)));
 mv[0] = an[0]+bn[0]+cn[0]+bias;
 if(mv[0]>hilim)
   mv[0]=hilim;
 if(mv[0]<lowlim)
   mv[0]=lowlim;
 /********   Process   ********/
 v[0] = mv[0] + a[1]*mv[1] + a[2]*mv[2] + b[1]*v[1] + b[2]*v[2];
 o[0] = c[0]*v[0] + c[1]*v[1] + c[2]*v[2] + d[1]*o[1] + d[2]*o[2];
 /********  Iteration  ********/
 an[1] = an[0];
 bn[1] = bn[0];
 cn[1] = cn[0];
 for(i=8;i>=3;i--)
  {
   mv[i+1] = mv[i];
   o[i+1] = o[i];
   en[i+1] = en[i];
  }
 for(i=2;i>=0;i--)
  {
   mv[i+1] = mv[i];
   o[i+1] = o[i];
   en[i+1] = en[i];
   u[i+1] = u[i];
   v[i+1] = v[i];
  }
 } while (status==0);
er = er*ts;
return(er);
}
```

```
double      nonshoot(double ti,double pb,double td,double eps)

{

extern  double  ts,step,mvc,pvc,anc,bnc,cnc;
extern  double  maxdelaytime,maxsteadytime,presentsv;
extern  double  a[],b[],c[],d[],hilim,lowlim;
double           u[4],mv[10],v[4],o[10],en[10];
double               an[2],bn[2],cn[2];
double          kp,tf,t,er,inp,bias,x,y,n,epps;



double               slope,sumx,sumy,sumxy,sumxx,weight;
int             count,i,status,num;

/**************** initialize **********************/

t = 0;
epps = 0.0001;
status = 0;
count = 0;
bias = 0.5;
er = 0.0;
num = 10;
kp = 100/pb;
n = 8.0;
tf = td/n;
for(i=0;i<=1;i++)
  {
    mv[i] = v[i] = mvc;
    o[i] = pvc;
    u[i] = presentsv/100;
    an[i] = anc;
    bn[i] = bnc;
    cn[i] = cnc;
    en[i] = 0.0;
  }
for(i=2;i<=3;i++)
  {
    mv[i] = v[i] = mvc;
    o[i] = pvc;
    u[i] = presentsv/100;
    en[i] = 0.0;
  }
for(i=4;i<=9;i++)
  {
    mv[i] = mvc;
    o[i] = pvc;
    en[i] = 0.0;
  }

inp = u[0] + step;
```

```
/*********** Find steady state of MV[] ************/

do {
    t = t + ts;
    count = count + 1;
    en[0] = u[0] - o[0];
    /******** Stop Cond. ********/
    if( count > 10 && t > 3*maxdelaytime )
      {
        sumx = sumy = sumxy = sumxx = 0.0;
        for( i=0;i<=9;i++)
          {
            x = t-(i*ts);
            y = o[i];
            sumx = sumx + x;
            sumy = sumy + y;
            sumxy = sumxy + x*y;
            sumxx = sumxx + pow(x,2);
          }
        slope = (num*sumxy - sumx*sumy)/(num*sumxx - pow(sumx,2.0));
        if( fabs(slope) < epps )
          {



            if ( fabs(en[0]) > eps )
              {
                if ( t >= 2*maxsteadytime )
                    status = 1;
              }
              else
                status = 1;
          }
          else
            if ( t > 3*maxsteadytime )
                status = 1;
      }
    /******** Controller ********/
    an[0] = kp*en[0];
    bn[0] = bn[1] + an[0]*ts/ti;
    cn[0] = (tf*cn[1] + kp*td*(o[0]-o[3]-3.0*o[2]+3.0*o[1])
    /(6.0*(tf+ts)));
    mv[0] = an[0]+bn[0]+cn[0]+bias;
    if(mv[0]>hilim)
        mv[0]=hilim;
    if(mv[0]<lowlim)
        mv[0]=lowlim;
    /********   Process   ********/
    v[0] = mv[0] + a[1]*mv[1] + a[2]*mv[2] + b[1]*v[1] + b[2]*v[2];
    o[0] = c[0]*v[0] + c[1]*v[1] + c[2]*v[2] + d[1]*o[1] + d[2]*o[2];
    /********  Iteration  ********/
    an[1] = an[0];
    bn[1] = bn[0];
    cn[1] = cn[0];
    for(i=8;i>=3;i--)
      {
        mv[i+1] = mv[i];
        o[i+1] = o[i];
        en[i+1] = en[i];
      }
    for(i=2;i>=0;i--)
      {
        mv[i+1] = mv[i];
        o[i+1] = o[i];
        en[i+1] = en[i];
        u[i+1] = u[i];
        v[i+1] = v[i];
      }
   } while (status==0);

/****************     Loop     **********************/

pvc = o[0];
mvc = mv[0];
t = 0;
count = 0;
status = 0;
do {
    t = t + ts;
    count = count + 1;
    u[0] = inp;
    en[0] = u[0] - o[0];
    /********    non shoot      ********/
    if ( en[0] < 0 )
        weight = 1000.0;
      else
        weight = 0.0;
    er = er + t*fabs(en[0]) + weight*fabs(en[0]);


    /******** Stop Cond. ********/
    if( t > maxdelaytime && count > 10 )
      {
        sumx = sumy = sumxy = sumxx = 0.0;
        for( i=0;i<=9;i++)
          {
            x = t-(i*ts);
            y = o[i];
            sumx = sumx + x;
            sumy = sumy + y;
            sumxy = sumxy + x*y;
            sumxx = sumxx + pow(x,2);
          }
        slope = (num*sumxy - sumx*sumy)/(num*sumxx - pow(sumx,2.0));
        if( fabs(slope) < epps )
          {
            if ( fabs(en[0]) > eps )
              {
                if ( t >= 2*maxsteadytime )
                     status = 1;
              }
              else
                status = 1;
          }
        else
           if ( t > 3*maxsteadytime )
                status = 1;
     }
   /******** Controller ********/
   an[0] = kp*en[0];
   bn[0] = bn[1] + an[0]*ts/ti;
   cn[0] = (tf*cn[1] + kp*td*(o[0]-o[3]-3.0*o[2]+3.0*o[1])
           /(6.0*(tf+ts)));
   mv[0] = an[0]+bn[0]+cn[0]+bias;
   if(mv[0]>hilim)
       mv[0]=hilim;
   if(mv[0]<lowlim)
       mv[0]=lowlim;
   /********   Process  ********/
   v[0] = mv[0] + a[1]*mv[1] + a[2]*mv[2] + b[1]*v[1] + b[2]*v[2];
   o[0] = c[0]*v[0] + c[1]*v[1] + c[2]*v[2] + d[1]*o[1] + d[2]*o[2];
   /********  Iteration  ********/
   an[1] = an[0];
   bn[1] = bn[0];
   cn[1] = cn[0];
   for(i=8;i>=3;i--)
     {
       mv[i+1] = mv[i];
       o[i+1] = o[i];
       en[i+1] = en[i];
     }
    for(i=2;i>=0;i--)
     {
       mv[i+1] = mv[i];
       o[i+1] = o[i];
       en[i+1] = en[i];
       u[i+1] = u[i];
       v[i+1] = v[i];
     }
   } while (status==0);
er = er*ts;
return(er);
}



clr(void)

{
```

```
_AH = 6;
_AL = 25;
_CH = 0;
_CL = 0;
_DH = 24;
_DL = 79;
_BH = 7;
geninterrupt(0x10);
gotorc(0,0);

}

int gotorc ( int row, int col)

{

_AH = 2;
_BH = 0;
_DH = row;
_DL = col;
geninterrupt(0x10);

}

writemem(int page,int row,int col,int attribute,char *string)

{

register    int offset = (page<<12) + col + col + row*160;

     while(*string!='\0')
     {
     pokeb(monitor, offset++, *string++);
     pokeb(monitor, offset++, (char)attribute);
     }

}

writeva(int page, int row, int col, int attrib, char *msg, ...)

{

char      buf[256];
va_list   ap;
va_start  (ap,msg);
vsprintf  (buf,msg,ap);
writemem(page,row,col,attrib,buf);

}

int cattrib(int page,int row,int col,int lenght,int newattrib)

{
```

```
register    int offset=(page<<12)+col+col+row*160;
register    int i;

offset++;



for(i=0;i<lenght;i++)
  {
    pokeb(monitor,offset,(char)newattrib);
    offset+=2;
  }

}

int getkey()

{

register    int c;

_AH =0;
geninterrupt(0x16);
c = _AX & 0x00FF;
if(c==0)  c=0x100+(_AX>>8);
return c;

}

movup(int rowb,int rowe,int colb,int cole,int line,int attrib)

{

_AH = 6;
_AL = line;
_CH = rowb;
_CL = colb;
_DH = rowe;
_DL = cole;
_BH = attrib;
geninterrupt(0x10);
gotorc(rowe-line+1,colb);

}

movdn(int rowb,int rowe,int colb,int cole,int line,int attrib)

{

_AH = 6;
_AL = line;
_CH = rowb;
_CL = colb;
_DH = rowe;
_DL = cole;
```

```
_BH = attrib;
geninterrupt(0x10);
gotorc(rowe-line+1,colb);

}

int showcur(void)

{

_AH = 1;
_CH = 12;
_CL = 13;
geninterrupt(0x10);




}

int hidecur(void)

{

_AH = 1;
_CH = 14;
_CL = 14;
geninterrupt(0x10);

}

int     selectmenu(int startrow,int startcol,int lenght,int step,
        int number)

{

int     index,p;
int     row,col;

index = 1;
row = startrow;
col = startcol;
cattrib(0,row,col,lenght,Reverse);
do {
    p = getkey();
    writemem(0,20,49,Normal,"                      ");
    writemem(0,21,49,Normal,"                      ");
    switch(p)
      {
        case HOME_KEY:
        case PGUP_KEY:  {
                index = 1;
                cattrib(0,row,col,lenght,Highlight);
                row = startrow;
                cattrib(0,row,col,lenght,Reverse);
                break;
                }
        case END_KEY:
        case PGDN_KEY:  {
                index = number;
                cattrib(0,row,col,lenght,Highlight);
                row = startrow + (number-1)*step;
                cattrib(0,row,col,lenght,Reverse);
                break;
                }
        case DOWN_KEY:  {
                cattrib(0,row,col,lenght,Highlight);
                index++;
                row = row + step;
                if( index > number )
                  {
                    index = 1;
                    row = startrow;
                  }
                cattrib(0,row,col,lenght,Reverse);
                break;
                }
        case UP_KEY:    {
                cattrib(0,row,col,lenght,Highlight);
                index--;
                row = row - step;


                if(index < 1)
                  {
                    index = number;
                    row = startrow + (number-1)*step;
                  }
                cattrib(0,row,col,lenght,Reverse);
                break;
                }
        case '\r':
        case '\n':
                {
                row = 1000;
                break;
                }
        default  :
                {
                writemem(0,20,49,Reverse,"   Keypress Error !   ");
                writemem(0,21,49,ReverseBlink,"     TRY  AGAIN !  ");
                break;
                }
      }
} while (row != 1000);
return(index);

}
```

```
void    logo(void)

{

int          i;

/******************      LOGO      ******************/

clr();
cattrib(0,0,0,80,Reverse);
cattrib(0,1,0,80,Reverse);
cattrib(0,2,0,80,Reverse);
for(i=3;i<=21;i++)
  {
    cattrib(0,i,0,16,Reverse);
    cattrib(0,i,64,16,Reverse);
   }
cattrib(0,22,0,80,Reverse);
cattrib(0,23,0,80,Reverse);
cattrib(0,24,0,80,Reverse);
writemem(0,6,24,Highlight,"Optimum Tuning of PID Controller");
writemem(0,9,39,Normal,"By");
writemem(0,11,31,Normal,"Pisanu  Kijpaitulaya");
writemem(0,13,30,Normal,"Master Degree Student");
writemem(0,14,28,Normal,"Chulalongkorn University");
writemem(0,17,20,Highlight,"Adviser :- ");
writemem(0,19,28,Highlight,"Dr. Somboon  Chongchaikij");
delay(5000);

}

void            drawbox(void)

{

int          i;


clrscr();
writemem(0,0,0,Normal,"ฬ");
for(i=1;i<=78;i++)
    writemem(0,0,i,Normal,"ภ");
writemem(0,0,79,Normal,"ฝ");
writemem(0,1,0,Normal,"ผ");
writemem(0,1,79,Normal,"ฝ");
writemem(0,2,0,Normal,"ส");
writemem(0,2,79,Normal,"ซ");
for(i=1;i<=78;i++)
    writemem(0,2,i,Normal,"ว");
for(i=3;i<=15;i++)
  {
    writemem(0,i,0,Normal,"ผ");
    writemem(0,i,79,Normal,"ผ");
  }
writemem(0,16,0,Normal,"ะ");
for(i=1;i<=78;i++)
    writemem(0,16,i,Normal,"ภ");
writemem(0,16,39,Normal,"ฮ");
writemem(0,16,79,Normal,"ป");
writemem(0,17,0,Normal,"ผ");
writemem(0,17,39,Normal,"ผ");
writemem(0,17,79,Normal,"ผ");
writemem(0,18,0,Normal,"ส");
writemem(0,18,79,Normal,"ธ");
for(i=1;i<=78;i++)
    writemem(0,18,i,Normal,"ว");
writemem(0,18,39,Normal,"ฯ");
for(i=19;i<=23;i++)
  {
    writemem(0,i,0,Normal,"ผ");
    writemem(0,i,39,Normal,"ผ");
    writemem(0,i,79,Normal,"ผ");
  }
writemem(0,24,0,Normal,"ห");
for(i=1;i<=78;i++)
    writemem(0,24,i,Normal,"ภ");
writemem(0,24,79,Normal,"ฟ");
writemem(0,24,39,Normal,"อ");
writemem(0,1,37,Highlight,"INPUT");
writemem(0,17,15,Highlight,"MESSAGE");
writemem(0,17,52,Highlight,"ERROR MESSAGE");

}
steady.asm

              dosseg

              .model small

              extrn      _slopetes:PROC   ;

              .data

              count      db    0bh        ;
              extrn      _otimes:word     ;
              extrn      _otimem:word     ;
              extrn      _ntimes:word     ;
              extrn      _ntimem:word     ;
              extrn      _chanal:word     ;
              extrn      _point:word      ;
              extrn      _number:word     ;



              extrn      _num:word        ;
              extrn      _interval:word   ;
              extrn      _errorcode:word  ;
              extrn      _opt:word        ;
```

```
                .code

                PUBLIC    _steady

_steady         proc  near
                push  bp
                push  si
                push  di
                mov   _errorcode,00h
                mov   ah,2ch
                int   21h
                mov   cx,0000h
                mov   cl,dl
                mov   _otimem,cx
                mov   cl,dh
                mov   _otimes,cx
_begin      :   mov   ah,2ch
                int   21h
                mov   cx,0000h
                mov   cl,dl
                mov   _ntimem,cx
                mov   cl,dh
                mov   _ntimes,cx
                cmp   _otimes,cx
                jz    _nadjust
_adjust     :   mov   ax,_otimem
                mov   bl,64h
                sub   bl,al
                mov   ax,_ntimem
                add   al,bl
                cmp   ax,_interval
                jae   _atd
                jmp   _begin
_nadjust    :   mov   bx,_ntimem
                sub   bx,_otimem
                cmp   bx,_interval
                jb    _begin
_atd        :   mov   ax,_ntimes
                mov   _otimes,ax
                mov   ax,_ntimem
                mov   _otimem,ax
                add   _number,02h
                mov   di,_number
                cmp   di,_num
                jne   _inp
                mov   _errorcode,0ffffh
                pop   di
                pop   si
                pop   bp
                ret
_inp        :   mov   count,0bh
                mov   dx,0388h
                mov   ax,_chanal
                out   dx,ax
```

```
                   mov   ax,1800h
                   push  ax
                   mov   bx,0800h
_agn          :    mov   dx,0380h


                   mov   al,bl
                   out   dx,al
                   inc   dl
                   mov   al,bh
                   out   dx,al
                   mov   cx,0010h
_wt1          :    loop  _wt1
                   mov   dx,0389h
                   in    al,dx
                   and   al,01h
                   jnz   _anx
                   pop   ax
                   sar   ax,1
                   or    bx,ax
                   push  ax
                   cmp   count,0
                   je    _ext
                   dec   count
                   jmp   _agn
_anx          :    pop   ax
                   sar   ax,1
                   xor   bx,ax
                   push  ax
                   cmp   count,0
                   je    _ext
                   dec   count
                   jmp   _agn
_ext          :    pop   ax
                   mov   ax,bx
                   mov   bx,_point
                   mov   [bx+di],ax
                   call  _slopetes
                   cmp   _opt,0
                   jnz   _last
                   jmp   _begin
_last         :    pop   di
                   pop   si
                   pop   bp
                   ret
_steady            endp
                   end
```

testmag.asm

```
                   dosseg

                   .model small
                   extrn      _slopetes:PROC    ;

                   .data

                   count      db     0bh          ;
                   cond       dw     0000h        ;
                   extrn      _otimes:word        ;
                   extrn      _otimem:word        ;
                   extrn      _ntimes:word        ;
                   extrn      _ntimem:word        ;
                   extrn      _chanal:word        ;
                   extrn      _point:word         ;
                   extrn      _number:word        ;
                   extrn      _num:word           ;
                   extrn      _interval:word      ;



                   extrn      _errorcode:word     ;
                   extrn      _opt:word           ;

                   .code

                   PUBLIC     _testmag

_testmag           proc    near
                   push    bp
                   push    si
                   push    di
                   mov     bp,sp
                   mov     dx,[bp+14]
                   mov     cond,dx
                   mov     dx,[bp+8]
                   xor     dh,dh
                   shl     dl,1
                   add     dx,0380h
                   mov     ax,[bp+10]
                   out     dx,al
                   inc     dx
                   mov     al,ah
                   out     dx,al
                   mov     _errorcode,00h
                   mov     ah,2ch
                   int     21h
                   mov     cx,0000h
                   mov     cl,dl
                   mov     _otimem,cx
                   mov     cl,dh
                   mov     _otimes,cx
_begin          :  mov     ah,2ch
                   int     21h
                   mov     cx,0000h
                   mov     cl,dl
                   mov     _ntimem,cx
```

```
                mov   cl,dh
                mov   _ntimes,cx
                cmp   _otimes,cx
                jz    _nadjust
_adjust      :  mov   ax,_otimem
                mov   bl,64h
                sub   bl,al
                mov   ax,_ntimem
                add   al,bl
                cmp   ax,_interval
                jae   _atd
                jmp   _begin
_nadjust     :  mov   bx,_ntimem
                sub   bx,_otimem
                cmp   bx,_interval
                jb    _begin
_atd         :  mov   ax,_ntimes
                mov   _otimes,ax
                mov   ax,_ntimem
                mov   _otimem,ax
                add   _number,02h
                mov   di,_number
                cmp   di,_num
                jne   _inp
                mov   _errorcode,0ffffh
                pop   di
                pop   si


                pop   bp
                ret
_inp         :  mov   count,0bh
                mov   dx,0388h
                mov   ax,_chanal
                out   dx,ax
                mov   ax,1800h
                push  ax
                mov   bx,0800h
_agn         :  mov   dx,0380h
                mov   al,bl
                out   dx,al
                inc   dl
                mov   al,bh
                out   dx,al
                mov   cx,0010h
_wt1         :  loop  _wt1
                mov   dx,0389h
                in    al,dx
                and   al,01h
                jnz   _anx
                pop   ax
                sar   ax,1
                or    bx,ax
                push  ax
```

```
                    cmp   count,0
                    je    _ext
                    dec   count
                    jmp   _agn
_anx           :    pop   ax
                    sar   ax,1
                    xor   bx,ax
                    push  ax
                    cmp   count,0
                    je    _ext
                    dec   count
                    jmp   _agn
_ext           :    pop   ax
                    mov   ax,bx
                    mov   bx,_point
                    mov   [bx+di],ax
                    cmp   ax,cond
                    jae   _last
                  mov   cx,[bp+16]
                    cmp   cx,_number
                    jb    _maxtime
                    jmp   _begin
_last          :    mov   _opt,1
_maxtime       :    mov    dx,[bp+8]
                    xor    dh,dh
                    shl    dl,1
                    add    dx,0380h
                    mov    ax,[bp+12]
                    out    dx,al
                    inc    dx
                    mov    al,ah
                    out    dx,al
                    pop   di
                    pop   si
                    pop   bp
                    ret
_testmag            endp
                    end
```

keepdata.asm

```
                    dosseg

                    .model small

                    extrn     _slopetes:PROC    ;

                    .data

                    count     db    0bh         ;
```

```
                time       dw    0000h        ;
                cond       dw    0000h        ;
                extrn      _otimes:word       ;
                extrn      _otimem:word       ;
                extrn      _ntimes:word       ;
                extrn      _ntimem:word       ;
                extrn      _chanal:word       ;
                extrn      _point:word        ;
                extrn      _number:word       ;
                extrn      _num:word          ;
                extrn      _interval:word     ;
                extrn      _errorcode:word    ;
                extrn      _opt:word          ;

                .code

                PUBLIC     _keepdata

_keepdata       proc    near
                push    bp
                push    si
                push    di
                mov     bp,sp
                mov     dx,[bp+8]
                xor     dh,dh
                shl     dl,1
                add     dx,0380h
                mov     ax,[bp+10]
                out     dx,al
                inc     dx
                mov     al,ah
                out     dx,al
                mov    _errorcode,00h
                mov    ah,2ch
                int    21h
                mov    cx,0000h
                mov    cl,dl
                mov    _otimem,cx
                mov    cl,dh
                mov    _otimes,cx
_begin       :  mov    ah,2ch
                int    21h
                mov    cx,0000h
                mov    cl,dl
                mov    _ntimem,cx
                mov    cl,dh
                mov    _ntimes,cx
                cmp    _otimes,cx
                jz     _nadjust
_adjust      :  mov    ax,_otimem
                mov    bl,64h


                sub    bl,al
                mov   ax,_ntimem
                add   al,bl
                cmp   ax,_interval
                jae   _atd
                jmp   _begin
_nadjust    :   mov   bx,_ntimem
                sub   bx,_otimem
                cmp   bx,_interval
                jb    _begin
_atd        :   mov   ax,_ntimes
                mov   _otimes,ax
                mov   ax,_ntimem
                mov   _otimem,ax
                add   _number,02h
                mov   di,_number
                cmp   di,_num
                jne   _inp
                mov   _errorcode,0ffffh
                pop   di
                pop   si
                pop   bp
                ret
_inp        :   mov   count,0bh
                mov   dx,0388h
                mov   ax,_chanal
                out   dx,ax
                mov   ax,1800h
                push  ax
                mov   bx,0800h
_agn        :   mov   dx,0380h
                mov   al,bl
                out   dx,al
                inc   dl
                mov   al,bh
                out   dx,al
                mov   cx,0010h
_wt1        :   loop  _wt1
                mov   dx,0389h
                in    al,dx
                and   al,01h
                jnz   _anx
                pop   ax
                sar   ax,1
                or    bx,ax
                push  ax
                cmp   count,0
                je    _ext
                dec   count
                jmp   _agn
_anx        :   pop   ax
                sar   ax,1
                xor   bx,ax
                push  ax
```

```
                 cmp    count,0
                 je     _ext
                 dec    count
                 jmp    _agn
_ext         :   pop    ax
                 mov    ax,bx
                 mov    bx,_point
                 mov    [bx+di],ax
                 call   _slopetes


                 cmp    _opt,1
                 je     _last
                 jmp    _begin
_last        :   mov     dx,[bp+8]
                 xor     dh,dh
                 shl     dl,1
                 add     dx,0380h
                 mov     ax,[bp+12]
                 out     dx,al
                 inc     dx
                 mov     al,ah
                 out     dx,al
                 pop    di
                 pop    si
                 pop    bp
                 ret
_keepdata        endp
                 end

output.asm

                 dosseg
                 .model small

                 .data

                 .code

                 PUBLIC _output

_output          proc   near
                 push   bp
                 push   si
                 push   di
                 mov    bp,sp
                 mov    dx,[bp+8]
                 xor    dh,dh
                 shl    dl,1
                 add    dx,0380h
                 mov    ax,[bp+10]
                 out    dx,al
                 inc    dx
```

```
                mov     al,ah
                out     dx,al
                pop     di
                pop     si
                pop     bp
                ret
_output         endp
                end
```

ประวัติผู้เขียน

นายพิษณุ กิจไพฑูรย์ เกิดเมื่อวันที่ 1 ตุลาคม พ.ศ. 2507 ที่กรุงเทพมหานคร สำเร็จการศึกษาขั้นปริญญาบัณฑิต ในสาขาวิศวกรรมไฟฟ้า จากคณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น เมื่อปีการศึกษา 2529.