รายการอ้างอิง

ภาษาไทย

บุญมี. อย่างธารา. ดร. เอกสารประกอบการเรียนวิชา 162-601 Numerical Method and Digital Computation. 2535.

ภาษาอังกฤษ

DeCarlo , R. A. , Zak . S. H. , and Mathews , G. P. "Variable structure control of nonlinear multivariable systems: A Tutorial." Proc. IEEE Vol. 76 (1988) :212-232.

Drakunov , S. I. and Utkin , V. I. "On discrete-time sliding modes." IFAC Nonlinear Control Systems Design (1989) : 273-278.

Furuta , K. "Sliding mode control of discrete system." System & Control Letters 14 (1990) : 145-152.

Hung , J. Y. , Gao , W. , and Hung , J. C. "Variable structure control:A survey." IEEE Transaction on Industrial Electronics. Vol. 40 (February 1993) : 2-22.

Kotta , U. "Comment on "On the Stability of Discrete-Time Sliding Mode Control Systems"" IEEE Transcation on Automatic Control. Vol. 34 (September 1989) : 1021-1022.

Milosavljevic , C. "General condition for the existence of a quasisliding mode on the switching hyperplane in discrete variable structure systems." Automat. Remote Contr. Vol. 46 (1985) : 307-314.

Ogata , K. "Discrete-time control systems." Prentice-Hall International , 1987. pp.994.

Pieper , J. K. , Surgenor , B. W. "Discrete sliding control of a coupled-drives apparatus with optimal sliding surface and switching gain" IEE Proc. D. Vol. 140 (March 1993) : 70-78

Sarpturk , S. Z. , Istefanopulos , Y. , and Kaynak , O. "On the stability of discrete-time sliding mode control systems." IEEE Transaction on Automatic Control. Vol. AC-32 (October 1987) : 930-932.

Utkin , V. I. "Variable structure systems with sliding modes." IEEE Transaction on Automatic Control Vol. AC-22 (April 1977) : 212-222.

White , B. A. "Reduced-order switching functions in variable-structure control systems." IEE Proc. Pt. D Vol. 130 (March 1983) : 33-40.

_____. , and Silson , P. M. "Reachability in variable structure control systems." IEE Proc. Pt. D Vol. 131 (May 1984) : 85-91.

Yu , X. "Discrete variable structure control systems." INT. J. SYSTEMS SCI. Vol 13 (1993) : 373-386.

_____. , and Potts , R. B. "Analysis of discrete variable structure systems with pseudo-sliding modes." INT. J. SYSTEMS SCI. Vol. 23 (1992) :503-516.

ภาคผนวก ก

ทฤษฎีพื้นฐานของระบบควบคุมไม่ต่อเนื่องเชิงเวลา

ในภาคผนวกบทนี้จะกล่าวถึงการทำการแปลงระบบในโดเมนเวลาต่อเนื่องเป็นโดเมนเวลาไม่ต่อเนื่อง เมื่อแบบจำลองทางคณิตศาสตร์ของระบบในโดเมนเวลาต่อเนื่องเขียนอยู่ในรูปของปริภูมิสเตทดังสมการ

$$\dot{\mathbf{x}} = \mathbf{A}.\mathbf{x} + \mathbf{B}.\mathbf{u} \qquad \text{(ก-1)}$$

เป็นที่ทราบกันโดยทั่วไปว่าผลเฉลยของระบบดังสมการ (ก-1) อยู่ในรูปของ

$$\mathbf{x}(t) = e^{\mathbf{A}.t}.\mathbf{x}(0) + e^{\mathbf{A}.t}.\int_0^t e^{-\mathbf{A}.\tau}.\mathbf{B}.\mathbf{u}(\tau).d\tau \qquad \text{(ก-2)}$$

และเนื่องจากระบบควบคุมที่พิจารณาในวิทยานิพนธ์นี้ใช้การชักตัวอย่างและการคงค่าอันดับศูนย์ ดังนั้นค่าสัญญาณควบคุมจะคงที่ตลอดในระหว่างช่วงเวลาชักตัวอย่างดังสมการ

$$\mathbf{u}(t)=\mathbf{u}(k.T) \qquad ; k.T \le t < k.T+T \qquad \text{(ก-3)}$$

และจากสมการที่ (ก-2) ได้ว่า

$$\mathbf{x}((k+1).T) = e^{\mathbf{A}.((k+1).T)}.\mathbf{x}(0) + e^{\mathbf{A}.((k+1).T)}.\int_0^{(k+1).T} e^{-\mathbf{A}.\tau}.\mathbf{B}.\mathbf{u}(\tau).d\tau \qquad \text{(ก-4)}$$

และ

$$\mathbf{x}(k.T) = e^{\mathbf{A}.k.T}.\mathbf{x}(0) + e^{\mathbf{A}.k.T}.\int_0^{k.T} e^{-\mathbf{A}.\tau}.\mathbf{B}.\mathbf{u}(\tau).d\tau \qquad \text{(ก-5)}$$

คูณสมการที่ (ก-5) ด้วย $e^{\mathbf{A}.T}$ และนำไปลบออกจากสมการที่ (ก-4) ได้ว่า

$$\mathbf{x}((k+1).T) = e^{\mathbf{A}.T}.\mathbf{x}(k.T) + e^{\mathbf{A}.((k+1).T)}.\int_{k.T}^{(k+1).T} e^{-\mathbf{A}.\tau}.\mathbf{B}.\mathbf{u}(\tau).d\tau \qquad \text{(ก-6)}$$

จากสมการที่ (ก-3) และ (ก-6) ได้ว่า

$$\mathbf{x}((k+1).T) = e^{\mathbf{A}.T}.\mathbf{x}(k.T) + e^{\mathbf{A}.T}.\int_0^{T} e^{-\mathbf{A}.t}.\mathbf{B}.\mathbf{u}(k.T).d\,t \qquad \text{(ก-7)}$$

เปลี่ยนตัวแปรด้วยสมการ λ=T–t

$$\mathbf{x}((k+1).T) = e^{\mathbf{A}.T}.\mathbf{x}(k.T) + \int_0^T e^{\mathbf{A}.\lambda}.\mathbf{B}.\mathbf{u}(k.T).d\,\lambda \qquad (ก\text{-}8)$$

กำหนดให้

$$\mathbf{G}(T) = e^{\mathbf{A}.T} \qquad (ก\text{-}9)$$

และ

$$\mathbf{H}(T) = (\int_0^T e^{\mathbf{A}.\lambda} d\,\lambda).\mathbf{B} \qquad (ก\text{-}10)$$

เขียนสมการที่ (ก-8) ได้ว่า

$$\mathbf{x}((k+1).T)=\mathbf{G}(T).\mathbf{x}(k.T)+\mathbf{H}(T).\mathbf{u}(k.T) \qquad (ก\text{-}11)$$

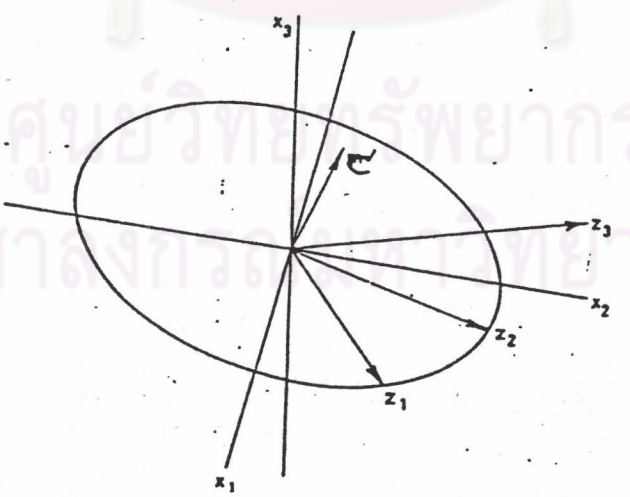
สังเกตว่าเมตริกซ์ **G** และเมตริกซ์ **H** ขึ้นกับเวลาชักตัวอย่างและเมตริกซ์ที่แสดงสมบัติของระบบในโดเมนเวลาต่อเนื่องประกอบกัน

ภาคผนวก ข

การหาสมการพลวัตของสวิตชิงฟังก์ชัน

พิจารณาระบบดังสมการที่ (2-8) ถึงสมการที่ (2-14) ในบทที่ 2 เป็นที่ทราบว่าผู้ออกแบบสามารถออกแบบระบบให้มีค่าเจาะจงได้ n-1 ค่าบนพื้นผิวสวิตชิง จากค่าเจาะจงของระบบทั้งหมด n ค่า โดยที่เวกเตอร์เจาะจง n-1 เวกเตอร์ที่สัมพันธ์กับค่าเจาะจงทั้ง n-1 ค่านั้นจะอยู่บนปริภูมิที่มีมิติเท่ากับ n-1 ซึ่งเรียกว่าปริภูมิว่าง และเวกเตอร์เจาะจงตัวที่ n (λ_n) จะเป็นเวกเตอร์ที่เกี่ยวข้องกับพลวัตของสวิตชิงฟังก์ชัน

สมมุติให้ระบบมีมิติเท่ากับสาม เวกเตอร์เจาะจงที่หนึ่งและสองที่อยู่บนปริภูมิว่างแสดงด้วย z_1 และ z_2 ส่วนเวกเตอร์เจาะจงตัวที่สามที่สัมพันธ์กับค่าเจาะจง λ_n หรือ λ_3 ในกรณีนี้ แสดงด้วย z_3 ดังรูปที่ ข-1



รูปที่ ข-1 แสดงปริภูมิว่างและเวกเตอร์เจาะจงของระบบที่มีมิติเท่ากับสาม จากรูปเห็นได้ว่าเวกเตอร์ c จะตั้งฉากกับปริภูมิว่างด้วย

พิจารณากรณีที่ $\Delta k = 0$ ก่อน สมมุติว่าค่าเจาะจงที่ 1 ถึง n-1 มีค่าไม่เท่ากันและจะเป็นจำนวนเชิงซ้อนหรือไม่ก็ได้ แต่ค่าเจาะจงที่ n (λ_n) ต้องเป็นจำนวนจริงเท่านั้น

ดังนั้นเวกเตอร์ $\mathbf{x}$ ใดๆบนปริภูมิว่างสามารถเขียนได้เป็น

$$\mathbf{x} = \sum_{i=1}^{n-1} \beta_i . \mathbf{z}_i \qquad \text{(ข-1)}$$

เมื่อ β_i เป็นค่าคงที่

จากสมการที่ (2-13) ถ้าเวกเตอร์ $\mathbf{x}$ อยู่บนปริภูมิว่าง ได้ว่า

$$s = \mathbf{c}'.\mathbf{x} = 0$$

แทนค่าเวกเตอร์ $\mathbf{x}$ จากสมการที่ (ข-1)

$$s = \sum_{i=1}^{n-1} \beta_i . \mathbf{c}'.\mathbf{z}_i \qquad \text{(ข-2)}$$

และถ้าเวกเตอร์ $\mathbf{x}$ ไม่ได้อยู่บนปริภูมิว่าง เขียนค่าของสวิตชิงฟังก์ชันได้ว่า

$$s = \beta_n . \mathbf{c}'.\mathbf{z}_n + \sum_{i=1}^{n-1} \beta_i . \mathbf{c}'.\mathbf{z}_i \qquad \text{(ข-3)}$$

$$s = \beta_n . \mathbf{c}'.\mathbf{z}_n \qquad \text{(ข-4)}$$

เขียนอนุพันธ์ของสมการที่ (ข-4) ได้ว่า

$$\begin{aligned} \dot{s} &= \beta_n . \mathbf{c}'.\dot{\mathbf{z}}_n \\ &= \lambda_n . \beta_n . \mathbf{c}'.\mathbf{z}_n \end{aligned} \qquad \text{(ข-5)}$$

หรือ

$$\dot{s} = \lambda_n . s \qquad \text{(ข-6)}$$

จากสมการที่ (ข-6) แสดงให้เห็นว่าถ้าค่าเจาะจงค่าที่ n (λ_n) มีเสถียรภาพและมีขนาดใหญ่เมื่อเทียบกับค่าเจาะจงที่สัมพันธ์กับปริภูมิว่าง ค่าของสวิตชิงฟังก์ชันจะลดลงแบบฟังก์ชันเอกซ์โพเนนเชียล แต่ถ้าค่าเจาะจง λ_n ไม่มีเสถียรภาพหรือมีขนาดเล็ก ค่าอัตราขยาย $\Delta \mathbf{k}$ ที่สัมพันธ์กับสัญญาณควบคุมแบบสวิตชิงจะถูกนำมาใช้เพื่อทำให้ปริภูมิว่างนี้มีเสถียรภาพหรือมีเสถียรภาพมากขึ้น

ดังนั้นจากสมการที่ (2-8) (2-12) และ (2-14)

$$\dot{\mathbf{x}} = \mathbf{A}_c . \mathbf{x} - \mathbf{B}. \Delta \mathbf{k}' . \mathbf{x}$$

และจากอนุพันธ์ของสมการที่ (2-13)

$$\begin{aligned} \dot{s} &= \mathbf{c}'.\dot{\mathbf{x}} \\ &= \mathbf{c}'.\mathbf{A}_c . \mathbf{x} - \mathbf{c}'.\mathbf{B}. \Delta \mathbf{k}' . \mathbf{x} \end{aligned} \qquad \text{(ข-7)}$$

จากสมการนี้แสดงให้เห็นว่าเมื่อป้อนกลับระบบด้วยสัญญาณควบคุมแบบสวิตชิง พลวัตของปริภูมิว่างจะกลายเป็นพลวัตของสวิตชิงฟังก์ชัน ดังนั้นจึงสามารถแทนพลวัตของปริภูมิว่างในสมการของพลวัตสวิตชิงฟังก์ชันได้ จากสมการที่ (ข-6) และ (ข-7) ได้ว่า

$$\dot{s} = \lambda_{n}.s - c'.B.\Delta k'.x$$

ซึ่งคือสมการที่ (2-15) ในบทที่สองที่แสดงพลวัตของสวิตชิงฟังก์ชันนั่นเอง

ภาคผนวก ค

โปรแกรมคอมพิวเตอร์ที่ใช้จำลองแบบระบบ

```
/*                    VSS26.C                    */
/* This file is the simulation file in Variable Structure Control.    */
/* This file is used with Linear System in form diff(X) = AX+BU.      */
/* By Input parameter of A,B,initial condition of X from other file   */
/* in form of array of matrix which has m*n+2 components              */
/* the first component store m (number of row)                        */
/* the second component store n (number of column)                    */
/* the third contain the 1,1 component of Matrix.                     */
/* We can access each data at i,j element by the formula which is     */
/* (i-1)*n+(j-1)+2 (define in the ij function)                        */
```



```
/*The objective of vss26.c in this subdirectory is for testing a hypothesis */ /*about a
relation between sampling time and sigma function.                 */

#include<stdlib.h>
#include<stdio.h>
#include<math.h>
```

```
#include<time.h>        /* Randomize function */
#include<string.h>
#define h 0.000001    /* Resolution of Integration */
double T = 0.005;        /* Sampling Interval change because this is variable T */
/* T/h must be integer */

char *name_N_p = "e_AT_0.005_p";   /* Name of constant matrix N(T) File */
char *name_N_n = "e_AT_0.005_n";



double *A=NULL  ;  /* Array for Matrix A */
double *B=NULL  ;  /* Array for Matrix B */
double *U=NULL  ;  /* Array for Initial of U */
double *S=NULL  ;  /* Array for matrix sliding plan constant */
double *K=NULL  ;  /* Array for constant feedback gain */
double *VK=NULL ;  /* Array for variable feedback gain */
double *N_p=NULL;  /* Array for N(T) matrix (positive) */
double *N_n=NULL;  /* Array for N(T) matrix (negative) */
double *CX=NULL;   /* store Xn in RK4 function for constant control input */

double re_sizeofS;    /* store reciprocal of size of S vector used in boundary plot */
double ps;          /* Global variable for store sigma value for plot */
double pn;          /* Global variable for store noise value for plot */
double pb;          /* Global variable for store boundary value for plot */
char   dc;          /* Global variable for store dummy character */
double EPS=0.000000000001;    /* use in function RK4 */
/* This value is a boundary of numerical error around a value 0 of switching plane */
```

```
int kp;  /* dummy variable for recieve keypressed */
FILE *InputDatafile;          /* For Input Data File from Disk */
FILE *WriteData;              /* For Write Data into File in Disk */

void Write_file(double *D,char nf[]);
void Write_file_plot(double *D1,double *D2,char nf[]);
/* This function make the output file for plotting graph suitable for GP.EXE */
void Write_file_mat_3(double *D1,double *D2,double *D3,char nf[]);
/* This function make the output file for plotting out by mathematica program*/
/* in three dimensions system or states */
double *Retrieve_write_file(char nf[]);
int ij(double *M,int i,int j);  /* For go to i,j component of Matrix M */
double *InputMatrix_File(char nf[]);
/* For Input Matrix from file which is in name and path in nf[31] */
void outputmatrix(double *M);   /* For display result */
void makezero(double *M);       /* Make all of matrix element = 0.0 */
void makeone(double *M);        /* Make all of matrix element = 1.0 */
void makeminus(double *M);      /* Make all of matrix element = -previous value */

double *transpose(double *M);
double *mul_con_matrix(double k,double *M);
double *addmatrix(double *M1,double *M2);
double *submatrix(double *M1,double *M2);
double *multiplication(double *M1,double *M2);
void copymatrix(double *S,double *D);  /* S source ; D destination */
double *hfvalue(double *X,double t);
void RK4(double *Xn,double t,double tf);
```

```
int main(void)
{
double *X=NULL     ; /* Array for Matrix Initial of X */
double *PX=NULL ;  /* array for plot data in time axis */
double *PX1=NULL ;  /* array for plot data in y axis - the x1 state */
double *PX2=NULL;   /* array for plot data in y axis - the x2 state */
double *PX3=NULL;   /* array for plot data in y axis - the x3 state */
double *PX4=NULL;   /* array for plot data in y axis - the x4 state */
double *PPS=NULL;  /* array for plot sigma value using with GLobal variable *PS */
double *PPN=NULL;  /* array for plot noise */
double *PPB=NULL;  /* array for plotting boundary */

double t = 0.0;       /* Time variable */
double tf = 4.5 ;      /* Final Time for integration */
int plot_point = 640;  /* number of point to plot is resolution of plotting */

/* because of this function is for plot value  respect to time 640 maximum */
/* point in X (time) axis.                                                  */
double x_min = 0;
double x_max = tf;
int i=0;
double call_step=0; /* for each increment of step in call RK4 function */
double dummy1=0,dummy2=0;

time_t start_time,stop_time;  /* for calculate time consumed */
```

```
start_time = time(NULL);

N_p = InputMatrix_File(name_N_p);
N_n = InputMatrix_File(name_N_n);



A = InputMatrix_File("a_matrix");
B = InputMatrix_File("b_matrix");
X = InputMatrix_File("init_x");
U = InputMatrix_File("init_u");
S = InputMatrix_File("s_matrix");
K = InputMatrix_File("k_matrix");
VK = InputMatrix_File("init_vk");
CX = InputMatrix_File("init_x");

/*
outputmatrix(A);
outputmatrix(B);
outputmatrix(X);
outputmatrix(U);
*/

PX = (double *)calloc(plot_point+1,sizeof(double));
PX1 = (double *)calloc(plot_point+1,sizeof(double));
PX2 = (double *)calloc(plot_point+1,sizeof(double));
PX3 = (double *)calloc(plot_point+1,sizeof(double));
```

```
PX4 = (double *)calloc(plot_point+1,sizeof(double));
PPS = (double *)calloc(plot_point+1,sizeof(double));
PPN = (double *)calloc(plot_point+1,sizeof(double));
PPB = (double *)calloc(plot_point+1,sizeof(double));

/* Begin part of calculation of size of S */
dummy1 = *S;
dummy2 = *(S+1);
re_sizeofS= 0.0;
for (i=2;i<((int)dummy1+(int)dummy2+3);i++)
    {
    re_sizeofS = pow(*(S+i),2)+re_sizeofS;
    }
re_sizeofS = sqrt(re_sizeofS);
re_sizeofS = 1.0/re_sizeofS;
dummy1 = 0.0;
dummy2 = 0.0;
i=0;
/* End part of calculation of size of S */


if (PX && PX1 && PX2 && PX3 && PX4 && PPS && PPN && PPB)
{
*PX = plot_point;
*PX1 = plot_point;
*PX2 = plot_point;
```

```
*PX3 = plot_point;
*PX4 = plot_point;
*PPS = plot_point;
*PPN = plot_point;
*PPB = plot_point;

*(PX+1) = t;
*(PX1+1) = *(X+2); /* Initial Condition */
*(PX2+1) = *(X+3);
*(PX3+1) = *(X+4);
*(PX4+1) = *(X+5);
*(PPS+1) = 0.0;
*(PPN+1) = 0.0;
*(PPB+1) = 0.0;
 /* clrscr();*/
printf("\nStart at time %lf \n the initial value =
%lf,%lf,%lf,%lf\n",*(PX+1),*(PX1+1),*(PX2+1),*(PX3+1),*(PX4+1));
printf("Resolution of Integration = %.20lf\n",h);
printf("Sampling Time = %.20lf\n",T);
printf("T/h = %.20lf\n",(double)T/h);

call_step = (tf-t)/(double)plot_point;
printf("initial call_step = %.20lf\n",call_step);
if (call_step < h)
   {
   printf("\nh (step size) for integration is more than the time between point to plot so
program terminate");
   exit(1);
```

```
    }
    else
    {
    dummy1 = call_step/h;
    dummy2 = modf(dummy1,&call_step);
    printf("Intermediate call_step = %.20lf\n",call_step);
    if (dummy2>=(h*10))
       {
       call_step = call_step+1.0;
       }
   call_step = call_step*h;
    }
printf("final call_step = %.20lf\n",call_step);
printf("size of S is %.20lf\n",re_sizeofS);
/*
printf("A key for start simulation\n");
kp = getchar();
*/
for (i=2;i<plot_point+1;i++)
{
tf = t + call_step;
RK4(X,t,tf);
*(PX + i) = tf;      /* End of integration procedure time */
*(PX1 + i) = *(X + ij(X,1,1)); /* Now PX1,PX2,PX3 store the plotting of X1,X2,X3
respectively */
*(PX2 + i) = *(X + ij(X,2,1));
*(PX3 + i) = *(X + ij(X,3,1));
*(PX4 + i) = *(X + ij(X,4,1));
```

```
*(PPS + i) = ps;
*(PPN + i) = pn;
*(PPB + i) = pb;

printf("%d\t",i);
if (((fabs(pb)-fabs(ps))<0.0) && (pb != 0.0))
   {
   printf("\nOut of boundary at
t=%lf,[%lf,%lf,%lf],pb=%lf,ps=%lf\n",tf,*(PX1+i),*(PX2+i),*(PX3+i),pb,ps);
   }


/*
printf("\nat time %.20lf , the value = %lf,%lf,%lf i =
%d\n",*(PX+i),*(PX1+i),*(PX2+i),*(PX3+i),i);
printf("\n out of RK4\n");
outputmatrix(X);
*/
/*
printf("A Key to continue\n");
kp = getchar();
*/

t = tf;
}
Write_file(PX,"x_axis");
Write_file(PX1,"x1_axis");
Write_file(PX2,"x2_axis");
```

```
Write_file(PX3,"x3_axis");
Write_file(PX4,"x4_axis");
Write_file(PPS,"pps");
Write_file(PPN,"ppn");
Write_file(PPB,"ppb");
/*
Write_file_plot(PX,PX1,"xx1_axis");
*/
Write_file_mat_3(PX,PPB,PPS,"PX_PPB_PPS");
Write_file_mat_3(PX,PX1,PX2,"PX_PX1_PX2");
Write_file_mat_3(PX,PX3,PPS,"PX_PX3_PPS");

free(PX);
free(PX1);
free(PX2);
free(PX3);
free(PX4);
free(PPS);
free(PPN);
free(PPB);
free(N_p);
free(N_n);
}
else
{
 /* clrscr();*/
printf(" Allocation memory error ");
exit(1);
```

```
}


free(A);
free(B);
free(X);
free(U);
free(S);
free(K);
free(VK);
free(CX);
stop_time = time(NULL);
printf("\n The total time used in this program is %ld second ",stop_time - start_time);
return(0);
}    /* End of Main */



/*==========================================================
========*/
/*====================== Begin of Matrix and RK4
=======================*/
/*==========================================================
========*/

/*========================= RK4
================================= */
double *hfvalue(double *X,double t)
```

```
/* This function is used with the RK4 function which for integration */
/* This function must be modified when the structure of input is changed */
/* such as a constant input,a state feedback input or system definition */
/* such as a time varying case or non-linear case.                    */
/* For the constant input case with linear time invariant system this  */
/* function uses global variable A,B,U in the calculation.            */

{
double *R1=NULL,*R2=NULL; /* R1 and R2 are the dummy Matrix */
double *R3=NULL;     /* R3 is a Matrix R3 = AX+BU  */
double *R4=NULL;     /* R4 is a return value h*R3 */
R1 = multiplication(A,X);
R2 = multiplication(B,U);
R3 = addmatrix(R1,R2);
free(R1);
free(R2);
R4 = mul_con_matrix(h,R3);
free(R3);
return(R4);
}

double *mul_con_matrix(double k,double *M)
{
/* This function multiply Martix M my constant k. */
int    m=0,n=0,i=0;
double *RESULT=NULL;
m = (int)(*M);
n = (int)(*(M+1));
```

```
RESULT = (double *)calloc(m*n+2,sizeof(double));
if(RESULT)
{
*(RESULT) = m;
*(RESULT+1) = n;
for(i=2;i<(m*n+2);i++)
   {
   *(RESULT+i) = *(M+i) * k;
   }
return(RESULT);
}
else
{
 /* clrscr();*/
printf("\n\n\nNot enough memory\n");
exit(0);
return(RESULT);       /* not necessary but complier warning */
}
}

void RK4(double *Xn,double t,double tf)
{
/* This function returns result in *Xn. */
/* t is initial time and for each loop t will increase by step size */
/* h (step size) is defined in constant (Constant Global) */
/* T is sampling interval (Constant Global) */
double *k1=NULL,*k2=NULL,*k3=NULL,*k4=NULL;
double *D1=NULL,*D2=NULL;  /* Dummy Variable */
```

```
double dummyt; /* Dummy variable for t */
double *Xn1;   /* Store Xn for next step */

/* Additional variable for variable structure control */
int dummy1,dummy2,total;
double *DA=NULL,*DS=NULL,*SIGMA=NULL,*DK=NULL,*DB=NULL;
double ds; /* store some *(SIGMA) for ease of write */
int i,j; /* Counter */
/* End of additional variable for variable structure control */

/* Additional variable for discrete system */
static double TT;                    /* Static Variable is initial = 0 */
static int flag ;
/* flag is ued for telling program about discrete information */
/* That are when flag = 1 mean that the algorithm just pass the first time */
/* of new sampling's hold mechanism and in other times flag = 0          */
/* use Global variable *DX for holding value of X for feedback u = -kx */
/* End of Additional variable for discrete system */

/* Additional Variable for Changing Step Size */
/* Static Variable is initial = 0 */
static double pds;    /* this variable will be used in plotting boundary too */
        static int cross;
/* End of Additional Variable for Changing Step Size */

/* Additional variable for plotting boundary */
int dd1,dd2;
double *DUMMY_p = NULL;
```

```
double *DUMMY_n = NULL;
double dummy3=0,dummy4=0;
double *DUMMY_X1=NULL;
double *DUMMY_X2=NULL;
double *BOUND = NULL;
double *DIRECTION = NULL; /* Direction with perpendicular to sliding plane */
double *DIFF_X_p = NULL;
double *DIFF_X_n = NULL;
/* End of Additional variable for plotting boundary */




while ( (t+(h/10.0)) < tf )
/* h/10 is a value from experiment which is wanted by this function */
/* because in C have some error in calculation of t<tf - this is an important */
/* stopping criterion in function */
{
                                        /*
                                        printf("At time %lf\n",t);
                                        */
/* This is the portion of changing U */


dummy1 = (int)*(A);     /* Row of A */
dummy2 = (int)*(A+1);   /* Column of A */
total = (dummy1*dummy2)+2;
DA = (double *)calloc(total,sizeof(double));
```

```
*(DA) = dummy1;
*(DA+1) = dummy2;

dummy1 = (int)*(B);
dummy2 = (int)*(B+1);
total = (dummy1*dummy2)+2;
DB = (double *)calloc(total,sizeof(double));
*(DB) = dummy1;
*(DB+1) = dummy2;
if ( DA && DB)
   {}
   else
   {
    /* clrscr();*/
   printf("\n\n Not enough Memory in RK4 function Hit a key to exit");
    /* getch(); */
   exit(1);
   }
copymatrix(A,DA);          /* DA store original A */
copymatrix(B,DB);          /* DB store original B */


DS = transpose(S);
SIGMA = multiplication(DS,Xn);      /* SIGMA is the sliding eqution evaluating at
each previous x */
free(DS);
/* The value of Sigma is very important in the decision of sliding plane */
/* Be careful */
```

```
ds = *(SIGMA+2);
dummy1 = (int)*(Xn);
dummy2 = (int)*(Xn+1);
if ( dummy2 != 1)
   {
   printf("\n\nIt may have some error in program in RK4 function because column of
Xn != 1\n");
   printf("Hit a Key for exit");
   exit(1);
   }
free(SIGMA);



if ( (t + (h/10.0))-> TT )
{
dummy1 = 1;
for (i=1;i<dummy1+1;i++)
   for (j=1;j<dummy2+1;j++)
        {
        if ((ds * *(Xn+ij(Xn,i,j) ) < EPS) && (ds * *(Xn+ij(Xn,i,j) ) > (-EPS)))
          {
          *(VK+ij(VK,i,1)) = 0.0;
          }
          else if (ds * *(Xn+ij(Xn,i,j) ) > 0.0)
                {
                *(VK+ij(VK,i,1)) = 300.0;
                }
```

```
                else
                {
                *(VK+ij(VK,i,1)) = -300.0;
                }
        }


D1 = addmatrix(K,VK);      /* k+delta_k */
DK = transpose(D1);
free(D1);

D2 = multiplication(B,DK);      /* b*transpose(k+delta_k) */
free(DK);
DS = submatrix(A,D2);           /* new A when already feedback */
free(D2);
copymatrix(DS,A);               /* store in Global variable for another function*/

free(DS);
makezero(U);
copymatrix(Xn,CX); /* store for next integration step for discrete time system*/
TT = TT+T;
flag = 1;

/* Begin Portion of Changing Step Size */
/*
if ( ((pds*ps)<0.0)  &&  (cross == 0) )
  {
   cross = 1;
```

```
    }
if (cross == 1)
   {
   T = T/5.0;
   cross = 2;
   printf("\nNow The Step size is %lf\n",T);
   printf("T/h = %.20lf\n",(double)T/h);
   printf("      above value should be a integer number\n");
   }
if ( (cross == 2) && (tf > 0.5) )
   {
   T = T * 5.0/2.0;
   cross = 3;
   printf("\nNow The Step size is %lf\n",T);
   printf("T/h = %.20lf\n",(double)T/h);
   printf("      above value should be a integer number\n");
   }

pds = ps;   /* copy present value to prevoius value for next loop */

/* End Portion of Changing Step size */
}
else  if (flag == 1)
        {
        D1 = addmatrix(K,VK);
        DK = transpose(D1);
        free(D1);
        D2 = multiplication(DK,CX);
```

```
        free(DK);
        copymatrix(D2,U);
        makeminus(U);
        free(D2);
        flag = 0;
        }
        /* This last if havn't else */


/* BT1 */
pn = *(U+2);
/* ET1 */


k1 = hfvalue(Xn,t);
/* Begin calculate Xn+0.5k1 */
D1 = mul_con_matrix(0.5,k1);
D2 = addmatrix(Xn,D1);
dummyt = t+(0.5*h);
/* End calculate Xn+0.5k1 */
k2 = hfvalue(D2,dummyt);
free(D1);
free(D2);
/* Calculate Xn+0.5k2 */
D1 = mul_con_matrix(0.5,k2);
D2 = addmatrix(Xn,D1);
/* End caluculate Xn+0.5k2 */
k3 = hfvalue(D2,dummyt);
```

```
free(D1);
free(D2);
/* Begin calculate Xn+k3 */
D1 =  addmatrix(Xn,k3);
dummyt = t+h;
/* End calculate Xn+k3 */
k4 = hfvalue(D1,dummyt);
free(D1);
D1 = addmatrix(k1,k2); /* D1 = k1+k2 */
D2 = addmatrix(D1,k3); /* D2 = k1+k2+k3 */
free(D1);
D1 = addmatrix(D2,k4); /* D1 = k1+k2+k3+k4 */
free(D2);
D2 = addmatrix(D1,k2); /* D2 = k1+2*k2+k3+k4 */
free(D1);
D1 = addmatrix(D2,k3); /* D1 = k1+2*k2+2*k3+k4 */
free(D2);
D2 = mul_con_matrix((1.0/6.0),D1); /* D2 = 1/6(k1+2*k2+2*k3+k4) */
free(D1);
Xn1 = addmatrix(Xn,D2); /* D1 = Xn+1 = Xn + 1/6(k1+2*k2+2*k3+k4) */
free(D2);


DS = transpose(S);
SIGMA = multiplication(DS,Xn);      /* SIGMA is the sliding eqution evaluating at the
most new X for the reason of main program storing for plot */
free(DS);
pds = *(SIGMA+2);
```

```
free(SIGMA);

DS = transpose(S);
SIGMA = multiplication(DS,Xn1);      /* SIGMA is the sliding eqution evaluating at
the most new X for the reason of main program storing for plot */
free(DS);
ds = *(SIGMA+2);
ps = ds;            /* For plot Sigma value ; ps is global variable */
free(SIGMA);



/* Begin part of storing the value for plot boundary */
if ( pds*ds < 0.0 )
{
DUMMY_X1 = addmatrix(Xn,Xn1);
DUMMY_X2 = mul_con_matrix(0.5,DUMMY_X1);
free(DUMMY_X1);
DUMMY_p = multiplication(N_p,DUMMY_X2);
dd1 = (int)*(DUMMY_p);
dd2 = (int)*(DUMMY_p+1);
if ( (dd1 != 4) && (dd2 != 1) )
  {
  printf("The size of matrix in function RK4 (boundary section) is mismatch\n");
  printf("program terminate\n");
  exit(1);
  }
```

```
DUMMY_n = multiplication(N_n,DUMMY_X2);
dd1 = (int)*(DUMMY_n);              /* this declaration used in accumulation */
dd2 = (int)*(DUMMY_n+1);
if ( (dd1 != 4) && (dd2 != 1) )
   {
   printf("The size of matrix in function RK4 (boundary section) is mismatch\n");
   printf("program terminate\n");
   exit(1);
   }
DIFF_X_p = submatrix(DUMMY_p,Xn1);
DIFF_X_n = submatrix(DUMMY_n,Xn1);
dummy3=0;
dummy4=0;
for (i=1;i<(dd1+1) ;i++)
   {
   dummy3 = pow(*(DIFF_X_p+ij(DIFF_X_p,i,1)),2) + dummy3;
   dummy4 = pow(*(DIFF_X_n+ij(DIFF_X_n,i,1)),2) + dummy4;
   }
dummy3 = sqrt(dummy3);
dummy4 = sqrt(dummy4);
if (dummy3 > dummy4)
  {}
  else
  {
  dummy3 = dummy4;
  }
DIRECTION = mul_con_matrix(re_sizeofS,S);
BOUND = mul_con_matrix(dummy3,DIRECTION);
```

```
DUMMY_X1 = addmatrix(BOUND,DUMMY_X2);
DS = transpose(S);
SIGMA = multiplication(DS,DUMMY_X1);      /* SIGMA is the sliding eqution */
free(DS);
dummy4 = *(SIGMA+2);
pb = dummy4;          /* For plot Sigma value ; ps is global variable */
free(SIGMA);



free(DUMMY_p);
free(DUMMY_n);
free(DUMMY_X1);
free(DUMMY_X2);
free(DIFF_X_p);
free(DIFF_X_n);
free(BOUND);
free(DIRECTION);
}
/* End part of storing the value for plot boundary */



copymatrix(Xn1,Xn);              /* Define Xn for new loop */
free(Xn1);
t = t+h;

/*
```

```
printf("\nat time %.20lf , the value jjj = %d\n",t,jjj);
printf("\n inner of RK4\n");
outputmatrix(Xn);
*/
/*
printf("A key to continue\n");
kp = getchar();
*/



free(k1);
free(k2);
free(k3);
free(k4);

copymatrix(DA,A);
free(DA);
copymatrix(DB,B);
free(DB);

}   /* End of While Loop */
                                    /*
                                    printf("\nafter while loop\n");
                                    */
}   /* End of RK4 function */
```

```
void Write_file(double *D,char nf[])
/* This function is for store the data used for plot graph which is the */
/* result of simulation function in the form of array of double which   */
/* has the first data store the number of data stored in this array,so  */
/* the total number of this array is number of data+1 that relevant to  */
/* the form of array of data in function plot.                          */
/* When you want to retrieve the data use function "Retrieve_write_data"*/
{
int i,number;    /* i for counter , number is the number of data */
if ( (WriteData = fopen(nf,"w")) == NULL )
{
   /* clrscr();*/
  printf("Open File %s for write data ERROR\n",nf);
  exit(1);
  }
number = (int)*(D);
for (i=0;i<(number+1);i++)
   {
   fprintf(WriteData,"%.15lf\n",*(D+i));
   }
printf("\nSave data to file %s ready\n",nf);
fclose(WriteData);
}

void Write_file_plot(double *D1,double *D2,char nf[])
/* This function is for store the data used for plot graph which is the */
/* result of simulation function in the form of coordinate of double which   */
/* has the first data store the number of data stored in this array,so  */
```

```
/* the total number of this array is number of data+1  which can use for plot */
/* by program that include from Japan (GP.EXE). */

{
int i,number1,number2;    /* i for counter , number is the number of data */

if ( (WriteData = fopen(nf,"w")) == NULL )
{
   /* clrscr();*/
   printf("Open File %s for write data ERROR\n",nf);
   exit(1);
   }
number1 = (int)*(D1);
number2 = (int)*(D2);
if (number1 != number2)
   {
   printf("Data in Write_file_plot not equal\n");
   exit(1);
   }
fprintf(WriteData,"%lf\n",*(D1));
for (i=1;i<(number1+1);i++)
   {
   fprintf(WriteData,"%lf  %lf\n",*(D1+i),*(D2+i));
   }
printf("\nSave data to file %s ready\n",nf);
fclose(WriteData);
}
```

```
void Write_file_mat_3(double *D1,double *D2,double *D3,char nf[])
/* This function is for store the data used for plot graph which is the */
/* result of simulation function in the form which suitable for mathamatica */
/* program here is to plot three states of the problem by mathematica. */
{
int i,number1,number2,number3;
/* i for counter , number is the number of data */

if ( (WriteData = fopen(nf,"w")) == NULL )
{
   /* clrscr();*/
   printf("Open File %s for write data ERROR\n",nf);
   exit(1);
   }
number1 = (int)*(D1);
number2 = (int)*(D2);
number3 = (int)*(D3);
if ((number1 != number2) && (number1 != number3))
   {
   printf("Data in Write_file_mat_3 not equal\n");
   exit(1);
   }
/* fprintf(WriteData,"%lf\n",*(D1)); */
/* In This case not store the number of data for plot */
for (i=1;i<(number1+1);i++)
   {
   fprintf(WriteData,"%lf %lf %lf\n",*(D1+i),*(D2+i),*(D3+i));
   }
```

```
printf("\nSave data to file %s ready\n",nf);
fclose(WriteData);
}



double *Retrieve_write_file(char nf[])
/* This function is used with "Write_file" for retrieved the data from a  */
/* file in disk that store the data for manipulating.                     */
{
double *R=NULL;
int i,number;
double dummy;

if ( (InputDatafile = fopen(nf,"rt")) == NULL )
   {
    /* clrscr();*/
   printf("Not have this file name or loading error\n");
   exit(1);
   }
fscanf(InputDatafile,"%lf",&dummy);
number = (int)dummy;
R = (double *)calloc(number+1,sizeof(double));
if(R)
{
*(R) = number;
                                                    /* BT 1
```

```
                                                    printf("\n number of data in this file
= %lf\n",*R);
                                                    ET 1 */
for (i=1;i<number+1;i++)
    {
    fscanf(InputDatafile,"%lf",&dummy);
    *(R+i) = dummy;
       }
fclose(InputDatafile);
return(R);
}  /* belong to  if(R) */
else
{
 /* clrscr();*/
printf("\n\n\nNot enough memory\n");
exit(1);
return(R);      /* not necessary but complier warning */
}   /* belong to else of if(R) */
}
```



```
double *InputMatrix_File(char nf[])
{
double *M;
int    total,m,n;
int    i,j;        /* counter */
double dummy;
/* total collect the number of element of matrix which is equal m*n+2 */
```

```
/* m is the number of ROW ;  n is the number of COLUMN */
if ( (InputDatafile = fopen(nf,"rt")) == NULL )
   {
    /* clrscr();*/
   printf("Not have this file name or loading error\n");
   exit(1);
   }
fscanf(InputDatafile,"%d",&m);
fscanf(InputDatafile,"%d",&n);
total = (m*n)+2;
M =(double *)calloc(total,sizeof(double)); /* allocate memory for matrix */
if (M)
{
*(M) = m;
*(M+1) = n;
for (i=1;i<(m+1);i++)
   {
   for (j=1;j<(n+1);j++)
        {
        fscanf(InputDatafile,"%lf",&dummy);
        *(M+ij(M,i,j)) = dummy;
        }
   }

fclose(InputDatafile);
return (M);
}
else
```

```
{
 /* clrscr();*/
printf("\n\n\nNot enough memory\n");
exit(1);
return(M);      /* not necessary but complier warning */
}
}

void makezero(double *M)
{
int i,j;
int m,n;
m = *(M);
n = *(M+1);
for (i=1;i<m+1;i++)
    for (j=1;j<n+1;j++)
        {
        *(M+ij(M,i,j)) = 0;
        }
}

void makeone(double *M)
{
int i,j;
int m,n;
m = *(M);
n = *(M+1);
for (i=1;i<m+1;i++)
```

```
    for (j=1;j<n+1;j++)
        {
        *(M+ij(M,i,j)) = 1.0;
        }
}

void makeminus(double *M)
{
int i,j;
int m,n;
double dummy;
m = *(M);
n = *(M+1);
for (i=1;i<m+1;i++)
    for (j=1;j<n+1;j++)
        {
        dummy = 0.0 - ( *(M+ij(M,i,j)) );
        *(M+ij(M,i,j)) = dummy;
        }
}

double *transpose(double *M)
{
int m,n,rm,rm,total;
int i,j; /* counter */
double *R;
m = *M;
n = *(M+1);
```

```
total = (m*n)+2;
R = (double *)calloc(total,sizeof(double));
if (R)
   {}
   else
   {
    /* clrscr();*/
   printf("\n\nNot Enough Memory in calloc matrix in transpose funciton");
   printf("\n Hit a key to exit");
    /* getch(); */
   exit(1);
   }
rm  = n;    /* rm store transposed matrix row */
m  = m;    /* m store transposed matrix column */
*R = rm;
*(R+1) = m;
for (i=1;i<(m+1);i++)
   {
   for(j=1;j<(n+1);j++)
      {
      *(R+ij(R,j,i)) = *(M+ij(M,i,j));
      }
   }
return(R);
}

int ij(double *M,int i,int j)
{
```

```
int ij;
int total;
int m,n;
m = (int)(*(M));
n = (int)(*(M+1));
total = m*n+2;
if ( total > i*j)
   {
   ij = (int) ( (i-1)*(int)n+(j-1)+2 );
   return(ij);
   }
   else
   {
    /* clrscr();*/
   printf("\n\n\nOver Range in access matrix in function ij");
   exit(0);
   return(0); /* not necessary but complier warning */
   }
}

void outputmatrix(double *M)
{
int m,n;
int i,j;
/*
printf("\n\n\n");
*/
m = (int)(*M);
```

```
n = (int)(*(M+1));
for (i=1;i<(m+1);i++)
    {
    for (j=1;j<(n+1);j++)
        {
        printf("element [%d][%d] = %lf\n",i,j,*(M+ij(M,i,j)));
        }
    }
}

/* ======================== Matrix Arithematic
========================== */

/*---------------------- Addition of two Matrix ----------------------*/
double *addmatrix(double *M1,double *M2)
/* When use this function the caller function must define pointer to   */
/* double for recieve RESULT from this function and free it when finish */
/* the variable ussage */
{
int m,n,o,p;
int i,j,total;
double *RESULT;    /* for output */
m = *M1;
n = *(M1+1);
o = *M2;
p = *(M2+1);
if ( (m != o)||(n != p) )
  {
```

```
    /* clrscr();*/
    printf("\n\n\n\n\n\n\n\n\n");
    printf("Cannot add two Matrix\n");
    exit(1);
    }
total = (m*n)+2;
RESULT = (double *)calloc(total,sizeof(double));
/* allocate memory for matrix */
if (RESULT)
{
*(RESULT)   = m;
*(RESULT+1) = n;
for (i=1; i<(m+1) ;i++)
    {
    for(j=1; j<(n+1) ;j++)
       {
       *(RESULT + ( (i-1)*(n)+(j-1)+2 )) = *(M1 + ( (i-1)*(n)+(j-1)+2 )) + *(M2 + (
(i-1)*(n)+(j-1)+2 ));
       }
     }
return (RESULT);
}
else
{
 /* clrscr();*/
printf("\n\n\nNot enough memory\n");
exit(0);
return(RESULT);       /* not necessary but complier warning */
```

```
 }
 }
/*---------------------- End of Add Matrix ----------------------*/
/*------------------- Substraction of two Matrix --------------------*/
double *submatrix(double *M1,double *M2)
{
int m,n,o,p;
int i,j,total;
double *M3;    /* for output */
m = *M1;
n = *(M1+1);
o = *M2;
p = *(M2+1);
if ( (m != o)||(n != p) )
   {
    /* clrscr();*/
   printf("\n\n\n\n\n\n\n\n");
   printf("Cannot substract these two Matrix\n");
   exit(1);
   }
total = (m*n)+2;
M3 = (double *)calloc(total,sizeof(double));/* allocate memory for matrix */
if (M3)
{
*(M3)   = m;
*(M3+1) = n;
for (i=1; i<(m+1) ;i++)
   {
```

```
    for(j=1; j<(n+1) ;j++)
       {
       *(M3 + ( (i-1)*(n)+(j-1)+2 )) = *(M1 + ( (i-1)*(n)+(j-1)+2 )) - *(M2 + ( (i-
1)*(n)+(j-1)+2 ));
       }
    }
return (M3);
}
else
{
 /* clrscr();*/
printf("\n\n\nNot enough memory\n");
exit(0);
return(M3);       /* not necessary but complier warning */
}
}
/*---------------------- End of Substraction ----------------------*/


/*-------------------- Multiplication of two Matrix ---------------*/
double *multiplication(double *M1,double *M2)
/* When use this function the caller function must define pointer to   */
/* double for recieve RESULT from this function and free it when finish */
/* the variable ussage */
{
int m,n,o,p,k;
int i,j,total;
double *RESULT;              /* for output to main program */
```

```
double dummy1;
m = *M1;
n = *(M1+1);
o = *M2;
p = *(M2+1);
if ( n != o )
   {
    /* clrscr();*/
   printf("\n\n\n\n\n\n\n\n\n");
   printf("Cannot make multiplication of these two Matrix\n");
   exit(1);
   }
total = (m*p)+2;
RESULT = (double *)calloc(total,sizeof(double));
/* allocate memory for matrix */
if (RESULT)
{
*(RESULT)   = m;
*(RESULT+1) = p;
for (i=1; i<(m+1) ;i++)
   {
   for(j=1; j<(p+1) ;j++)
     {
      dummy1 = 0;
      for(k=1; k<(n+1) ;k++)
         {
         dummy1 = (*(M1 + ( (i-1)*(n)+(k-1)+2 ))) * (*(M2 + ( (k-1)*(p)+(j-1)+2 )))
+ dummy1 ;
```

```
            }
        *(RESULT + ( (i-1)*(p)+(j-1)+2 )) = dummy1;
        }
      }
 return (RESULT);
 }
 else
 {
  /* clrscr();*/
 printf("\n\n\nNot enough memory\n");
 exit(0);
 return(RESULT);       /* not necessary but complier warning */
 }
 }
 /*----------------------- End of multiplication ----------------------*/

 /*-------------------------- Copy Matrix -------------------------*/
 void copymatrix(double *S,double *D)  /* S source ; D destination */
 {
 int i,j,m,n,o,p;
 m = (int)(*(S));
 n = (int)(*(S+1));
 o = (int)(*(D));
 p = (int)(*(D+1));
 if ( (m!=o) || (n!=p))
    {
    printf("\n size not match in copy_matrix function\n");
    printf("m = %d,n = %d,o = %d,p = %d\n",m,n,o,p);
```

```
    outputmatrix(S);
    printf("\n");
    outputmatrix(D);
    printf("\nHit a Key for exit");
    exit(1);
    }
    else
    {
         for (i=1; i<(m+1) ;i++)
             {
             for(j=1; j<(n+1) ;j++)
                {
                *(D + ( (i-1)*(n)+(j-1)+2 )) = *(S + ( (i-1)*(n)+(j-1)+2 ));
                }
             }
    }

} /* End of function copymatrix */
/*---------------------- End of copy Matrix ------------------*/

/*==========================================================
========*/
/*========================= End of Matrix and RK4
=======================*/
/*==========================================================
========*/
```

ภาคผนวก ง

ตัวอย่างของเมตริกซ์เริ่มต้นที่ใช้ในโปรแกรมจำลองแบบระบบ

เมตริกซ์เริ่มต้นที่ใช้ในโปรแกรมจำลองแบบระบบจะแยกอยู่ในแต่ละแฟ้มข้อมูล โดยแต่ละแฟ้มข้อมูลจะเริ่มต้นด้วยจำนวนแถวของเมตริกซ์ในบรรทัดแรก และตามด้วยจำนวนสดมภ์ในบรรทัดที่สอง ส่วนในบรรทัดต่อๆไปจะเป็นข้อมูลของแต่ละสมาชิกดังนี้ คือ บรรทัดที่สามเป็นสมาชิกในแถวที่หนึ่ง สดมภ์ที่หนึ่งของเมตริกซ์ บรรทัดที่สี่เป็นสมาชิกในแถวที่หนึ่ง สดมภ์ที่สองของเมตริกซ์ บรรทัดที่ห้าเป็นสมาชิกในแถวที่หนึ่ง สดมภ์ที่สามของเมตริกซ์ เรียงกันไปเรื่อยๆจนหมดตามจำนวนแถว และเริ่มต้นที่แถวที่สองสดมภ์ที่หนึ่งต่อไป เช่น

เมตริกซ์ **A** ในตัวอย่างที่หนึ่ง เขียนได้ดังนี้คือ

3
3
0
1
0
0
0
1
0
-2
-3

และตั้งชื่อว่าแฟ้มข้อมูลนี้ว่า a_matrix

เมตริกซ์ **B** ในตัวอย่างเดียวกัน เขียนได้ว่า

3
1
0
0
1

และตั้งชื่อแฟ้มข้อมูลนี้ว่า b_matrix

เมตริกซ์ที่เก็บค่าเริ่มต้นของระบบในตัวอย่างที่หนึ่ง เขียนได้ว่า

3
1
1
-1
1

และตั้งชื่อแฟ้มข้อมูลนี้ว่า init_x

เมตริกซ์ที่เก็บค่าเริ่มต้นของสัญญาณควบคุม เขียนได้ว่า

1
1
0

และตั้งชื่อแฟ้มข้อมูลนี้ว่า init_u

เมตริกซ์ที่ที่เก็บค่าของพารามิเตอร์ของพื้นผิวสวิตชิงในตัวอย่างที่หนึ่ง เขียนได้ว่า

3
1
12
5
1

และตั้งชื่อแฟ้มข้อมูลนี้ว่า s_matrix

เมตริกซ์ที่เก็บค่าของอัตราขยายที่สัมพันธ์กับพื้นผิวสวิตชิงในตัวอย่างที่หนึ่ง เขียนได้ว่า

3

1

-24

0

0

และตั้งชื่อแฟ้มข้อมูลนี้ว่า k_matrix

ส่วนเมตริกซ์ที่เก็บค่าเริ่มต้นของอัตราขยายที่สัมพันธ์กับสัญญาณควบคุมแบบสวิตชิง เขียนได้ว่า

3

1

0

0

0

และตั้งชื่อแฟ้มข้อมูลนี้ว่า init_vk

ส่วนเมตริกซ์ที่สำคัญอีกสองเมตริกซ์คือเมตริกซ์ **M**(T) ก็สามารถเขียนได้ในทำนองเดียวกันแต่วิธีการตั้งชื่อต้องตั้งให้สัมพันธ์กับชื่อแฟ้มข้อมูลที่เก็บอยู่ในตัวแปร *name_N_p และ *name_N_n ซึ่งอยู่ในส่วนต้นของโปรแกรมจำลองแบบระบบ นอกจากนั้นเมตริกซ์ **M**(T) นี้จะต้องเปลี่ยนค่าของสมาชิกเมื่อเปลี่ยนค่าของเวลาชักตัวอย่าง

ภาคผนวก จ

โปรแกรมคอมพิวเตอร์ที่ใช้แสดงผลออกทางหน้าจอคอมพิวเตอร์

```
#include<stdio.h>
#include<stdlib.h>
#include<process.h>
#include<conio.h>
#include<alloc.h>
#include<math.h>
#include<graphics.h>

/* Graphic Function */
void prepare_graph(void);
int convy(int real_x);
void plot(double *X,double *Y,double x_min_plot,double x_max_plot,double
y_min_plot,double y_max_plot,int color);
void plot_to_monitor_at(int x,int y,int color);
int make_suitable(double v,double v_min,double step);
void displaytext(int left,int top,int right,int bottom,int fcolor,int tcolor,char *A);

FILE *InputDatafile;          /* For Input Data File from Disk */
double *Retrieve_write_file(char nf[]);
```

```
int main(void)
{

double *PX,*PX1,*PX2,*PX3,*PX4,*PPS,*PPN,*PPB;
double x_min = 0;
double x_max = 5;
double y_min = -1;
double y_max = 1;
/* begin variable for changing Boundary to + and - value */
int number=0,i;
/* end variable for changing Boundary to + and - value */
clrscr();
printf("\nHit a key for printgraph");
getch();
PX = Retrieve_write_file("b:\\4_19\\x_axis.");

PX1 = Retrieve_write_file("b:\\4_19\\x1_axis.");
PX2 = Retrieve_write_file("b:\\4_19\\x2_axis.");
PX3 = Retrieve_write_file("b:\\4_19\\x3_axis.");
PX4 = Retrieve_write_file("b:\\4_19\\x4_axis.");


PPS= Retrieve_write_file("b:\\4_19\\pps.");
PPN= Retrieve_write_file("b:\\4_19\\ppn.");
PPB= Retrieve_write_file("b:\\4_19\\ppb.");

prepare_graph();
/*
```

```
plot(PX,PPN,x_min,x_max,y_min,y_max,MAGENTA);
*/

plot(PX,PX1,x_min,x_max,y_min,y_max,WHITE);

/*
plot(PX,PX2,x_min,x_max,y_min,y_max,BLUE);
plot(PX,PX3,x_min,x_max,y_min,y_max,CYAN);
plot(PX,PX4,x_min,x_max,y_min,y_max,YELLOW);
*/
/*
plot(PX,PPB,x_min,x_max,y_min,y_max,GREEN);
number = *PPB;
for (i=1;i<number+1;i++)
    {
    *(PPB+i) = 0.0 - *(PPB+i);
    }
plot(PX,PPB,x_min,x_max,y_min,y_max,GREEN);
plot(PX,PPS,x_min,x_max,y_min,y_max,BROWN);
*/
getch();
closegraph();
return(0);
}

double *Retrieve_write_file(char nf[])
/* This function is used with "Write_file" for retrieved the data from a  */
/* file in disk that store the data for manipulating.                      */
```

```
{
double *R=NULL;
int i,number;
double dummy;

if ( (InputDatafile = fopen(nf,"rt")) == NULL )
   {
   clrscr();
   printf("Not have this file name or loading error\n");
   exit(1);
   }
fscanf(InputDatafile,"%lf",&dummy);
number = (int)dummy;
R = (double *)calloc(number+1,sizeof(double));
if(R)
{
*(R) = number;
                                        /* BT 1
                                        printf("\n number of data in this file
= %lf\n",*R);
                                        ET 1 */
for (i=1;i<number+1;i++)
   {
   fscanf(InputDatafile,"%lf",&dummy);
   *(R+i) = dummy;
   /*
   printf("data number %d = %lf\n",i,*(R+i));
   getch();  */
```

```
    }
fclose(InputDatafile);
return(R);
}  /* belong to  if(R) */
else
{
clrscr();
printf("\n\n\nNot enough memory\n");
exit(1);
return(R);       /* not necessary but complier warning */
}   /* belong to else of if(R) */
}
```

```
/*                              GFI.CPP                                  */
/* This file is for include to another program for graph output purpose */
/*          Program contains the functions for graphic output          */
/* An important function is the plot function which call another function */
/* to manipulate datas which coming for plotting. The "PLOT" function   */
/* requires seven parameters that are                                    */
/* 1. An array of X axis's datas which form in an array of double having */
/*    the first component store the number of data contained in the array */
/*    and the other components contained datas, so this array must have  */
/*    the total number of data = number of data + 1.                     */
/* 2. An array of Y axis'datas which are stored in the same way as X     */
/* 3. The minimum and maximum of X and Y which will display on screen.   */
/*    These datas are divined into four parameters.                      */
/* 4. Color for that graph.                                              */
```

```
/* The important thing in plot the coordinate to screen is the screen   */
/* coordinate (0,0) is on the upper left of screen which don't locate   */
/* at the same position to the X-Y axis,so for plot the value of Y      */
/* the conversion must be done before                                   */



void prepare_graph(void)
{
/* This function is for opening the graph */
/* You must remember to use "CLOSEGRAPH" function before leave out  */
/* the graphic screen. */
int gdriver = DETECT,gmode,errorcode;

initgraph(&gdriver,&gmode,"c:\\bc\\bgi");
errorcode = graphresult();
if (errorcode != grOk)
   {
   clrscr();
   printf("Graphic Error : %s\n",grapherrormsg(errorcode));
   exit(1);
   }
/* Graph initialization is OK. Now let us go to the user program */
cleardevice();
}

void plot_to_monitor_at(int x,int y,int color)
{
```

```
/* This  function sends the ready made point for plot to screen */
moveto(x,y);
putpixel(x,y,color);
}


int convy(int real_y)
{
/* This program convert a real y value to a plot y for plot. */
int plot_point_y;
plot_point_y = getmaxy()-real_y;
return(plot_point_y);
}

void plot(double *X,double *Y,double x_min_plot,double x_max_plot,double
y_min_plot,double y_max_plot,int color)
/* This function plot the coordinate (x,y) out. */
/* The coordinates are stored in a separate array,each array begin with */
/* the number of data point (Array size = data point + 1).          */
/* This function requires the range of x and y for plotting.        */
/* Must be use with the function make_suitable.                     */

{
double number_x,number_y;   /* the number of elemant of x and y must equal */
/* This number_,number_y store the number of point for plot. */
double plot_span_x,plot_span_y; /* the span of plot (real value) */
double real_step_x,real_step_y; /* the increment of real value for plot */
int i; /* counter */
```

```
int x_plot_here,y_plot_here;   /* screen point (x,y) */
int y_should_plot_here;/* plot to point y before compensate by convy function */
char A[80];
double dummyx,dummyy;

number_x = *X;
number_y = *Y;
if (number_x == number_y)
{
plot_span_x = x_max_plot - x_min_plot;
plot_span_y = y_max_plot - y_min_plot;
real_step_x = plot_span_x/(getmaxx()+1);
real_step_y = plot_span_y/(getmaxy()+1);

/* Draw Frame */
setcolor(GREEN);
line(0,0,getmaxx(),0);
line(getmaxx(),0,getmaxx(),getmaxy());
line(getmaxx(),getmaxy(),0,getmaxy());
line(0,getmaxy(),0,0);

/* Tell about Maximum Range on screen */
sprintf(A,"[%.2lf,%.2lf]",x_min_plot,y_min_plot);
displaytext(1,getmaxy()-1-textheight(A),1+textwidth(A),getmaxy()-1,RED,CYAN,A);
sprintf(A,"[%.2lf,%.2lf]",x_max_plot,y_min_plot);
displaytext(getmaxx()-1-textwidth(A),getmaxy()-1-textheight(A),getmaxx()-1,getmaxy()-
1,RED,CYAN,A);
sprintf(A,"[%.2lf,%.2lf]",x_min_plot,y_max_plot);
```

```
displaytext(1,1,1+textwidth(A),1+textheight(A),RED,CYAN,A);
sprintf(A,"[%.2lf,%.2lf]",x_max_plot,y_max_plot);
displaytext(getmaxx()-1-textwidth(A),1,getmaxx()-1,1+textheight(A),RED,CYAN,A);

/* Draw center line */
setcolor(RED);
dummyx = make_suitable(0.5*plot_span_x+x_min_plot,x_min_plot,real_step_x);
line((int)dummyx,0,(int)dummyx,getmaxy());
dummyy =
convy(make_suitable(0.5*plot_span_y+y_min_plot,y_min_plot,real_step_y));
line(0,dummyy,getmaxx(),dummyy);



for (i=1;i<(number_x+1);i++)
    {
    if( ((*(X+i) > x_max_plot)||(*(X+i) < x_min_plot))||((*(Y+i) >
y_max_plot)||(*(Y+i) < y_min_plot)) )
      {
      i =i+1;
      }
      else
      {
      x_plot_here = make_suitable(*(X+i),x_min_plot,real_step_x);
      y_should_plot_here = make_suitable(*(Y+i),y_min_plot,real_step_y);
      y_plot_here = convy(y_should_plot_here);
      /* send to monitor */
      plot_to_monitor_at(x_plot_here,y_plot_here,color);
      /* BT 1 */
```

```
      /*
      dummyx = x_plot_here;
      dummyy = y_plot_here;
      sprintf(A,"The last point that plot is (x,y) = (%lf,%lf)",*(X+i),*(Y+i));
      getch();
      */
      } /* else of if */
    }   /* end of for loop */
    /*
    displaytext(1,1,1+textwidth(A),1+textheight(A),RED,CYAN,A);
    sprintf(A,"the last point on screen coordinate is (x,y) =
(%lf,%lf)",dummyx,dummyy);
    displaytext(1,12,1+textwidth(A),12+textheight(A),RED,CYAN,A);
    */
}
/* if (number_x == number_y) */
else
{
closegraph();
clrscr();
printf("\n\n\n   Can't plot graph because the incomplete coordinate\n");
exit(1);
}
}   /* End of function plot */


int make_suitable(double v,double v_min,double step)
{
```

```
/* This function use with the function name "plot" only. */
/* This function does the repeat functions in plot coordinate x and y  */
/* calculation. */
double real_distance=0;
double plot_to_point=0;
int result;
double dummy1=0,dummy2=0;

real_distance = v - v_min;
/* Test */
/*
if (real_distance<0.0)
   {
  real_distance = 0.0 - real_distance;
   }
*/
plot_to_point = real_distance/step;
/* plot_to_point have the integer plus fraction double value */
/* dummy1 store the fraction , dummy2 store the integer */
dummy1 = modf(plot_to_point,&dummy2);
if (dummy1>=0.5)
   {
   dummy2 = dummy2+1;
   }
result = (int)dummy2;
return(result);
}
```

```
void displaytext(int left,int top,int right,int bottom,int fcolor,int tcolor,char *A)
{
/* This function output array of character in graphic screen */
/* The value which are sent to this function is in the screen coordinate */
/* not a normal coordinate                                              */
/* fcolor is frame color and tcolor is textcolor                        */
int  check;

setcolor(fcolor);
line(left-1,top-1,right+1,top-1);
line(right+1,top-1,right+1,bottom+1);
line(right+1,bottom+1,left-1,bottom+1);
line(left-1,bottom+1,left-1,top-1);
setcolor(tcolor);
setviewport(left,top,right,bottom,1); /* 1 mean clip text */
check = graphresult();
if (check != -11)
   {
   clearviewport();
   outtextxy(1,1,A);
   setviewport(0,0,getmaxx(),getmaxy(),1); /* 1 mean clip text */
   }
   else
   {
   outtextxy(0,0,"Error in setting set view port");
   outtextxy(0,10,"You should check your value in function displaytext call");
   getch();
   }
```

```
}
/*================================================================
========*/
/*============================ End of Graphic
============================*/
/*================================================================
========*/
```

ประวัติผู้เขียน

นายมณเฑียร เศาภายน เกิดวันที่ 27 พฤษภาคม พ.ศ. 2513 ที่กรุงเทพมหานคร สำเร็จการศึกษาปริญญาตรีวิศวกรรมศาสตรบัณฑิต สาขาระบบควบคุม ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2533 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต ที่จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2534 ระหว่างการศึกษาได้ทำหน้าที่เป็นผู้ช่วยสอนของห้องปฏิบัติการระบบควบคุม ภาควิชาวิศวกรรมไฟฟ้า