เอกสารอ้างอิง

ภาษาไทย

บุญมี อย่างธารา ดร. , เอกสารประกอบวิชา Numerical Methods , พ.ศ. 2536

ภาษาต่างประเทศ

Asada H. and Slotine J. J., "Robot Analysis and Control" , John Wiley & Son , 1986 .

Bakshi , "Wave-Net" , AIChe Journal , Vol. 39 , No. 12 , 1993 .

Bhat N. , "System Identification and Control of Dynamic Chemical Process using Backpropagation Network", PhD Dissertation at University of Maryland , 1990 .

——— and McAvoy T. J. , "Use of Neural Networks for Dynamic Modeling and Control of Chemical Process Systems", Comp Chem Engng , Vol. 14 , No. 4 , 1990 .

Cooper D. J. , Hinde Jr. R. F. and Megan L. , "Pattern-Based Adaptive Process Control" , Comp Chem Engng , Vol. 14 , No. 12 , 1990 .

——— , Hinde Jr. R. F. and Megan L. , "Comparing Two Neural Networks for Pattern-Based Adaptive Process Control" , AIChe Journal , Vol. 38 , No. 1 , 1992 .

Craig J J. , "Introduction to Robotics Mechanics and Control " , Addison-Wesley , 1986 .

Cybenko G. , " Approximation by Superposition of a Sigmoidal Function", Math. Control Signals System , No. 2 , 1989 .

Fukuda T. , Shibata T. , Sato M. and Harashima F. , "Neuromorphic Control" , IEEE Industrial Electronics , Vol 39 , No. 6 , 1992 .

Hecht-Nielsen R. , "Neurocomputing" , Addison-Wesley , 1990 .

Hunt K.J. and Sbarbaro D.,"Neural Networks for Nonlinear Internal Model Control" , IEE-D , Vol. 138 , No. 5 , 1991 .

Ishiguro A., Furuhashi T. and Uchikawa Y., "A Neural Network Compensator for Uncertainities of Robotic Manipulators", IEEE Industrial Electronics , Vol 39 , No. 6 , 1992 .

Jacek Z. , "Artificial Neural Systems", West Info Access , 1992 .

Jin L. , Nikiforuk P. N. and Gupta M. M. , "Direct Adaptive Output Tracking Control using Multilayered Neural Networks" . IEE-D , Vol. 140 , No. 6 , 1993 .

Johansson R. , "Adaptive Control of Robot Manipulator Motion", IEEE Tran on Robotics and Automation , Vol. 6 , No. 4 , 1990 .

Kosko B. , "Neural Networks and Fuzzy Systems" , Prentice Hall , 1992 .

Kramer M.A. and Leonard J.A. , "Diagnosis Using Backpropagation Neural Networks-Analysis and Criticism", Comp Chem Engng , Vol. 14 , No. 12 , 1990 .

———— , "Nonlinear Principal Component Analysis using Autoassociative Neural Networks", AIChe Journal , Vol. 37 , No. 2 , 1991 .

Levin A. U. , "Neural Networks in Dynamical System", PhD Dissertation at Yale University , 1992 .

Mathews J. H. , "Numerical Methods" , Prentice Hall , 1992 .

Nahas E. P. , Henson M. A. and Seborg D. E. , "Nonlinear Internal Model Control Strategy for Neural Network Models" , Comp Chem Engng , Vol. 16 , No. 12 , 1992 .

Nordgren R. E. and Meckl P. H. , "An Analytical Comparison of a Neural Network and a Model-Based Adaptive Controller", IEEE Neural Networks , Vol. 4 , No. 4 , 1993 .

Ogata K. , "Discrete-time Control Systems" , Prentice-Hall , 1987 .

Psidogios P. C. , "Direct and Indirect Model-Based Control using Artificial Neural Networks" , Ind Eng Chem Research , Vol. 30 , 1991 , p2567-p2573 .

Rumelhart D. E. and McClelland J. L. , "Parallel Distributed Processing" , Vol I . MIT Press , 1986 .

Sanner R. M. and Slotine J. J., "Gaussian Network for Direct Adaptive Control " , IEEE Neural Network , Vol 3 , No. 6 , 1992 .

Slotine J.J. and Li W., "Applied Nonlinear Control", Prentice Hall , 1991 .

Song Y. D. and Gao W. B. , "Path Tracking Control of Robot manipulators via the VSS approach" , Int J. Systems Sci , Vol. 22 , No. 1 , 1991 .

Sontag E. D. , "Feedback Stabilization using Two-Hidden-Layer Nets" , IEEE Neural Networks , Vol 3 , No. 6 , 1992 .

Spong M. , "Robot Dynamics and Control" , John Wiley and Son , 1989 .

——— , "On Robust Control of Robot Manipulators", IEEE AC , Vol 37 , No. 11 , 1992 .

Stoten D. P. , "Model Reference Adaptive Control of Manipulators" , Research Studies Press , 1990 .

Wu Q. H. , Hogg B. W. and Irwin G. W. , "A Neural Networks Regulator for Turbogenerators " , IEEE Neural Networks , Vol 3 , No. 1 , 1992 .

Yabuta T. and Yamada T. , "Neural Network Controller Characteristics with Regard to Adaptive Control", IEEE System, Man ,and Cybernetics , Vol 22 , No.1 , 1992 .

Ydstie B. E. , "Forcasting and Control using Adaptive Connectionist Network" , Comp Chem Engng , Vol. 14 , No. 4 , 1990 .

Zhang Q. and Benveniste A. , "Wavelet Networks" , IEEE Neural Networks , Vol 3 , No.6 , 1992 .

ภาคผนวกที่ 1

แบบจำลองพลวัตของแขนกล

สมการพลวัตของแขนกลในรูปที่ 3-1 สามารถเขียนได้ในรูปของสมการดังนี้ (Craig ,1986 ; Spong,1989 ;Spong 1992) คือ

$$M(x)\cdot\ddot{x}+C(x,\dot{x})\cdot\dot{x}+g(x)=u \qquad \text{(ผ1-1)}$$

โดยที่ $x,\dot{x},\ddot{x}$ คือ เมตริกซ์ที่แสดงตำแหน่ง ความเร็วและความเร่งของแต่ละข้อต่อของแขนกล มีมิติ 2 x 1

$M(x)$ คือ เมตริกซ์ความเฉื่อยของแขนกลและเป็นเมตริกซ์สมมาตร และมากกว่าศูนย์ (Symmetric and Positive Definite Metrix) มีมิติ 2 x 2

$C(x,\dot{x})$ คือ เมตริกซ์ที่แสดงแรงคอริโอริส และ แรงเซนตริพิทอล (Coriolis and Centripetal Forces) มีมิติ 2 x 1

$g(x)$ คือ เมตริกซ์ที่แสดงผลกระทบจากแรงโน้มถ่วง มีมิติ 2 x 1

u คือ เมตริกซ์ที่แสดงแรงบิดที่กระทำในแต่ละข้อต่อ ทำหน้าที่เป็นสัญญาณควบคุมแขนกล มีมิติ 2 x 1

สมการฟังก์ชันเมตริกซ์ในสมการ(ผ1-1) มีดังนี้

$$M(x)=\begin{bmatrix} p_1+p_2+2\cos(x_2)p_3 & p_2+\cos(x_2)p_3 \\ p_2+\cos(x_2)p_3 & p_2 \end{bmatrix} \qquad \text{(ผ1-2)}$$

$$C(x,\dot{x})=\begin{bmatrix} -2\dot{x}_2\sin(x_2)p_3 & -\dot{x}_2\sin(x_2)p_3 \\ \dot{x}_1\sin(x_2)p_3 & 0 \end{bmatrix} \qquad \text{(ผ1-3)}$$

$$g(x)=\begin{bmatrix} g\cos(x_1)(p_4+p_5)+g\cos(x_1+x_2)p_6 \\ g\cos(x_1+x_2)p_6 \end{bmatrix} \qquad \text{(ผ1-4)}$$

โดยที่ $p_1=m_1l_{c1}^2+m_2l_1^2+I_1$, $p_2=m_2l_{c2}^2+I_2$

$p_3 = m_2 l_1 l_{c2}$, $p_4 = m_1 l_{c1}$

$p_5 = m_2 l_1$, $p_6 = m_2 l_{c2}$

m_1 คือ มวลของแขนที่ 1 ของแขนกล

m_2 คือ มวลของแขนที่ 2 ของแขนกล

l_1 คือ ความยาวของแขนที่ 1

l_2 คือ ความยาวของแขนที่ 2

l_{c1} คือ ระยะจุดศูนย์กลางมวลของแขนที่ 1

l_{c2} คือ ระยะจุดศูนย์กลางมวลของแขนที่ 2

I_1 คือ โมเมนต์ความเฉื่อยของแขนที่ 1

I_2 คือ โมเมนต์ความเฉื่อยของแขนที่ 2

ค่าพารามิเตอร์ต่างๆของแขนกลที่ใช้ในการทดลอง ได้แสดงไว้ในตารางที่ ผ1-1

ตารางที่ ผ1-1 ตารางแสดงค่าพารามิเตอร์ต่างๆของแขนกล

m_1 (Kg)	m_2 (Kg)	l_1 (m)	l_2(m)	l_{c1}(m)	l_{c2}(m)	I_1 (Kg.m^2)	I_2 (Kg.m^2)
10 (Kg)	5	1	1	0.5	0.5	10/12	5/12

นอกจากนี้ยังสามารถเขียนสมการ(ผ1-1) ได้ในรูปของสมการพารามิเตอร์ไรเซชันเชิงเส้น(linear parameterization) คือ

$$\boldsymbol{M}(\boldsymbol{x})\cdot\ddot{\boldsymbol{x}}+\boldsymbol{C}(\boldsymbol{x},\dot{\boldsymbol{x}})\cdot\dot{\boldsymbol{x}}+\boldsymbol{g}(\boldsymbol{x})=\boldsymbol{T}(\boldsymbol{x},\dot{\boldsymbol{x}},\ddot{\boldsymbol{x}})\boldsymbol{p}=\boldsymbol{u} \quad \text{(ผ1-5)}$$

โดยที่ $\boldsymbol{T}(\boldsymbol{x},\dot{\boldsymbol{x}},\ddot{\boldsymbol{x}})$ เป็นเมตริกซ์พารามิเตอร์มีมิติ 2 x 6 ซึ่งมีค่าดังนี้

$T_{11} = \ddot{x}_1$, $T_{12} = \ddot{x}_1 + \ddot{x}_2$

$T_{13} = \cos(x_2)(2\ddot{x}_1 + \ddot{x}_2) - \sin(x_2)(\dot{x}_2^2 + 2\dot{x}_1\dot{x}_2)$

$T_{14} = g\cos(x_1)$, $T_{15} = g\cos(x_1)$

$T_{16} = g\cos(x_1 + x_2)$

$T_{21} = 0$, $T_{22} = \ddot{x}_1 + \ddot{x}_2$

$T_{23} = \cos(x_2)\ddot{x}_1 + \sin(x_2)\dot{x}_1^2$

$$T_{24} = 0 \qquad , T_{25} = 0$$
$$T_{26} = g\cos(x_1 + x_2)$$

และ $\boldsymbol{p}$ เป็นเวคเตอร์ของค่าพารามิเตอร์ มีมิติ 6 x 1 ซึ่งมีค่าดังนี้

$$\boldsymbol{p} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \end{bmatrix}^T$$
$$p_1 = m_1 l_{c1}^2 + m_2 l_1^2 + I_1 \quad , \quad p_2 = m_2 l_{c2}^2 + I_2$$
$$p_3 = m_2 l_1 l_{c2} \qquad , \quad p_4 = m_1 l_{c1}$$
$$p_5 = m_2 l_1 \qquad , \quad p_6 = m_2 l_{c2}$$

ภาคผนวกที่ 2

โปรแกรมเครือข่ายนิวรอล

โปรแกรมเกี่ยวกับเครือข่ายนิวรอลสำหรับการทำวิทยานิพนธ์ ซึ่งเขียนไว้เป็นภาษา C ได้แยกออกเป็น 4 ส่วนคือ

1. ไฟล์ NN.CPP,MAIN.Hและ NN.H เป็นไฟล์เกี่ยวกับการสร้างและการจัดเก็บข้อมูลของเครือข่ายนิวรอล
2. ไฟล์ LEARNING.CPP เป็นไฟล์เกี่ยวกับการเรียนรู้ของเครือข่ายนิวรอล
3. ไฟล์ SIMNN.CPP เป็นไฟล์เกี่ยวกับการจำลองการทำงานของเครือข่ายนิวรอล
4. ไฟล์ WEIGHTADJ.CPP เป็นไฟล์เกี่ยวกับการปรับค่าพารามิเตอร์ของเครือข่ายนิวรอลด้วยอัลกอริทึม Backpropagation
5. ไฟล์ PLOT.H และ PLOT.CPP เป็นไฟล์เกี่ยวกับการวาดกราฟแสดงผลการเรียนรู้

1. ไฟล์ NN.CPP,MAIN.H และNN.H

1.1 MAIN.H

```
#include <dos.h>
#include <bios.h>
#include <math.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <graphics.h>
#include <time.h>
#include <alloc.h>
#include <ctype.h>
#define cntlq '\021'
```

1.2 NN.H

```
#include "main.h"
typedef struct { float **in,**out ;
                 int   numin,numout;
                 int   numlong;
               } data;
```

```
/************ NN Structure *********************/
typedef struct  { float far   ***w,***pw,***initw,**a;
                  float far   **wbias,**initwbias,**pwbias;
                  float far   **y,**py,**sy,**y_sum,**pa;
                  float far   ***dykbydyi;
                  int       *nnode_layer,*sgmtype;
                  int       layer;
                  float     lrate[8],lbias_rate[9];
                 float     lmomentum_rate[8],lsgm_rate[8];
                  char   name[30];
                  int    status;
                  FILE   *file;
                 } nn ;

/********* NN1 & NN2  **************/
extern void   initnn(nn *n,char display[])          ;
extern void   newnn(nn *n,char display[]);
extern void   chkmemnn(nn *n)   ;
extern void   chkmem(float *n)  ;
extern void   savenn(nn *n,char display[]);
extern void   loadnn(nn *n,char display[]);
extern void   freenn(nn *n)     ;
extern void   allocnn(nn *n); /*   know  layer , nnodelayer*/
extern void   control(nn *n)            ;
extern int    chkbioskey(void);
extern float  get(char display[],int x,int y);
extern int    chkeofnn(nn *n);
extern void   autosavenn(nn *n,char fname[]);
extern void   autoloadnn(nn *n,char fname[]);
```

1.3) NN.CPP

```
#include "main.h"
#include "nn.h"
void screendat();
void initnn(nn *n,char display[]);
void getdatalearning(nn *n);
void newnn(nn *n,char display[]);
void allocnn(nn *n) ; /*  know  layer , nnodelayer*/
void chkmemnn(nn *n);
void chkmem(float *n);
void chkmemfile(FILE *n);
void savenn(nn *n,char display[]);
void autosavenn(nn *n,char fname[]);
void autoloadnn(nn *n,char fname[]);
void loadnn(nn *n,char display[]);
void freenn(nn *n);
int  chkbioskey(void);
void screendat()
{    clrscr();
   printf("\n L-oad network ");
   printf("\n N-ew  network ");
   printf("\n S-ave network ");
   printf("\n D-ata for learning ");
   printf("\n Q-uit network ");
}
void initnn(nn *n,char display[])
{
char c;
 screendat();
 c=getch();
while(c!='q'){
 switch(c) {
 case 'l' : loadnn(n,display) ;  screendat();  break;
 case 'n' : newnn(n,display);    screendat();  break;
 case 's' : savenn(n,display) ;  screendat();  break;
 case 'd' : getdatalearning(n) ;   screendat();  break;
           };
 c=getch();
           };
}
```

```
void  getdatalearning(nn *n)
{char file[30];
clrscr();
printf("Data file name=    "); scanf("%s",&file);
strcpy(n->name,file);
}
void  newnn(nn *n,char display[])
{
int i,l,j,*nnodelayer,layer,sgmtype;
int dummy;
char key='m';
FILE  *f;
float  delta,***w,***initw,***pw,**pwbias,**wbias
     ,**initwbias,**a,**pa,minn=-0.25,maxx=0.25,fr;
long buf;
char file[30];
freenn(n);
clrscr();
gotoxy(30,30);

/*************
 FORMAT OF DATA FILE

    layer
    nnodelayer[0]
    nnodelayer[1]
    .
    .
    nnodelayer[]
    nnodelayer[layer-1]
    SGMTYPE  OF LAYER 0 :  sgmtype {0 == sgm(tanh)
                                    1 == gaussian
                                    2 == LINEAR SLOPE = 1
                                    3 == sgm (1/(1+exp(-ax))}
    SGMTYPE OF LAYER 1
    SGMTYPE OF LAYER 2
    .
    .
    SGMTYPE OF LAYER LAYER-1

***************/
printf("%s \n" ,display);
printf("Enter Neural net config file =    "); scanf("%s",&file);
f = fopen(file,"r");   if(f==NULL) goto end;
fscanf(f,"%d",&layer); n->layer = layer ;
```

```
n->nnode_layer  = (int *)calloc(layer,sizeof(int));
n->sgmtype      = (int *)calloc(layer,sizeof(int));
nnodelayer = n->nnode_layer;
for(i=0;i<layer;i++)
  {fscanf(f,"%d",nnodelayer+i);};
for(i=0;i<layer;i++)
 {fscanf(f,"%d",&sgmtype); n->sgmtype[i] = sgmtype;
 };
allocnn(n);
printf("\nNUMBER OF LAYER = %d \n",n->layer);
for(i=0;i<layer;i++)
  {printf("NUM NODELAYER[%d] = %d  \n",i,nnodelayer[i]);};
for(i=0;i<layer;i++)
 {printf("SGMTYPE LAYER[%d] =  %d  \n",i,n->sgmtype[i]);
 };
w  =n->w;
pw =n->pw;
initw =n->initw;
wbias  =n->wbias;
initwbias  =n->initwbias;
pwbias =n->pwbias;
while( (key != 'y') && (key != 'n'))
{  printf("\nDo you want to set range of random weight.? (Y/N)") ;
   key  =  getch();
};
if(key=='y')
{
 for(l=1;l<layer;l++)
 { dummy = l-1;
 printf("\nEnter range of WEIGHT LAYER[%d] :  ",l);
 printf("\n                min =  ");scanf("%f",&minn);
 printf("\n                max =  ");scanf("%f",&maxx);
 fr =  (maxx-minn)*200;
   for(i=0;i<nnodelayer[dummy];i++)
   for(j=0;j<nnodelayer[l];j++)
    {
     w[l][i][j]  =  (int) random((int)fr) ;
     w[l][i][j]  = pw[l][i][j]  =  w[l][i][j]/200.+minn ;
     if(w[l][i][j]==0)  pw[l][i][j]  =w[l][i][j]=0.0001;
    };
 };
 for(l=1;l<layer;l++)
 {
 printf("\nEnter range of BIAS WEIGHT LAYER[%d] :  ",l);
 printf("\n                min =  ");scanf("%f",&minn);
```

```
 printf("\n              max =  ");scanf("%f",&maxx);
if(nnodelayer[l]>1)
{  fr =  (maxx-minn)/((float)nnodelayer[l]-1);
  for(i=0;i<nnodelayer[l];i++)
    {
     wbias[l][i]  = initwbias[l][i]  =  pwbias[l][i]  =  fr*(float)i+minn ;
     if(wbias[l][i]==0)
        { pwbias[l][i] =initwbias[l][i]  = wbias[l][i] = 0.0001; }
    };
}else
{ fr =  (maxx-minn)*200;
  for(i=0;i<nnodelayer[l];i++)
    {
     wbias[l][i]  =  (int) random((int)fr) ;
     wbias[l][i]  = initwbias[l][i]  =  pwbias[l][i]  =  fr/200.+minn ;
     if(wbias[l][i]==0)
        { pwbias[l][i]  = wbias[l][i] = initwbias[l][i] = 0.0001; }
    };
};
};
a  = n->a;
pa = n->pa;
  for(l=1;l<layer;l++)
    for(i=0;i<nnodelayer[l];i++)
      {
      if(n->sgmtype[l]==2)
      {a[l][i]  = pa[l][i]  = 1.0;
      }else
      {
      if(n->sgmtype[l]==0)
      {a[l][i]  = pa[l][i]  = 0.2;
      };
      if(n->sgmtype[l]==3)
      {a[l][i]  = pa[l][i]  = 0.2;
      };
      if(n->sgmtype[l]==1)
      {a[l][i]  = pa[l][i]  = 1/.1;
      };
      }
      };
 };/** END IF random loop  ***/
 fclose(f);
 end : return;
 }
```

```
/*  know  layer , nnodelayer*/
void allocnn(nn *n)
{int  i=0,l=0,layer,*nnodelayer,dummy;
float ***w,***initw,***pw,***dykbydyi;
float **ybuf;
layer = n->layer ;
nnodelayer   = n->nnode_layer;
n->dykbydyi  = (float far ***)farcalloc(layer,sizeof(float));
chkmem((float *) n->dykbydyi);
dykbydyi = n->dykbydyi;
for(l=0;l<layer;l++)
{  *(dykbydyi+l) = (float **)calloc(nnodelayer[l],sizeof(float));
  chkmem((float *) *(dykbydyi+l));
};
 for(l=1;l<layer;l++)
  { for(i=0;i<nnodelayer[l];i++)
    {dummy = l-1;
     *(*(dykbydyi+l)+i) = (float *)calloc(nnodelayer[dummy],sizeof(float));
     chkmem(*(*(dykbydyi+l)+i));
    };
  };
/*********** calloc W  ***********/
n->w  = (float far ***)farcalloc(layer,sizeof(float));chkmem((float *)n->w);
w = n->w;
for(l=1;l<layer;l++)
{ dummy = l-1;
 *(w+l) = (float **)calloc(nnodelayer[dummy],sizeof(float));
 chkmem((float *)*(w+l));
};
 for(l=1;l<layer;l++)
  { dummy =l-1;
    for(i=0;i<nnodelayer[dummy];i++)
    {*(*(w+l)+i) = (float *)calloc(nnodelayer[l],sizeof(float));
     chkmem(*(*(w+l)+i));
    };
  };
/*********** calloc PW  ***********/
n->pw = (float far ***)farcalloc(layer,sizeof(float));chkmem((float *)n->pw);
pw = n->pw;
for(l=1;l<layer;l++)
{ dummy =l-1;
*(pw+l) = (float **)calloc(nnodelayer[dummy],sizeof(float));
 chkmem((float *)*(pw+l));
};
```

```
 for(l=1;l<layer;l++)
 { dummy =l-1;
  for(i=0;i<nnodelayer[dummy];i++)
  {  *(*(pw+l)+i) = (float *)calloc(nnodelayer[l],sizeof(float));
    chkmem(*(*(pw+l)+i));
  };
 };
/*********** calloc initW  ***********/
n->initw = (float far ***)farcalloc(layer,sizeof(float));
chkmem((float *)n->initw);
initw = n->initw;
for(l=1;l<layer;l++)
{ dummy =l-1;
*(initw+l) = (float **)calloc(nnodelayer[dummy],sizeof(float));
 chkmem((float *)*(initw+l));
};
 for(l=1;l<layer;l++)
 { dummy =l-1;
  for(i=0;i<nnodelayer[dummy];i++)
  {  *(*(initw+l)+i) = (float *)calloc(nnodelayer[l],sizeof(float));
    chkmem(*(*(initw+l)+i));
  };
 };
/*********** calloc Wbias  ***********/
n->wbias  = (float far **)farcalloc(layer,sizeof(float));
chkmem((float *)n->wbias);
ybuf = n->wbias;
for(l=0;l<layer;l++)
{*(ybuf+l) = (float *)calloc(nnodelayer[l],sizeof(float));
chkmem(*(ybuf+l));
};
/*********** calloc initWbias  ***********/
n->initwbias  = (float far **)farcalloc(layer,sizeof(float));
chkmem((float *)n->initwbias);
ybuf = n->initwbias;
for(l=0;l<layer;l++)
{*(ybuf+l) = (float *)calloc(nnodelayer[l],sizeof(float));
chkmem(*(ybuf+l));
};
```

```
/*********** calloc PWbias ***********/
n->pwbias  = (float far **)farcalloc(layer,sizeof(float));
chkmem((float *)n->pwbias);
ybuf = n->pwbias;
for(l=0;l<layer;l++)
{*(ybuf+l) = (float *)calloc(nnodelayer[l],sizeof(float));
 chkmem(*(ybuf+l));
};
/*********** calloc Y  ***********/
n->y  = (float far **)farcalloc(layer,sizeof(float));chkmem((float *)n->y);
ybuf = n->y;
for(l=0;l<layer;l++)
{*(ybuf+l) = (float *)calloc(nnodelayer[l],sizeof(float));chkmem(*(ybuf+l));
};
/*********** calloc SY  ***********/
n->sy = (float far **)farcalloc(layer,sizeof(float));chkmem((float *) n->sy);
ybuf = n->sy;
for(l=0;l<layer;l++)
{*(ybuf+l) = (float *)calloc(nnodelayer[l],sizeof(float));
 chkmem((float *)*(ybuf+l));
};
/*********** calloc Y_sum  ***********/
n->y_sum = (float far **)farcalloc(layer,sizeof(float));
chkmem((float *) n->y_sum);
ybuf = n->y_sum;
for(l=0;l<layer;l++)
{*(ybuf+l) = (float *)calloc(nnodelayer[l],sizeof(float));chkmem(*(ybuf+l));
};
/*********** calloc PY  ***********/
n->py = (float far **)farcalloc(layer,sizeof(float));
chkmem((float *) n->py);
ybuf = n->py;
for(l=0;l<layer;l++)
{*(ybuf+l) = (float *)calloc(nnodelayer[l],sizeof(float));chkmem(*(ybuf+l));
};
/*********** calloc A ***********/
n->a  = (float far **)farcalloc(layer,sizeof(float));
chkmem((float *) n->a);
ybuf = n->a;
for(l=0;l<layer;l++)
{*(ybuf+l) = (float *)calloc(nnodelayer[l],sizeof(float));chkmem(*(ybuf+l));
};
```

```
/********** calloc PA **********/
n->pa  = (float far **)farcalloc(layer,sizeof(float));
chkmem((float *) n->pa);
ybuf = n->pa;
for(l=0;l<layer;l++)
{*(ybuf+l) = (float *)calloc(nnodelayer[l],sizeof(float));chkmem(*(ybuf+l));
};
for (l=layer-1;l>=0;l--)
n->lrate[l]         = 0.000  ;
for (l=layer-1;l>=0;l--)
n->lmomentum_rate[l] = 0.000  ;
for (l=layer-1;l>=0;l--)
n->lbias_rate[l]    = 0.000  ;
for (l=layer-1;l>=0;l--)
n->lsgm_rate[l]     = 0.000  ;
return;
}
void chkmemnn(nn *n)
{ if(n==NULL)
  {sound(300);delay(5);nosound();
  printf("NULL POINTER ASSIGNMENT  \n \n");
  };
}
void chkmem(float *n)
{ if(n==NULL)
  {sound(300); delay(5);nosound();
  printf("NULL POINTER ASSIGNMENT \n \n");
  };
}
void chkmemfile(FILE *n)
{ if(n==NULL)
  {sound(300); delay(5) ;nosound();
  printf("NULL POINTER ASSIGNMENT \n \n");
  };
}
```

```
void savenn(nn *n,char display[])
{ FILE *f;
 char fname[30];
 int  l=0,i=0,j=0,layer=0,dummy;
 float ***w,**a,**wbias;
 int *nnodelayer;
 clrscr();
    gotoxy(30,30);
    printf("%s\n",display);
    printf("filename =      ");
    scanf ("%s",&fname);
/*  layer
   nnode_layer
   sgmtype
   w
   wbias
   a
*/
a  = n->a  ;
w =  n->w;
wbias =  n->wbias;
layer =n->layer;
nnodelayer =n->nnode_layer;
f = fopen(fname,"w");  chkmemfile(f);
fprintf(f,"%d   \n"    ,n->layer);
for(i=0;i<layer;i++)
{ fprintf(f,"%d \n",nnodelayer[i]);};
for(i=0;i<layer;i++)
fprintf(f,"%d \n",n->sgmtype[i]);
for(l=1;l<layer;l++)
 {dummy =l-1;
  for(i=0;i<nnodelayer[dummy];i++)
  for(j=0;j<nnodelayer[l];j++)
   fprintf(f," %f \n",w[l][i][j]);
 }
for(l=1;l<layer;l++)
 for(i=0;i<nnodelayer[l];i++)
  { fprintf(f,"%f \n",wbias[l][i]);
  };
for(l=1;l<layer;l++)
 for(i=0;i<nnodelayer[l];i++)
  fprintf(f," %f \n",a[l][i]);
fclose(f);
return;
}
```

```
void loadnn(nn *n,char display[])
{ FILE  *f;
 char  fname[30];
 int  i,l,j,*nnodelayer,layer,sgmtype=9,dummy;
 float ***w,**a,***initw,***pw,**pa,**initwbias,**wbias,**pwbias;
 clrscr(); gotoxy(30,30);
    printf("%s\n",display);
    printf("input filename =      ");
    scanf ("%s",&fname);
freenn(n);
/*** File Structure ********************
   layer
   nnode_layer
   sgmtype
   w
   wbias
   a
*/
f = fopen(fname,"r");   chkmemfile(f);
fscanf(f,"%d",&layer); n->layer = layer ;
nnodelayer =  n->nnode_layer = (int *)calloc(layer,sizeof(int));
n->sgmtype    = (int *)calloc(layer,sizeof(int));
chkmem((float *)nnodelayer);
for(i=0;i<layer;i++)
{ fscanf(f,"%d",nnodelayer+i);};
for(i=0;i<layer;i++)
{fscanf(f,"%d",&sgmtype); n->sgmtype[i] = sgmtype;
};
allocnn(n);
printf("\nNUMBER OF LAYER = %d \n",n->layer);
for(i=0;i<layer;i++)
  {printf("NUM NODELAYER[%d] = %d  \n",i,nnodelayer[i]);};
for(i=0;i<layer;i++)
 {printf("SGMTYPE LAYER[%d] =  %d  \n",i,n->sgmtype[i]);
 };
w =  n->w;
a  = n->a ;
pw =  n->pw;
initw =  n->initw;
pa  = n->pa ;
wbias =  n->wbias;
initwbias =  n->initwbias;
pwbias =  n->pwbias;
nnodelayer =n->nnode_layer;
```

```
for(l=1;l<layer;l++)
 {dummy =l-1;
 for(i=0;i<nnodelayer[dummy];i++)
 for(j=0;j<nnodelayer[l];j++)
  { fscanf(f,"%f",&w[l][i][j]);
    pw[l][i][j] = initw[l][i][j] = w[l][i][j];
  };
 };
for(l=1;l<layer;l++)
 for(i=0;i<nnodelayer[l];i++)
  { fscanf(f,"%f",&wbias[l][i]);
    pwbias[l][i]  = initwbias[l][i] = wbias[l][i] ;
  };
for(l=1;l<layer;l++)
 for(i=0;i<nnodelayer[l];i++)
  { fscanf(f,"%f",&a[l][i]);
    pa[l][i]  = a[l][i] ;
  };
fclose(f);
return;
}
void freenn(nn *n)
{
int  i=0,l=0,layer;
int  *nnodelayer,dummy;
float ***w,***initw,***pw,***dykbydyi;
float **ybuf;
layer = n->layer ;
nnodelayer   = n->nnode_layer;
w = n->w;
/*********** FREE W  ***********/
  for(l=1;l<layer;l++)
   {dummy =l-1;
    for(i=0;i<nnodelayer[dummy];i++)
      free(*(*(w+l)+i));
   };
  for(l=1;l<layer;l++)
    free(*(w+l));
 free(w);
 /*********** FREE PW  ***********/
 pw = n->pw;
  for(l=1;l<layer;l++)
    { dummy =l-1;
      for(i=0;i<nnodelayer[dummy];i++)
      free(*(*(pw+l)+i));
```

```
    }
  for(l=1;l<layer;l++)
    free(*(pw+l));
free(pw);
/*********** FREE initW  ***********/
initw = n->initw;
  for(l=1;l<layer;l++)
    { dummy =l-1;
      for(i=0;i<nnodelayer[dummy];i++)
      free(*(*(initw+l)+i));
    }
  for(l=1;l<layer;l++)
    free(*(initw+l));
free(initw);
/*********** FREE Wbias  ***********/
ybuf = n->wbias;
for(l=0;l<layer;l++)
  free(*(ybuf+l));
free(n->wbias);
/*********** FREE initWbias  ***********/
ybuf = n->initwbias;
for(l=0;l<layer;l++)
  free(*(ybuf+l));
free(n->initwbias);
/*********** FREE pWbias  ***********/
ybuf = n->pwbias;
for(l=0;l<layer;l++)
  free(*(ybuf+l));
free(n->pwbias);
/*********** FREE Y  ***********/
ybuf = n->y;
for(l=0;l<layer;l++)
  free(*(ybuf+l));
free(ybuf);
/*********** FREE SY  ***********/
ybuf = n->sy;
for(l=0;l<layer;l++)
 free(*(ybuf+l));
free(ybuf);
/*********** FREE Y_sum  ***********/
ybuf = n->y_sum;
for(l=0;l<layer;l++)
free(*(ybuf+l));
free(ybuf);
```

```
/********** FREE PY  **********/
ybuf = n->py;
for(l=0;l<layer;l++)
 free(*(ybuf+l));
free(ybuf);
/********** FREE A **********/
ybuf = n->a;
for(l=0;l<layer;l++)
free(*(ybuf+l));
free(ybuf);
/********** FREE PA **********/
ybuf = n->pa;
for(l=0;l<layer;l++)
free(*(ybuf+l));
free(ybuf);
/********** FREE dykbydyi  **********/
dykbydyi = n->dykbydyi;
for(l=0;l<layer;l++)
 for(i=0;i<nnodelayer[l];i++)
  free( *(*(dykbydyi+l)+i) );
for(l=0;l<layer;l++)
 free( *(dykbydyi+l) );
free(dykbydyi);
/********** FREE nnode_layer  **********/
free(n->nnode_layer);
}
int  chkbioskey(void)
{ int key;
  key = bioskey(1);
  if(key!=0) bioskey(0);
  return(key);
}
int chkeofnn(nn *n)
{
 /* check for EOF */
   if (feof(n->file))
   { fclose(n->file); n->file = fopen(n->name,"r");return(1);}
return(0);
}
```

```
void autosavenn(nn *n,char fname[])
{ FILE *f;
 int  l=0,i=0,j=0,layer=0,dummy;
 float ***w,**a,**wbias;
 int *nnodelayer;
/*  layer
   nnode_layer
   sgmtype
   w
   wbias
   a
*/
a  = n->a  ;
w =  n->w;
wbias =  n->wbias;
layer =n->layer;
nnodelayer =n->nnode_layer;
f = fopen(fname,"w");  chkmemfile(f);
fprintf(f,"%d   \n"    ,n->layer);
for(i=0;i<layer;i++)
{ fprintf(f,"%d \n",nnodelayer[i]);};
for(i=0;i<layer;i++)
fprintf(f,"%d \n",n->sgmtype[i]);
for(l=1;l<layer;l++)
 {dummy =l-1;
  for(i=0;i<nnodelayer[dummy];i++)
  for(j=0;j<nnodelayer[l];j++)
   fprintf(f," %f \n",w[l][i][j]);
 }
for(l=1;l<layer;l++)
 for(i=0;i<nnodelayer[l];i++)
  { fprintf(f,"%f \n",wbias[l][i]);
  };
for(l=1;l<layer;l++)
 for(i=0;i<nnodelayer[l];i++)
  fprintf(f," %f \n",a[l][i]);
fclose(f);
return;
}
```

```
void autoloadnn(nn *n,char fname[])
{ FILE  *f;
  int   i,l,j,*nnodelayer,layer,sgmtype=9,dummy;
  float ***w,**a,***initw,***pw,**pa,**initwbias,**wbias,**pwbias;
freenn(n);
/*  layer
    nnode_layer
    sgmtype
    w
    wbias
    a
*/
f = fopen(fname,"r");   chkmemfile(f);
fscanf(f,"%d",&layer); n->layer = layer ;
nnodelayer =  n->nnode_layer = (int *)calloc(layer,sizeof(int));
n->sgmtype    = (int *)calloc(layer,sizeof(int));
chkmem((float *)nnodelayer);
for(i=0;i<layer;i++)
{ fscanf(f,"%d",nnodelayer+i);};
for(i=0;i<layer;i++)
{fscanf(f,"%d",&sgmtype); n->sgmtype[i] = sgmtype;
};
allocnn(n);
printf("\nNUMBER OF LAYER = %d \n",n->layer);
for(i=0;i<layer;i++)
  {printf("NUM NODELAYER[%d] = %d  \n",i,nnodelayer[i]);};
for(i=0;i<layer;i++)
 {printf("SGMTYPE LAYER[%d] =  %d  \n",i,n->sgmtype[i]); };
w =  n->w;
a  = n->a ;
pw =  n->pw;
initw =  n->initw;
pa  = n->pa ;
wbias =  n->wbias;
initwbias =  n->initwbias;
pwbias =  n->pwbias;
nnodelayer =n->nnode_layer;
for(l=1;l<layer;l++)
 {dummy =l-1;
 for(i=0;i<nnodelayer[dummy];i++)
 for(j=0;j<nnodelayer[l];j++)
  { fscanf(f,"%f",&w[l][i][j]);
    pw[l][i][j] = initw[l][i][j] = w[l][i][j];
  };
};
```

```
for(l=1;l<layer;l++)
 for(i=0;i<nnodelayer[l];i++)
  { fscanf(f,"%f",&wbias[l][i]);
    pwbias[l][i]  = initwbias[l][i] = wbias[l][i] ;
  };
for(l=1;l<layer;l++)
 for(i=0;i<nnodelayer[l];i++)
  { fscanf(f,"%f",&a[l][i]);
    pa[l][i]  = a[l][i] ;
  };
fclose(f);
return;
}
```

2. LEARNING.CPPและLEARNING.H

2.1 LEARNING.H

```
#include "simnn.h"
extern void   learning(nn *n);
extern void   testing(nn *n);
extern void   weightadj(nn *n,float *data)                ;
extern void   extractdata(nn *n,float *input,float *output)    ;
extern float dykdai(nn *n,int l,int k,int m,int i);
extern float  dykdwbiasi(nn *n,int l,int k,int m,int i);
extern float dykdwij(nn *n,int l,int k,int m,int i,int j);
extern void simdykbydyi(nn *n);
extern void caldykbydyi(nn *n,int l,int k,int i);
extern float dykdyi(nn *n,int l,int k,int m,int i);
```

2.2 LEARNING.CPP

```
#include "main.h"
#include "nn.h"
#include "plot.h"
#include "learnnn.h"
#include "simnn.h"
#include <alloc.h>
void learning(nn *n);
void extractdata(nn *n,float *input,float *output);
void learning(nn *n)
{int i=0,numpoint,plotstate=1;
int dummy;
 char key='i';
 float *y,*input,*output;
 char string[25];
 long value = 123456789L;
 plotconfig  p;
 n->file = fopen(n->name,"r");
```

```
input  = (float *)calloc(n->nnode_layer[0],sizeof(float));
dummy  = (n->layer)-1;
output = (float *)calloc(n->nnode_layer[dummy],sizeof(float));
y      = (float *)calloc(n->nnode_layer[dummy],sizeof(float));
initgphc();    /*  must run initgphc  before  getmaxx(),getmaxy()  */
numpoint=n->nnode_layer[dummy];
setplotconfig(&p,.1,.9,-1,1);
plotregion(p,"L-rate M-momemtum Bias A-sgmrate On/oFf R-eset Q-uit");
for( i = 0 ;  ;  i++)
{
extractdata(n,input,output);
simnn(n,input,y);
weightadj(n,output);
key = chkbioskey();
  if(plotstate==1)
{
/*
simnn(n,input,y);
*/
scaledata(output,y,numpoint,p);
plotpoint(output,y,numpoint,i,p);
};
  if(i== (int) (p.maxx-p.minx))
{ setfillstyle(SOLID_FILL,0);
 bar(p.minx-5+1,p.miny-5+1,p.maxx+5-1,p.maxy+5-1);
 i=0;
};

  switch(key)
  {  case 'l': getrate("L_rate",n->layer,n->lrate); break;
     case 'b': getrate("Bias_rate",n->layer,n->lbias_rate); break;
     case 'm': getrate("Momentum_rate",n->layer,n->lmomentum_rate); break;
     case 'a': getrate("Sigmoid_rate",n->layer,n->lsgm_rate); break;
     case 'r': fclose(n->file); n->file = fopen(n->name,"r");break;
     case 'h': value = farcoreleft();ltoa(value,string,10);
                 setfillstyle(SOLID_LINE,BLACK);
                 bar(0,10,0+textwidth(string)+15+8*15,10+8);
                 outtextxy(0,10,string);
                 break;
     case 'o'    : plotstate = 1;break;
     case '\021' : key='q';   break;
     case 'f'    : plotstate = 0;break;
  };
  if(key=='q') break;
};
```

```
closegraph();
return;
}
void extractdata(nn *n,float *input,float *output)
{ int  layer,i,numinput,numoutput;
 int dummy;
 float in;
 FILE *f;
start : f = n->file;
layer     = n->layer;
numinput  = n->nnode_layer[0];
dummy = layer -1;
numoutput = n->nnode_layer[dummy];
for(i=0;i<numinput;i++)
 {fscanf(f,"%f",&in);
 *(input+i) =    in;
 if(chkeofnn(n)==1) goto start;
 };
for(i=0;i<numoutput;i++)
 { fscanf(f,"%f",&in);
  *(output+i) =  in;
  if(chkeofnn(n)==1) goto start;
 };
 return;
}
```

3. SIMNN.H และ SIMNN.CPP

3.1 SIMNN.H

```
extern float limit(float a,float x);
extern float limitb(float min,float max,float x);
extern void simnn(nn *n,float *data,float *yy);
extern float  dsgm(float a,float x,int sgmtype);
extern float  sgm(float a,float x,int sgmtype);
extern float  A;
extern void getrate(char string[30],int layer,float *rate);
extern float get(char display[],int x,int y);
```

3.2 SIMNN.CPP

```
#include "main.h"
#include "nn.h"
#include "simnn.h"
float get(char display[],int x,int y);
void getrate(char str[],int layer,float *rate);
void simnn(nn *n,float *data,float *yy);
float dsgm(float a,float x,int sgmtype);
float limit(float a,float x);
float limitb(float min,float max,float x);
float sgm(float a,float x,int sgmtype);
float A=0.2;
void simnn(nn *n,float *data,float *yy)
{int  layer;
 float **y,**sy,**y_sum,**a,***w,**wbias;
 int i=0,j=0,l=0;
 int *nnodelayer;
 int dummy1,dummy2,dummy3;
 float sum=0;
 nnodelayer = n->nnode_layer;
 layer     = n->layer;
 y  = n->y;
 y_sum  = n->y_sum;
 sy = n->sy;
 w  = n->w;
 wbias  = n->wbias;
 a  = n->a;
for(i=0;i<nnodelayer[0];i++)
   { y[0][i]=data[i];
   };
for(l=1;l<layer;l++)
{  for(j=0;j<nnodelayer[l];j++)
   { sum=0;
     dummy1=l-1;
     for(i=0; i<nnodelayer[dummy1];i++)
         { dummy2 =l-1;
           sum += w[l][i][j] * y[dummy2][i];
         };
     sum += wbias[l][j];

     y_sum[l][j] = sum;
     y[l][j]     = sgm(a[l][j],sum,n->sgmtype[l]);
     sy[l][j]    = dsgm(a[l][j],sum,n->sgmtype[l]);
   };
};
```

```
dummy1 = layer -1;
for(i=0;i<nnodelayer[dummy1];i++)
 { dummy3 = layer -1;
   yy[i] = y[dummy3][i];};
return;
}

float dsgm(float a,float x,int sgmtype)
{float y;
 switch(sgmtype)
{case 1: y  = (float) exp((double)limit(20.,-x*x*a*a))*(-2.*x*a*a);break;
 case 0: y  = (float) pow((double)limit(20.,1./(float)cosh((double) (a*x))),2.);
         y  = a*y ;break;
 case 2: y  = a   ;break;
 case 3:  /*   limit  y E [-50,50]  avoid floating
           pnt   error*/
          y =  (float) exp((double) limit(20.,-a*x));
          y =limit (3000.,y);
          y  = a*y/(1+y)/(1+y);break;

 default: printf("\n wrong sgmtype  \n");
};
return (y);
}
float limit(float a,float x)
{ if(x>a)   return(a);
  if(x<-a)  return(-a);
  return(x);
}
float limitb(float min,float max,float x)
{ if(x>max)   return(max);
  if(x<min)  return(min);
  return(x);
}
float sgm(float a,float xx,int sgmtype)
{float y;
 switch(sgmtype)
{case 1: y  = (float) exp((double) limit(20.,-xx*xx*a*a));break;
 case 0: y  = (float) tanh((double)(xx*a));break;
 case 2: y  = xx*a;break;
 case 3: y  = 1/(1+limit(10000.,(float)exp((double) limit(20.,-a*xx))))   ;break;
 default: printf("\n wrong sgmtype  \n");
};
 return (y);
}
```

```
void getrate(char str[],int layer,float *rate)
{ int l,i,pos;
 char *str1 = "layer[",string[30];
 strcpy(string,str);
 for(i=0;i<30;i++)
  { if(string[i]=='\x0')
    {pos =i;break;};
  };
strcpy(string+pos,str1);
for(l=1;l<layer;l++)
 {
itoa(l,string+pos+6,10);
strcpy(string+pos+7,"] = ");
rate[l] = get(string,0,10);
 };
return;
}

float get(char display[],int x,int y)
{ float  v;
 int i=0;
 int  wid=0;
 char  c[50]="",key;
  setfillstyle(SOLID_LINE,BLACK);
  bar(x,y,x+textwidth(display)+50+8*50,y+8);
 setcolor(WHITE);
 outtextxy(x,y,display); wid=x+textwidth(display)+15;
 while((key=getch())!='\r')
  {     if(key=='\b')
        {  setcolor(BLACK);  outtextxy(wid,y,c);
          *(c+i-1) = ' ';
          setcolor(WHITE);  outtextxy(wid,y,c);
          i-=1;
        }
        else
        {
         if(i<50)
          {*(c+i)=key;
           outtextxy(wid,y,c);
           ++i;
           };
        };
  };
 v= atof(c);
 return(v);}
```

4. WEIGHTADJ.CPP

```
#include "main.h"
#include "nn.h"
#include "learnnn.h"
void weightadj(nn *n,float *data);
float dykdai(nn *n,int l,int k,int m,int i);
float  dykdwbiasi(nn *n,int l,int k,int m,int i);
float dykdwij(nn *n,int l,int k,int m,int i,int j);
void simdykbydyi(nn *n);
void caldykbydyi(nn *n,int l,int k,int i);
float dykdyi(nn *n,int l,int k,int m,int i);
void weightadj(nn *n,float *data)
{
 int layer,ii=0,i=0,j=0,l=0;
 int *nnodelayer;
 float **y,**a,**pa,***w,***pw,**wbias,**pwbias;
 float *lr,*lm,*lb,*la,djdw=0,djda=0,momentum=0,aux1,aux2;
 int dummy1,dummy2;
 nnodelayer= n->nnode_layer;
 layer     = n->layer;
 lr        = n->lrate;
 lb        = n->lbias_rate;
 lm        = n->lmomentum_rate;
 la        = n->lsgm_rate;
 y  = n->y;   w  = n->w;
 wbias  = n->wbias; pwbias = n->pwbias;
 pw = n->pw;  a  = n->a;    pa = n->pa;
simdykbydyi(n);
/*****  Adjust Weight  *****/
for (l=1;l<layer;l++)
 {dummy1 = l-1;
  for (i=0;i<nnodelayer[dummy1];i++)
   for (j=0;j<nnodelayer[l];j++)
     { djdw=0;
       dummy2 = layer -1;
       for(ii=0; ii< nnodelayer[dummy2] ;ii++)
           { aux1  = -(data[ii]-y[dummy2][ii]);
             aux2  = dykdwij(n,layer-1,ii,l,i,j);
             djdw +=  aux1*aux2;
           };
           momentum = lm[l] * (w[l][i][j] - pw[l][i][j] );
           pw[l][i][j] = w[l][i][j];
           w[l][i][j] += -djdw*lr[l]+momentum;
     };
 };
```

```
/***** Adjust Bias Weight *****/
for (l=1;l<layer;l++)
 for (i=0;i<nnodelayer[l];i++)
    { djdw=0;
     dummy1 = layer-1;
     for(ii=0;ii<nnodelayer[dummy1];ii++)
        { aux1  = -(data[ii]-y[dummy1][ii]);
          aux2  = dykdwbiasi(n,layer-1,ii,l,i);
          djdw += aux1*aux2;
          };
          momentum     = lm[l] * (wbias[l][i] - pwbias[l][i] );
          pwbias[l][i]  = wbias[l][i];
          wbias[l][i]  += -djdw*lb[l]+momentum;
    };

/***** Adjust Sigmoid Shape *****/
for (l=1;l<layer;l++)
 for (i=0;i<nnodelayer[l];i++)
    { djda=0;
     dummy1 = layer -1;
     for(ii=0;ii<nnodelayer[dummy1];ii++)
         { aux1 = -(data[ii]-y[dummy1][ii]);
           aux2 =dykdai(n,layer-1,ii,l,i);
           djda += aux1*aux2;
          };
          momentum  = 0.05 * (a[l][i] - pa[l][i] );
          pa[l][i]  = a[l][i];
          a[l][i]  += -djda*la[l]+momentum;
    };
 return;
 }

 void caldykbydyi(nn *n,int l,int k,int i)
 { float ***dykbydyi,***w,**sy;
 dykbydyi= n->dykbydyi;
 sy =n->sy;
 w  =n->w ;
 dykbydyi[l][k][i] = sy[l][k] * w[l][i][k] ;
 }
```

```
void simdykbydyi(nn *n)
{
  /********
         l
      dy
      ____k__     =   n->dykbydyi[l][k][i]

         l-1
      d y
         i
   *********/
int l,i,j;
float ***dykbydyi,***w,**sy;
int *nnode_layer,dummy;
nnode_layer = n->nnode_layer;
dykbydyi= n->dykbydyi;
sy = n->sy;
w  = n->w;
for (l=1;l<n->layer;l++)
{ for (i=0;i<nnode_layer[l];i++)
  {  dummy = l-1;
     for (j=0;j<nnode_layer[dummy];j++)
      { dykbydyi[l][i][j] = sy[l][i] * w[l][j][i] ;
      };
  };
};
return;
}

float dykdwij(nn *n,int l,int k,int m,int i,int j)
{
  /*****      l
            dy
           ___k__
             m
            dw
              ij   ; i to j
    *****/
 float dydw;
 int   ii;
 float ***dykbydyi,**y,**sy;
 int *nnode_layer,dummy;
 nnode_layer = n->nnode_layer;
 dykbydyi= n->dykbydyi;
 sy = n->sy;
```

```
y  = n->y;
if(l==m)
 { dummy =l-1;
  if(k==j)
    dydw = sy[l][k]*y[dummy][i];
  else
    dydw = 0;
 } else
 {  if(l>m)
    { if(l-1==m)
      {  dydw = dykbydyi[l][k][j] * dykdwij(n,l-1,j,m,i,j);
      }else
      {  dydw    =  0;
          dummy  = l-1;
          for(ii=0;ii<nnode_layer[dummy];ii++)
          {
           dydw += dykbydyi[l][k][ii] * dykdwij(n,l-1,ii,m,i,j);
          };
      };
    } else
    { printf("dydw error \n");sound(500);delay(50);nosound();
    };
 };
return(dydw);
}

float  dykdwbiasi(nn *n,int l,int k,int m,int i)
{  /********
         l
      dy
     ____k__
          m
      dwbias
           i
   *********/
float dydw;
int ii,dummy;
float ***dykbydyi,**sy;
int *nnode_layer;
dykbydyi= n->dykbydyi;
sy = n->sy;
nnode_layer = n->nnode_layer;
```

```
if(l==m)
 { if(k==i)
    dydw = sy[l][k];
  else
    dydw = 0;
 } else
 {  if(l>m)
    { if(l-1==m)
      {  dydw  = dykbydyi[l][k][i] * dykdwbiasi(n,l-1,i,m,i);
      }else
      {  dydw   = 0;
          dummy = l-1;
          for(ii=0;ii<nnode_layer[dummy];ii++)
          dydw += dykbydyi[l][k][ii] * dykdwbiasi(n,l-1,ii,m,i);
      };
    } else
    { printf("dydw bias error \n");sound(500);delay(50);nosound();
    };
 };
return(dydw);
}

float dykdai(nn *n,int l,int k,int m,int i)
{
  /********
         l
     dy
     ____k__
         m
     d a
         i
  *********/
float dyda=0;
int ii,dummy;
float ***dykbydyi,**a,**y_sum;
int *nnode_layer;
dykbydyi= n->dykbydyi;
nnode_layer = n->nnode_layer;
a =n->a;
y_sum= n->y_sum;
```

```
if(l==m)
 { if(k==i)
    dyda = dsgm(1,a[m][i]*y_sum[l][k],n->sgmtype[l]) * y_sum[l][k];
  else
    dyda = 0;
 } else
 {  if(l>m)
    { if(l-1==m)
      {  dyda  =  dykbydyi[l][k][i] * dykdai(n,l-1,i,m,i);
      }else
      {  dyda   =  0;
           dummy = l-1;
           for(ii=0;ii<nnode_layer[dummy];ii++)
           dyda +=  dykbydyi[l][k][ii] * dykdai(n,l-1,ii,m,i);
      };
    } else
    { printf("dydw sigmoid error \n");sound(500);delay(50);nosound();
    };
 };
return(dyda);
}

float dykdyi(nn *n,int l,int k,int m,int i)
{  /********
          l
      dy
     ____k__
          m
      d y
          i
   *********/
 float dydy=0.000000000000000;
 int ii,dummy;
 float ***dykbydyi;
 int *nnode_layer;
 dykbydyi= n->dykbydyi;
 nnode_layer = n->nnode_layer;
```

```
if(l == m+1)
 { dydy =   dykbydyi[l][k][i];
 } else
 {  if(l > m+1)
    {  dydy   =  0;
        dummy = l-1;
        for(ii=0;ii<nnode_layer[dummy];ii++)
         dydy +=  dykbydyi[l][k][ii] * dykdyi(n,l-1,ii,m,i);
    }  else
    { printf("dykdyi sigmoid error \n");sound(500);delay(50);nosound();
    };
 };
return(dydy);
}
```

5. PLOT.H และ PLOT.CPP

5.1 PLOT.H

```
typedef struct { float  minx,maxx,miny,maxy,cpx,cpy,dy;
                 float  minofsignal,maxofsignal;
               }  plotconfig;
extern void   scaledata(float *r,float *y,int numpoint,plotconfig p);
extern void   plotpoint(float *r,float *y,int numpoint,int itr,plotconfig p);
extern void   plottestpoint(float *r,int numpoint,int itr,plotconfig p);
extern void   plotregion(plotconfig p,char display[]);
extern void   initgphc();
extern float  limit(float x,float a);
extern void   setplotconfig(plotconfig *p,float percentmin,float percentmax,
float minofsignal,float maxofsignal);
extern void   PrnChar(int c);
extern void   ResetPrinter(void);
extern void   SetPrinter(void);
extern void   SetPrnUniDirection(void);
extern void   ConvertOneRow(int row,int line);
extern void   PrnDataOneRow(void);
extern void   PrnOneRow(int row,int line);
extern void   DumpScr(void);
extern char   *PrintBuffer;
extern int    PrinterReady,graphdriver;
extern int    Xmax,Ymax;
extern int bits[8];
```

5.2 PLOT.CPP

```
#include "main.h"
#include "plot.h"
void initgphc();  /* must run initgphc before getmaxx(),getmaxy() */
void plotregion(plotconfig p,char display[]);
void plotpoint(float *r,float *y,int numpoint,int itr,plotconfig p);
void plottestpoint(float *y,int numpoint,int itr,plotconfig p);
void scaledata(float *r,float *y,int numpoint,plotconfig p);
void  setplotconfig(plotconfig *p,float percentmin,float percentmax,
float minofsignal,float maxofsignal );
/* dump.c */
void  PrnChar(int c);
void  ResetPrinter(void);
void  SetPrinter(void);
void  SetPrnUniDirection(void);
void  ConvertOneRow(int row,int line);
void  PrnDataOneRow(void);
void  PrnOneRow(int row,int line);
void  DumpScr(void);
char  *PrintBuffer;
int   PrinterReady = 1,graphdriver;
int   Xmax,Ymax;
int bits[8] = {128,64,32,16,8,4,2,1};
void initgphc()  /* must run initgphc before getmaxx(),getmaxy() */
{ int gd=DETECT,gm;
 clrscr();
 detectgraph(&gd,&gm);
 initgraph(&gd,&gm,"");
 }

void plotregion(plotconfig p,char display[])
{ setlinestyle(0,0,1);
 setcolor(WHITE);
 line(p.minx-5,p.miny-5,p.minx-5,p.maxy+5);
 line(p.maxx+5,p.cpy,p.minx-5,p.cpy);
 line(p.maxx+5,p.maxy+5,p.minx-5,p.maxy+5);
 line(p.minx-5,p.miny-5,p.maxx+5,p.miny-5);
 line(p.maxx+5,p.miny-5,p.maxx+5,p.maxy+5);
 setlinestyle(0, 0, 1);
 setcolor(YELLOW);
 settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
 outtextxy(25,getmaxy()*0.95,display);
 settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
}
```

```
void plotpoint(float *r,float *y,int numpoint,int itr,plotconfig p)
{ int i=0,rr=0,yy=0;                              /* itr = iteration  */
 for(i=0;i<numpoint;i++)
  {rr = r[i];
   yy = y[i];
        setfillstyle(SOLID_FILL,i+3);
   setcolor(i+4);
    /*   circle(itr+p.minx,rr,2);
*/
   line(itr+p.minx,rr,itr+p.minx,rr);
    /*  r =output =data output  */
   setlinestyle(0, 0, 1);
   setcolor(i+10);
   line(itr+p.minx,yy,itr+p.minx,yy);
};
}

void plottestpoint(float *y,int numpoint,int itr,plotconfig p)
{ int i=0;                                  /* itr = iteration  */
 setlinestyle(0, 0, 1);
 for(i=0;i<numpoint;i++)
  {
   setfillstyle(SOLID_FILL,i+3);
   setcolor(i+3);
   line(itr+p.minx,*(y+i),itr+p.minx,*(y+i));
   };
}
void scaledata(float *r,float *y,int numpoint,plotconfig p)
{int i;
 for(i=0;i<numpoint;i++)
 { *(r+i) = p.dy*(-*(r+i))/ (p.maxofsignal-p.minofsignal)   +p.cpy;
     if(*(r+i)>p.maxy) *(r+i)=p.maxy;
   if(*(r+i)<p.miny) *(r+i)=p.miny;
   *(y+i) = p.dy*(-*(y+i))/ (p.maxofsignal-p.minofsignal) +p.cpy;
   if(*(y+i)>p.maxy) *(y+i)=p.maxy;
   if(*(y+i)<p.miny) *(y+i)=p.miny;
 }
}
```

```
void  setplotconfig(plotconfig *p,float percentmin,float percentmax,
float minofsignal,float maxofsignal )
{ p->maxx =  getmaxx();          p->maxy  =  getmaxy();
 p->minx  =  p->maxx*percentmin ;  p->miny  =  p->maxy*percentmin;
 p->maxx  =  percentmax*p->maxx;   p->maxy  =  percentmax*p->maxy;
 p->cpy   =  (p->maxy+p->miny)/2;  p->dy    =  p->maxy-p->miny;
 p->cpx   =  (p->maxx+p->minx)/2;
 p->minofsignal = minofsignal;
 p->maxofsignal = maxofsignal;
};
void PrnChar(int c)
{
  union REGS regs;
  if( PrinterReady ){
    regs.x.dx=0;
    regs.h.ah=0;
    regs.h.al=c;
    int86(0x17,&regs,&regs);
    if (( regs.h.ah & 0x01 )&& PrinterReady)
         PrinterReady--;
  }
}

void ResetPrinter()
{
  PrnChar(27);
  PrnChar('@');
}
void SetPrinter()
{
  int n1,n2;
  n1 = Xmax % 256;
  n2 = Xmax / 256;
  PrnChar(27);PrnChar('3');PrnChar(24);
  PrnChar(27);PrnChar('*');PrnChar((graphdriver==HERCMONO)?6:4);
  PrnChar(n1);PrnChar(n2);
}

void SetPrnUniDirection()
{
  PrnChar(27);PrnChar('u');PrnChar(1);
}
```

```
void ConvertOneRow(int row,int line)
{
  int xcor,ycor,endline,startline;
  startline= row*8;
  endline=startline+line;
  memset(PrintBuffer,0,Xmax);

  for(xcor=0;xcor<Xmax;xcor++)
    for(ycor=startline;ycor<endline;ycor++)
        PrintBuffer[xcor] |= (getpixel(xcor,ycor) ? bits[ycor-startline] :0);
}

void PrnDataOneRow()
{
  int column;
  for(column=0;column<Xmax;column++)
    PrnChar(PrintBuffer[column]);
}

void PrnOneRow(int row,int line)
{
  ConvertOneRow(row,line);
  SetPrinter();
  PrnDataOneRow();
  PrnChar('\n');
}
void DumpScr()
{  int i, MaxFullRow,RemainLine , gmode;
  detectgraph(&graphdriver, &gmode);
  Ymax = getmaxy()+1;
  Xmax = getmaxx()+1;
  MaxFullRow = Ymax/8;
  RemainLine = Ymax%8;
  if (!PrinterReady)
    PrinterReady++;
  PrintBuffer=(char *) malloc(Xmax*sizeof(char));
  ResetPrinter();
  SetPrnUniDirection();
  for (i=0;i<MaxFullRow;i++)
    PrnOneRow(i,8);
  PrnOneRow(MaxFullRow,RemainLine);
  PrnChar(12);
  ResetPrinter();
  free(PrintBuffer);
}
```

ภาคผนวกที่ 3

โปรแกรมจำลองการชดเชยแบบปรับได้ด้วยเครือข่ายนิวรอล

โปรแกรมเกี่ยวกับเครือข่ายนิวรอลสำหรับการทำวิทยานิพนธ์ ซึ่งเขียนไว้เป็นภาษา C ได้แยกออกเป็น 4 ส่วนคือ

1. ไฟล์ LEARNRB.CPP เป็นไฟล์เกี่ยวกับการเรียนรู้สมการ(3-18) ด้วย BNN
2. ไฟล์ ADAPTNN.CPP เป็นไฟล์จำลองการชดเชยแบบปรับได้ด้วย BNN
3. ไฟล์ SIMRB.H และ SIMRB.CPP เป็นไฟล์จำลองการทำงานของแขนกล

1. LEARNRB.CPP

```
#include "nn.h"
#include "learnnn.h"
#include "simrb.h"
#include "plot.h"
#include "simnn.h"
typedef struct { robotconfig  *rc;
                 float       *nxm,*nx,*x,*dxm,*u;
                 double  *cost;
                 int         select,Nmodel,Nl,index,count;
                 nn          *nnc1,*nnc2;
               } NMP;
void submenu();
void learncontrolrb(nn *nnm1,nn *nnm2);
float r1=0,r2=0;
float *y,*inputm,*output;
float xf1_1=0,xf2_1=0,nxf1_1,nxf2_1,uf_1=0;
float xf1_2=0,xf2_2=0,nxf1_2,nxf2_2,uf_2=0;
void submenu()
{
dt = 0.05;//.025
clrscr();
printf("1- about NNC1 \n");
printf("2- about NNC2 \n");
printf("L- learning  \n");
printf("Q-uit \n");
}
```

```
void main()
{char m='z';
 nn *nnm1,*nnm2;
 nnm1 = (nn *) calloc(1,sizeof(nn));
 nnm2 = (nn *) calloc(1,sizeof(nn));
 submenu();
 while (m!='q')
{
switch(m)
 {case '1' :  initnn(nnm1,"Neural controller 1");  submenu();  break;
  case '2' :  initnn(nnm2,"Neural controller 2");  submenu();  break;
  case 'l' :  learncontrolrb(nnm1,nnm2);submenu();break;
 };
 m = getch();
}
}
void learncontrolrb(nn *nnc1,nn *nnc2)
{
float   *dx,*x,*px,*dxm,*xm,*u,*pu,*nx ,*nxm , *zx, *zu;
int  ii,ij,j=10,i,ip=1,numpoint,plotstate=1,printstate=0;
float kp1=200.0,kp2=200.,kd=20.0;
int   num=1,ir1,ir2;
FILE  *f,*fiter1,*fiter2;
char file[30],file1[30],file2[30];
char key='z';
float merror=0,u1,u2,up1,up2,e1,buf1,buf2,buf3,buf4,buf=1,pe1=0,e2,pe2=0;
plotconfig  p;
float k1,k2,k3,k4;
float  averror2,averror1,sqerror2=0,sqerror1=0,djdu1,djdu2,dde1=0,dde2=0;
unsigned long countsample=0,numsample=0,numloop=0;
float  un1,un2,unn1,unn2,em1,pem2=0,pem1=0,em2,dem1,dem2;
float  ddr1,ddr2,dr1,dr2,rr1,rr2;
robotconfig rcm;
float mon=10.;
int value;
char string[25]=" Avail mem = ",string1[25]=" Index  =  ";
inputm = (float *)calloc(nnc1->nnode_layer[0],sizeof(float));
output = (float *)calloc(4,sizeof(float));
y      = (float *)calloc(4,sizeof(float));
zx    = initvar(6 ,sizeof(float));
zu    = initvar(3 ,sizeof(float));
dx    = initvar(6 ,sizeof(float));
dxm    = initvar(6 ,sizeof(float));
x     = initvar(6 ,sizeof(float));
px     = initvar(6 ,sizeof(float));
```

```
nx    = initvar(6 ,sizeof(float));
xm    = initvar(6 ,sizeof(float));
nxm   = initvar(6 ,sizeof(float));
u     = initvar(3 ,sizeof(float));
pu    = initvar(3 ,sizeof(float));
 printf("Enter Neural net response file =  "); scanf("%s",&file);
 f = fopen(file,"w");
 printf("Enter  iterate data file 1=  "); scanf("%s",&file1);
 fiter1 = fopen(file1,"a");
 printf("Enter  iterate data file 2=  "); scanf("%s",&file2);
 fiter2 = fopen(file2,"a");
initrobotpara(&rcm);
start : initgphc();    /*  must run initgphc  before  getmaxx(),getmaxy()  */
numpoint = 4;
setplotconfig(&p,.1,.9,-5,5);
initrobotpara(&rcm);
plotregion(p,"L-rate  M-momentum Bias_rate Sgm_rate O/F raTio Q-uit");
calparameter(x);
buf1=rcm.m2;
rcm.m2  += 0;
rcm.I2  += 0;
buf = rcm.lc2;
rcm.lc2 = (buf1*rcm.lc2+(rcm.m2-buf1))/rcm.m2;
rc =&rcm;
mon = get("vary Load = ",0,10);
for(;;numloop++)
{ numsample = 5*5*50;
for(ir1=0;ir1<=4;ir1++)          /*  1  */
{
for(ir2=0;ir2<=4;ir2++)          /*  2  */
{
   rr2 = -pi/2+2*(pi/2)/4*ir2;
    rr2 *= -1;
    rr1 = -pi/2+2*(pi/2)/4*ir1;
    rr1 *= -1;
for(i=0;i<100;i++,ip++)
{ // min change of dm2=10 -> 0 is 15 loop:change  nm.nl=6
/****************** filter  1 ***************************/
/****************** filter  2***************************/

uf_1 =  rr1 - xf1_1 ;  /*  xf1_1 or x1  */
nxf1_1 = 1*xf1_1 + dt * xf2_1;
nxf2_1 =( 1 - dt *10) * xf2_1 +  25*dt*uf_1;
```

```
uf_2 =  rr2 - xf1_2 ;  /*  xf1_2 or x2  */
nxf1_2 = 1*xf1_2 + dt * xf2_2;
nxf2_2 = ( 1 - dt *10) * xf2_2 +  25*dt*uf_2;

r1=xf1_1;
r2=xf1_2;
dr1=xf2_1;
dr2=xf2_2;
ddr1=  -10*xf2_1+25*uf_1;;
ddr2=  -10*xf2_2+25*uf_2;;
e1 = r1-*(x);
e2 = r2-*(x+2);
calparameter(x);
*u   =kp1*e1+(dr1-x[1])*kd;
*(u+1)=kp2*e2+(dr2-x[3])*kd;
buf1 =  ddr1+u[0]+ff1;
buf1 += dd1+(cc1*x[1])*x[1]+(cc2*x[3])*x[3];
buf2 =  ddr2+u[1]+ff2+dd2;
buf2 += (cc3*x[1])*x[1]+(cc4*x[3])*x[3];
k1=kk1;
k2=kk2;
k3=kk3;
k4=kk4;
*u = kk4 * buf1 - kk2 * buf2;
*u /= kk1*kk4-kk2*kk3;
*(u+1)= ( -kk3*buf1 +kk1*buf2)/(kk1*kk4-kk2*kk3);
calnextstaterk4(nxm,x,dxm,u,dt);
/************* vary m2 ******************/
buf1=rcm.m2;
rcm.m2  += mon;
rcm.I2  +=10./12.;
buf = rcm.lc2;
rcm.lc2 = buf1*rcm.lc2+rcm.m2-buf1;
rcm.lc2 /= rcm.m2;
rc =&rcm;
calnextstaterk4(nx,x,dx,u,dt);
rcm.m2  = buf1;
rcm.I2  -= 10./12.;/* 7/12to be 12/12 in I2 */
rcm.lc2 = buf ;
if( i%2 == 0 )
{ inputm[0] =  x[0];
  inputm[1] =  x[1];
  inputm[2] =  x[2];
  inputm[3] =  x[3];
  inputm[4] =  r1 -x[0];
```

```
 inputm[5] =  dr1-x[1];
 inputm[6] =  r2 -x[2];
 inputm[7] =  dr2-x[3];
 inputm[8] =  ddr1;
 inputm[9] =  ddr2;
simnn(nnc1,inputm,&un1);
 unn1 = (un1-0.5)*200.;
 simnn(nnc2,inputm,&un2);
 unn2 = (un2-0.5)*400.;
 dde1 = dxm[1]-dx[1];
 dde2 = dxm[3]-dx[3];
 buf1 = ( kk4*dde1 -kk2*dde2)/(kk1*kk4-kk2*kk3);
 buf2 = ( -kk3*dde1 +kk1*dde2)/(kk1*kk4-kk2*kk3);
 buf3 =  k1*buf1 +k2*buf2;
 buf4 =  k3*buf1 +k4*buf2;
  buf1 =buf3/200.+0.5;
buf1=limitb(0.1,0.9,buf1);
  buf2 =buf4/400.+0.5;
buf2=limitb(0.1,0.9,buf2);
 if(numloop%2 != 0)
 {
  weightadj(nnc1,&buf1);
  weightadj(nnc2,&buf2);
 };
if(numloop%2 == 0)
 {
  sqerror1 += fabs((double)(buf1-un1));
  sqerror2 += fabs((double)(buf2-un2));
  countsample++;
 if(countsample==numsample)
  {averror1 =sqerror1/((float) numsample);
   averror2 =sqerror2/((float) numsample);
     sqerror1=sqerror2=countsample=0;
   fprintf(fiter1," %ld  %f \n",numloop,averror1);
   fprintf(fiter2," %ld  %f \n",numloop,averror2);
   if ((numloop%100)==0)
    {autosavenn(nnc1,"s11.50");
     autosavenn(nnc1,"ss11.50");
     autosavenn(nnc2,"s22.50");
     autosavenn(nnc2,"ss22.50");
    };

  };
  };
```

```
y[0] = buf3/20.;
y[1] = buf4/40.;
output[0] =unn1/20.;
output[1] =unn2/40.;
y[2] = dde1/10;
y[3] = dde2/10;
output[2] = r1;
output[3] =  r2;
if(plotstate==1)
  {scaledata(output,y,numpoint,p);
   plotpoint(output,y,numpoint,ip,p);
  };
};//end if(i%2==0)
//fprintf(f,"%f %f %f %f %f %f \n",x[0],x[2],r1,r2,e1,e2);
  switch(key)
  {  case 'l': getrate("L_rate",nnc1->layer,nnc1->lrate);
               getrate("L_rate",nnc2->layer,nnc2->lrate); break;
     case 'b': getrate("Bias_rate",nnc1->layer,nnc1->lbias_rate);
               getrate("Bias_rate",nnc2->layer,nnc2->lbias_rate); break;
     case 'm': getrate("Momentum_rate",nnc1->layer,nnc1->lmomentum_rate);
               getrate("Momentum_rate",nnc2->layer,nnc2->lmomentum_rate);
               break;
     case 'a': getrate("Sigmoid_rate",nnc1->layer,nnc1->lsgm_rate);
               getrate("Sigmoid_rate",nnc2->layer,nnc2->lsgm_rate);
               break;
     case 'h': value = farcoreleft();ltoa(value,string+13,10);
               setfillstyle(SOLID_LINE,BLACK);
               bar(0,10,800,10+8);
               outtextxy(0,10,string);
               break;
     case 'i': ltoa(nm.index,string1+13,10);
               setfillstyle(SOLID_LINE,BLACK);
               bar(0,10,800,10+8);
               outtextxy(0,10,string1);
               break;
     case 'p'      : mon = get("vary Load = ",0,10);
     case 'o'    : plotstate = 1;break;
     case 'd'    : printstate =1; break;
     case '\021' : key='q';   break;
     case 'f'    : plotstate = 0;break;
    };
```

```
if(ip == (int) (p.maxx-p.minx))
  {  if(printstate==1)
     {DumpScr(); printstate =0;
     };
    setfillstyle(SOLID_FILL,0);
    bar(p.minx-5+1,p.miny-5+1,p.maxx+5-1,p.maxy+5-1);
    ip=1;
  };
key = chkbioskey();
if(key=='q')  goto end;
if(i==50)  rr2=0 ;
   xf1_1 = nxf1_1 ;
   xf2_1 = nxf2_1 ;
   xf1_2 = nxf1_2 ;
   xf2_2 = nxf2_2 ;
   tranfer(nx ,x  ,6);
   tranfer(nxm ,xm  ,6);
};/*ip*/
/******** RESET STATE *************/
   tranfer(zx,x,6);
   tranfer(zx,xm,6);
   xf1_1 = 0;
   xf2_1 = 0;
   xf1_2 = 0;
   xf2_2 = 0;
};
};
}
end :closegraph();
clrscr();
printf("Learning again y/n");key=getch();
if(key == 'y') goto start;
fclose(fiter1);
fclose(fiter2);
fclose(f);
return;
}
```

2. ADAPTNN.CPP

```
#include "main.h"
#include "nn.h"
#include "learnnn.h"
#include "simrb.h"
#include "plot.h"
#include "simnn.h"
void submenu();
void learncontrolrb(nn *nnm1,nn *nnm2);
void contrlweightadj(nn *n,float *djdy);
void tranferinitweight(nn *n);
void initweight(nn *n);
void initrule(void);
void maprule(nn *nnc1,nn *nnc2);
int setofvariable(float e,float a,float b);
float a11,a12,a21,a22,b11,b12,b21,b22,rule1[5][5],rule2[5][5];
int LN=0,MN=1,Z=2,MP=3,LP=4;
float ns1,nc1,nm1,nb1,nl1,ns2,nc2,nm2,nb2,nl2;
float r1=0,r2=0;
float *y,*inputm,*output;
float xf1_1=0,xf2_1=0,nxf1_1,nxf2_1,uf_1=0;
float xf1_2=0,xf2_2=0,nxf1_2,nxf2_2,uf_2=0;
float  em1,pem2=0,pem1=0,em2,dem1,dem2;

void submenu()
{clrscr();
printf("1- about NNC1 \n");
printf("2- about NNC2 \n");
printf("L- learning  \n");
printf("Q-uit \n");
}

void main()
{char m='z';
 nn *nnm1,*nnm2;
 nnm1 = (nn *) calloc(1,sizeof(nn));
 nnm2 = (nn *) calloc(1,sizeof(nn));
 dt = 0.05;//.025
 submenu();
 while (m!='q')
{switch(m)
 {case '1' :  initnn(nnm1,"Neural controller 1");  submenu();  break;
  case '2' :  initnn(nnm2,"Neural controller 2");  submenu();  break;
  case 'l' :  learncontrolrb(nnm1,nnm2);submenu();break;
 };
```

```
m = getch();
}
}

void learncontrolrb(nn *nnc1,nn *nnc2)
{float  *dx,*x,*dxm,*xm,*u,*nx ,*nxm , *zx, *zu;
int  ij,i,ip=1,numpoint,plotstate=1,printstate=0;
float kp1=200.0,kp2=200.0,kd=20.0;
int  j,ir1,ir2;
FILE  *f;
char file[30];
char key='z';
float u1,u2,e1,buf1,buf2,buf3,buf4,buf=1,pe1=0,e2,pe2=0;
plotconfig  p;
float  djdu1,djdu2;
float  unn1,unn2;
float  tsine,ddr1,ddr2,dr1,dr2,rr1,rr2;
robotconfig rcm;
float mon=10.;
inputm = (float *)calloc(nnc1->nnode_layer[0],sizeof(float));
output = (float *)calloc(4,sizeof(float));
y     = (float *)calloc(4,sizeof(float));
zx    = initvar(6 ,sizeof(float));
zu    = initvar(3 ,sizeof(float));
dx    = initvar(6 ,sizeof(float));
dxm    = initvar(6 ,sizeof(float));
x     = initvar(6 ,sizeof(float));
nx    = initvar(6 ,sizeof(float));
xm     = initvar(6 ,sizeof(float));
nxm    = initvar(6 ,sizeof(float));
u     = initvar(3 ,sizeof(float));
 printf("Enter Neural net response file =  "); scanf("%s",&file);
 f = fopen(file,"w");
initrobotpara(&rcm);
start : initgphc();    /*  must run initgphc  before  getmaxx(),getmaxy()  */
numpoint = 4;
setplotconfig(&p,.1,.9,-5,5);
initrobotpara(&rcm);
rc=&rcm;
initrule();
plotregion(p,"L-rate  M-momentum Bias_rate Sgm_rate O/F raTio Q-uit");
calparameter(x);
buf1=rcm.m2;
rcm.m2  += 0;
rcm.I2  += 0;
```

```
buf = rcm.lc2;
rcm.lc2 = (buf1*rcm.lc2+(rcm.m2-buf1))/rcm.m2;
rc =&rcm;
nnc1->lrate[nnc1->layer-1]=0.01;
nnc2->lrate[nnc2->layer-1]=0.01;
mon = get("vary Load = ",0,10);
 initweight(nnc1);
 initweight(nnc2);
for(j=0;j<5;j++)
{for(ir1=0;ir1<=1;ir1++)          /*  1  */
{for(ir2=0;ir2<=3;ir2++)          /*  2  */
{   rr2= -pi/2+2*pi/2/4*ir2;rr2*=-1;
    rr1= -pi/2+2*pi/2/4*ir1;rr1*=-1;
for(i=0;i<150;i++)
{/****************** filter  1 & filter  2**************************/
uf_1 =  rr1 - xf1_1 ;  /*  xf1_1 or x1  */
nxf1_1 = 1*xf1_1 + dt * xf2_1;
nxf2_1 =( 1 - dt *10) * xf2_1 +  25*dt*uf_1;
uf_2 =  rr2 - xf1_2 ;  /*  xf1_2 or x2  */
nxf1_2 = 1*xf1_2 + dt * xf2_2;
nxf2_2 = ( 1 - dt *10) * xf2_2 +  25*dt*uf_2;
r1=xf1_1;
r2=xf1_2;
dr1=xf2_1;
dr2=xf2_2;
ddr1=  -10*xf2_1+25*uf_1;;
ddr2=  -10*xf2_2+25*uf_2;;
/*
 tsine=100;
    r1  = sin(2*pi*i/tsine);
 tsine=100;
    r2  = cos(2*pi*i/tsine);
 tsine=100;
 buf =  2*pi/tsine/dt;
    dr1 = buf*cos(2*pi*i/tsine);
    ddr1= -buf*buf*sin(2*pi*i/tsine);
 tsine=100;
 buf =  2*pi/tsine/dt;
    dr2 = -buf*sin(2*pi*i/tsine);
    ddr2= -buf*buf*cos(2*pi*i/tsine);
*/
e1 = r1-*(x);
e2 = r2-*(x+2);
calparameter(x);
*u   =kp1*e1+(dr1-x[1])*kd;
```

```
*(u+1)=kp2*e2+(dr2-x[3])*kd;
u1 =  ddr1+  u[0]   + ff1 + dd1
    +  cc1*x[1]*x[1]+ cc2*x[3]*x[3];
u2 =  ddr2+  u[1]  + ff2 + dd2
    + cc3*x[1]*x[1]+ cc4*x[3]*x[3]; //dd2 mistake  before
u[0] = ( kk4*u1 -kk2*u2)/(kk1*kk4-kk2*kk3);
u[1] = ( -kk3*u1 +kk1*u2)/(kk1*kk4-kk2*kk3);
calnextstaterk4(nxm,x,dxm,u,dt);
inputm[0] =  x[0];
inputm[1] =  x[1];
inputm[2] =  x[2];
inputm[3] =  x[3];
inputm[4] =  r1 -x[0];
inputm[5] =  dr1-x[1];
inputm[6] =  r2 -x[2];
inputm[7] =  dr2-x[3];
inputm[8] =  ddr1;
inputm[9] =  ddr2;
 em1 = xm[0]-x[0];
 dem1 = em1-pem1;
 em2 = xm[2]-x[2];
 dem2 = em2-pem2;
   simnn(nnc1,inputm,&unn1);
   unn1 = (unn1-0.5)*100.;
   u1 +=unn1;
   simnn(nnc2,inputm,&unn2);
  unn2 = (unn2-0.5)*100.;
 u2 += unn2;
 u[0] =( kk4*u1 -kk2*u2)/(kk1*kk4-kk2*kk3);
 u[1] =( -kk3*u1 +kk1*u2)/(kk1*kk4-kk2*kk3);
/************* vary m2 *****************/
 buf1=rcm.m2;
 rcm.m2  += mon;
 rcm.I2  +=10./12.;
 buf = rcm.lc2;
 rcm.lc2 = (buf1*rcm.lc2+(rcm.m2-buf1))/rcm.m2;
 rc =&rcm;
 calnextstaterk4(nx,x,dx,u,dt);
 rcm.m2  = buf1;
 rcm.I2  -= 10./12.;/* 7/12to be 12/12 in I2 */
 rcm.lc2 = buf ;
 djdu1 = -em1;
 djdu2 = -em2;
 if(i%10==0)//10 Good
  maprule(nnc1,nnc2);
```

```
contrlweightadj(nnc1,&djdu1);
contrlweightadj(nnc2,&djdu2);
//fprintf(f,"%f %f %f %f %f %f \n",x[0],x[2],r1,r2,e1,e2);
 y[0] = em1*100;
 y[1] = em2*100;
 output[0] =dem1*100*0;
 output[1] =dem2*100*0;
 y[2] = x[0];
 y[3] = x[2];
 output[2] = r1;
 output[3] = r2;
  ip++;
  switch(key)
  {  case 'l': getrate("L_rate",nnc1->layer,nnc1->lrate);
               getrate("L_rate",nnc2->layer,nnc2->lrate); break;
     case 'b': getrate("Bias_rate",nnc1->layer,nnc1->lbias_rate);
               getrate("Bias_rate",nnc2->layer,nnc2->lbias_rate); break;
     case 'm': getrate("Momentum_rate",nnc1->layer,nnc1->lmomentum_rate);
               getrate("Momentum_rate",nnc2->layer,nnc2->lmomentum_rate);
               break;
     case 'a': getrate("Sigmoid_rate",nnc1->layer,nnc1->lsgm_rate);
               getrate("Sigmoid_rate",nnc2->layer,nnc2->lsgm_rate);
               break;
     case 'p'      : mon = get("vary Load = ",0,10);
     case 'o'    : plotstate = 1;break;
     case 'd'    : printstate =1; break;
     case '\021' : key='q';   break;
     case 'f'    : plotstate = 0;break;
   };
 if(ip == (int) (p.maxx-p.minx))
    {  if(printstate==1)
       {DumpScr(); printstate =0;
       };
       setfillstyle(SOLID_FILL,0);
       bar(p.minx-5+1,p.miny-5+1,p.maxx+5-1,p.maxy+5-1);
       ip=1;
    };
 if(plotstate==1)
    {scaledata(output,y,numpoint,p);
     plotpoint(output,y,numpoint,ip,p);
    };
 key = chkbioskey();
 if(i==75) rr1=rr2=0;
  if(key=='q')   goto end;
     xf1_1 = nxf1_1 ;
```

```
    xf2_1 = nxf2_1 ;
    xf1_2 = nxf1_2 ;
    xf2_2 = nxf2_2 ;
    pem1=em1;
    pem2=em2;
    tranfer(nx ,x  ,6);
    tranfer(nxm ,xm  ,6);
};/*ip*/
/******** RESET STATE **********/
  tranferinitweight(nnc1);
  tranferinitweight(nnc2);
    tranfer(zx,x,6);
    tranfer(zx,xm,6);
    pem1=0;
    pem2=0;
    xf1_1 = 0;
    xf2_1 = 0;
    xf1_2 = 0;
    xf2_2 = 0;
};
};
};
end :closegraph();
clrscr();
fclose(f);
printf("Learning again y/n");key=getch();
if(key == 'y') goto start;
return;
}

void contrlweightadj(nn *n,float *djdy)
{int layer,ii=0,i=0,j=0,l=0;
 int *nnodelayer;
 float  **wbias,**pwbias,**a,**pa,***w,***pw;
 float *lr,*lm,*lb,*la,djdw=0,djda=0,momentum=0;
 nnodelayer= n->nnode_layer;
 layer    = n->layer;
 lr       = n->lrate;
 lb       = n->lbias_rate;
 lm       = n->lmomentum_rate;
 la       = n->lsgm_rate;
 w      = n->w;
 wbias  = n->wbias;
 pwbias = n->pwbias;
 pw = n->pw;
```

```
a  = n->a;
pa = n->pa;
simdykbydyi(n);
/***** Adjust Weight *****/        //******* only layer-1
//for (l=1;l<layer;l++)
for (l=layer-1;l<layer;l++)
 for (i=0;i<nnodelayer[l-1];i++)
  for (j=0;j<nnodelayer[l];j++)
    { djdw=0;
     for(ii=0; ii< nnodelayer[layer-1] ;ii++)
         {
         djdw = djdy[ii] * dykdwij(n,layer-1,ii,l,i,j);
         };
         momentum = lm[l] * (w[l][i][j] - pw[l][i][j] );
         pw[l][i][j] = w[l][i][j];
         w[l][i][j] += -djdw*lr[l]+momentum;
    };
/***** Adjust Bias Weight *****/    //******* only layer-1
//for (l=1;l<layer;l++)
for (l=layer-1;l<layer;l++)
 for (i=0;i<nnodelayer[l];i++)
    { djdw=0;
     for(ii=0;ii<nnodelayer[layer-1];ii++)
         { djdw = djdy[ii] * dykdwbiasi(n,layer-1,ii,l,i);
         };
         momentum     = lm[l] * (wbias[l][i] - pwbias[l][i] );
         pwbias[l][i]  = wbias[l][i];
         wbias[l][i]  += -djdw*lb[l]+momentum;
    };
/***** Adjust Sigmoid Shape *****/
//for (l=1;l<layer;l++)
for (l=layer-1;l<layer;l++)             //******* only layer-1
 for (i=0;i<nnodelayer[l];i++)
    { djda=0;
     for(ii=0;ii<nnodelayer[layer-1];ii++)
         { djda =  djdy[ii] * dykdai(n,layer-1,ii,l,i);
         };
         momentum  = lm[l] * (a[l][i] - pa[l][i] );
         pa[l][i]  = a[l][i];
         a[l][i]  += -djda*la[l]+momentum;
    };
return;
}
```

```
void tranferinitweight(nn *n)
{int  i,j,*nnodelayer,dummy1,dummy2;
float ***w,***initw,***pw;
nnodelayer =  n->nnode_layer;
w =  n->w;
pw =  n->pw;
initw =  n->initw;
dummy1 =n->layer-1-1;
dummy2 =n->layer-1;
 for(i=0;i<nnodelayer[dummy1];i++)
  {for(j=0;j<nnodelayer[dummy2];j++)
   { w[dummy2][i][j] = pw[dummy2][i][j]= initw[dummy2][i][j] ;
   };
  };
return;
}

void initweight(nn *n)
{int  i,j,*nnodelayer,dummy1,dummy2;
float ***w,***initw,***pw;
float **pwbias,**wbias,**initwbias;
wbias = n->wbias;
initwbias = n->initwbias;
pwbias = n->pwbias;
nnodelayer =  n->nnode_layer;
w =  n->w;
pw =  n->pw;
initw =  n->initw;
dummy1 =n->layer-1-1;
dummy2 =n->layer-1;
 for(i=0;i<nnodelayer[dummy1];i++)
  {for(j=0;j<nnodelayer[dummy2];j++)
   {  pw[dummy2][i][j] = w[dummy2][i][j]=  initw[dummy2][i][j]
   =  0.5*initw[dummy2][i][j] ;  //0.5good ,0.7osscil in em
     //****************set value ==0 if !=0 at dm3==0 it cause osscilation
   };
  };
 for(i=0;i<nnodelayer[dummy2];i++)
  { pwbias[dummy2][i]  = initwbias[dummy2][i]
   =wbias[dummy2][i]= 1*initwbias[dummy2][i] ;
   //fixed ==1.0
  };
 return;
 }
```

```
void initrule()
{
a11=.1;  b11= 2.5;
a12=0.1; b12=2.5;
a21=.1;  b21=2.5;
a22=0.1; b22=2.5;
ns1=1.;nc1=1.25;nm1=1.5;nb1=1.75;nl1=2.;
ns2=.5;nc2=.75;nm2=1.;nb2=1.25;nl2=1.5;
//**************************RULE 1
rule1[LN][LN]=nl1 ; rule1[LN][MN]=nl1 ;
rule1[LN][Z] =nl1 ; rule1[LN][MP]=nm1 ; rule1[LN][LP]= ns1;

rule1[MN][LN]=nl1 ; rule1[MN][MN]=nb1 ;
rule1[MN][Z] =nm1 ; rule1[MN][MP]=nc1 ; rule1[MN][LP]= ns1;

rule1[Z][LN]= ns1 ; rule1[Z][MN]= ns1 ;
rule1[Z][Z] = ns1 ; rule1[Z][MP]= ns1 ; rule1[Z][LP] = ns1;

rule1[MP][LN]=ns1 ; rule1[MP][MN]=nc1 ;
rule1[MP][Z]= nm1 ; rule1[MP][MP]=nb1 ; rule1[MP][LP]= nl1;

rule1[LP][LN]=ns1 ; rule1[LP][MN]=nm1 ;
rule1[LP][Z] =nl1 ; rule1[LP][MP]=nl1 ; rule1[LP][LP]= nl1;
//**************************RULE 2
rule2[LN][LN]=nl2 ; rule2[LN][MN]=nl2 ;
rule2[LN][Z] =nl2 ; rule2[LN][MP]=nm2 ; rule2[LN][LP]= ns2;

rule2[MN][LN]=nl2 ; rule2[MN][MN]=nb2 ;
rule2[MN][Z] =nm2 ; rule2[MN][MP]=nc2 ; rule2[MN][LP]= ns2;

rule2[Z][LN]= ns2 ; rule2[Z][MN]= ns2 ;
rule2[Z][Z] = ns2 ; rule2[Z][MP]= ns2 ; rule2[Z][LP] = ns2;

rule2[MP][LN]=ns2 ; rule2[MP][MN]=nc2 ;
rule2[MP][Z]= nm2 ; rule2[MP][MP]=nb2 ; rule2[MP][LP]= nl2;

rule2[LP][LN]=ns2 ; rule2[LP][MN]=nm2 ;
rule2[LP][Z] =nl2 ; rule2[LP][MP]=nl2 ; rule2[LP][LP]= nl2;

return;
}
```

```
void maprule(nn *nnc1,nn *nnc2)
{int setem1,setem2,setdem1,setdem2;
 setem1=setofvariable(em1*100,a11,b11);
 setem2=setofvariable(em2*100,a12,b12);
 setdem1=setofvariable(dem1*100,a21,b21);
 setdem2=setofvariable(dem2*100,a22,b22);
 nnc1->lrate[nnc1->layer-1]= rule1[setem1][setdem1];
 nnc2->lrate[nnc2->layer-1]= rule2[setem2][setdem2];
 return;
}

int setofvariable(float e,float a,float b)
{if(e<=0)
  { if((e>=-a)&&(e<=a))  return(Z);
    if((e>=-b)&&(e<-a)) return(MN);
    if(e<-b)          return(LN);
     }else
  {if((e>=-a)&&(e<=a))  return(Z);
    if((e>a)&&(e<=b))   return(MP);
    if(e>b)           return(LP);
  };
return(NULL);
}
```

3. SIMRB. H และ SIMRB.CPP

3.1 SIMRB. H

```
typedef struct { float   I1,  I2,  l1, l2, lc1 , lc2, m1, m2 ;
                 float   k1, k2,g;
               } robotconfig;
/******** simrb.cpp ************/
extern void    tranfer(float *destnationvar,float *targetvar ,int num) ;
extern float   *initvar(int num,int sizeofvar)        ;
extern void    initrobotpara(robotconfig *rc)                  ;
extern void calnextstaterk4(float *nx,float *x,float *dx,float *u,float h);
extern void caldx(float *dx,float *x,float *u);
extern void calparameter(float *x);
/********  Global Var ************/
extern float   pi,dt;
extern robotconfig *rc;
extern float ff1,ff2,kk1,kk2,kk3,kk4;
extern float cc1,cc2,cc3,cc4,dd1,dd2;
```

3.2 SIMRB. CPP

```
#include "main.h"
#include "nn.h"
#include "simrb.h"
void tranfer(float *destinationvar,float *targetvar ,int num);
void initrobotpara(robotconfig *rbc);
void calnextstaterk4(float *nx,float *x,float *dx,float *u,float h);
void caldx(float *dx,float *x,float *u);
void calparameter(float *x);
float pi=22./7.;
robotconfig  *rc;
float ff1,ff2,kk1,kk2,kk3,kk4;
float cc1,cc2,cc3,cc4,dd1,dd2;
float j1;
void tranfer(float *destinationvar,float *targetvar ,int num)
{int i;
for(i=0;i<num;i++)
  {*(targetvar+i)  =  *(destinationvar+i);
  };
return;
}

void calnextstaterk4(float *nx,float *x,float *dx,float *u,float h)
{ float *kt1,*kt2,*kt3,*kt4,*bx,*bdx;
kt1 =(float *)calloc(6,sizeof(float));
kt2 =(float *)calloc(6,sizeof(float));
kt3 =(float *)calloc(6,sizeof(float));
kt4 =(float *)calloc(6,sizeof(float));
bx  =(float *)calloc(6,sizeof(float));
bdx =(float *)calloc(6,sizeof(float));
tranfer(x,bx,6);
/* cal k1 */
caldx(bdx,bx,u);
tranfer(bdx,dx,6);
*(kt1+0)=h**(bdx+0);
*(kt1+1)=h**(bdx+1);
*(kt1+2)=h**(bdx+2);
*(kt1+3)=h**(bdx+3);
*(kt1+4)=h**(bdx+4);
*(kt1+5)=h**(bdx+5);
/* cal k2 */
*(bx+0) = *(x+0)  + 0.5* *(kt1+0);
*(bx+1) = *(x+1)  + 0.5* *(kt1+1);
*(bx+2) = *(x+2)  + 0.5* *(kt1+2);
*(bx+3) = *(x+3)  + 0.5* *(kt1+3);
```

```
*(bx+4) = *(x+4)  + 0.5* *(kt1+4);
*(bx+5) = *(x+5)  + 0.5* *(kt1+5);
caldx(bdx,bx,u);
*(kt2+0)=h**(bdx+0);
*(kt2+1)=h**(bdx+1);
*(kt2+2)=h**(bdx+2);
*(kt2+3)=h**(bdx+3);
*(kt2+4)=h**(bdx+4);
*(kt2+5)=h**(bdx+5);
/* cal k3 */
*(bx+0) = *(x+0)  + 0.5* *(kt2+0);
*(bx+1) = *(x+1)  + 0.5* *(kt2+1);
*(bx+2) = *(x+2)  + 0.5* *(kt2+2);
*(bx+3) = *(x+3)  + 0.5* *(kt2+3);
*(bx+4) = *(x+4)  + 0.5* *(kt2+4);
*(bx+5) = *(x+5)  + 0.5* *(kt2+5);
caldx(bdx,bx,u);
*(kt3+0)=h**(bdx+0);
*(kt3+1)=h**(bdx+1);
*(kt3+2)=h**(bdx+2);
*(kt3+3)=h**(bdx+3);
*(kt3+4)=h**(bdx+4);
*(kt3+5)=h**(bdx+5);
/* cal k4 */
*(bx+0) = *(x+0)  + *(kt3+0);
*(bx+1) = *(x+1)  + *(kt3+1);
*(bx+2) = *(x+2)  + *(kt3+2);
*(bx+3) = *(x+3)  + *(kt3+3);
*(bx+4) = *(x+4)  + *(kt3+4);
*(bx+5) = *(x+5)  + *(kt3+5);
caldx(bdx,bx,u);
*(kt4+0)=h**(bdx+0);
*(kt4+1)=h**(bdx+1);
*(kt4+2)=h**(bdx+2);
*(kt4+3)=h**(bdx+3);
*(kt4+4)=h**(bdx+4);
*(kt4+5)=h**(bdx+5);

*(nx+0)=*(x+0) + 1./6.*( *(kt1+0) +2.**(kt2+0) +2.**(kt3+0) +*(kt4+0));
*(nx+1)=*(x+1) + 1./6.*( *(kt1+1) +2.**(kt2+1) +2.**(kt3+1) +*(kt4+1));
*(nx+2)=*(x+2) + 1./6.*( *(kt1+2) +2.**(kt2+2) +2.**(kt3+2) +*(kt4+2));
*(nx+3)=*(x+3) + 1./6.*( *(kt1+3) +2.**(kt2+3) +2.**(kt3+3) +*(kt4+3));
*(nx+4)=*(x+4) + 1./6.*( *(kt1+4) +2.**(kt2+4) +2.**(kt3+4) +*(kt4+4));
*(nx+5)=*(x+5) + 1./6.*( *(kt1+5) +2.**(kt2+5) +2.**(kt3+5) +*(kt4+5));
```

```
free(kt1);free(kt2);free(kt3);free(kt4);
free(bx);free(bdx);
return;
}

void caldx(float *dx,float *x,float *u)
{ float lc1,lc2,l1,m1,m2,I1,I2,g;
float a1,a2,a3,a4,aa1,aa2,aa3,aa4;
float c1,c2;
float d1;
float f1,f2;
float k1,k2;
float buf;
double x1,x2;
float o1,o2,o3,o4,o5,o6;
x1=(double)(*(x));
x2=(double)(*(x+2));
lc1  = rc->lc1  ;
lc2  =  rc->lc2 ;
l1  =  rc->l1 ;
m1  =  rc->m1 ;
m2  =  rc->m2 ;
I1  =  rc->I1 ;
I2  =  rc->I2 ;
k1  =  rc->k1 ;
k2  =  rc->k2 ;
g  =  rc->g  ;
o1 =m1*lc1*lc1 + m2*l1*l1 + I1 ;
o2 =m2*lc2*lc2 + I2 ;
o3 =m2*l1*lc2;
o4 =m1*lc1;
o5 =m2*l1;
o6 =m2*lc2;
a1 = o1+o2+2*o3*(float)cos(x2);
a2 = a3 = o2+(float)cos(x2)*o3;
a4 = o2;
c1 = -(float)sin(x2)*o3;
c2 = -c1;
d1 = -2*(float)sin(x2)*o3*x[1]*x[3];
f1 = (o4+o5)*g*(float)cos(x1)+o6*g*(float)cos(x1+x2);
f2 = o6*g*(float)cos(x1+x2);
buf = (a1*a4-a2*a3);
aa1 = a4/buf;
aa2 = aa3 = -a3/buf;
aa4 = a1/buf;
```

```
cc1 = aa2*c2;
cc2 = aa1*c1;
cc3 = aa4*c2;
cc4 = aa3*c1;
dd1 = aa1*d1;
dd2 = aa3*d1;
ff1 = aa1*f1 +aa2*f2;
ff2 = aa3*f1 +aa4*f2;
kk1 = aa1*k1;
kk2 = aa2*k2;
kk3 = aa3*k1;
kk4 = aa4*k2;
*(dx+0) = *(x+1);
*(dx+1) = -(cc1*x[1]*x[1]+cc2*x[3]*x[3]) -dd1 - ff1+ kk1**(u+0) +
kk2**(u+1);
*(dx+2) = *(x+3);
*(dx+3) = -(cc3*x[1]*x[1]+cc4*x[3]*x[3]) - dd2 -ff2+ kk3**(u+0) +
kk4**(u+1);
return;
}

float *initvar(int num,int sizeofvar)
{float *var;
 var=(float *)calloc(num,sizeofvar);
 return(var);
}

void initrobotpara(robotconfig *rc)
{ rc->l1  = 1;
 rc->l2  = 1;
 rc->lc1  = 0.5;
 rc->lc2  = 0.5;
 rc->I1  = 10./12.;
 rc->I2  = 5./12.;
 rc->m1  = 10;
 rc->m2  = 5;
 rc->k1  = 40.;
 rc->k2  = 20.;
 rc->g   = 9.81;
}
```

```
void calparameter(float *x)
{ float lc1,lc2,l1,
m1,m2,I1,I2,g;
float a1,a2,a3,a4,aa1,aa2,aa3,aa4;
float c1,c2;
float d1;
float f1,f2;
float k1,k2;
float buf;
double x1,x2;
float o1,o2,o3,o4,o5,o6;
x1=(double)(*(x));
x2=(double)(*(x+2));
lc1 =rc->lc1 ;
lc2 = rc->lc2 ;
l1 = rc->l1 ;
m1 = rc->m1 ;
m2 = rc->m2 ;
I1 = rc->I1 ;
I2 = rc->I2 ;
k1 = rc->k1 ;
k2 = rc->k2 ;
g = rc->g ;
o1 =m1*lc1*lc1 + m2*l1*l1 + I1 ;
o2 =m2*lc2*lc2 + I2 ;
o3 =m2*l1*lc2;
o4 =m1*lc1;
o5 =m2*l1;
o6 =m2*lc2;
a1 = o1+o2+2*o3*(float)cos(x2);
a2 = a3 = o2+(float)cos(x2)*o3;
a4 = o2;
c1 = -(float)sin(x2)*o3;
c2 = -c1;
d1 = -2*(float)sin(x2)*o3*x[1]*x[3];
f1 = (o4+o5)*g*(float)cos(x1)+o6*g*(float)cos(x1+x2);
f2 = o6*g*(float)cos(x1+x2);
buf = (a1*a4-a2*a3);
aa1 = a4/buf;
aa2 = aa3 = -a3/buf;
aa4 = a1/buf;
cc1 = aa2*c2;
cc2 = aa1*c1;
cc3 = aa4*c2;
cc4 = aa3*c1;
```

```
dd1 = aa1*d1;
dd2 = aa3*d1;
ff1 = aa1*f1 +aa2*f2;
ff2 = aa3*f1 +aa4*f2;
kk1 = aa1*k1;
kk2 = aa2*k2;
kk3 = aa3*k1;
kk4 = aa4*k2;
return;
}
```

ประวัติผู้เขียน

นาย มนชัย อัศวรุ่งเรืองโชติ เกิดเมื่อวันที่ 8 กุมภาพันธ์ พ.ศ. 2513 ที่กรุงเทพมหานคร สำเร็จปริญญาวิศวกรรมศาสตร์บัณฑิต จาก จุฬาลงกรณ์มหาวิทยาลัย เมื่อปี พ.ศ. 2533 หลังจากนั้นได้เข้าศึกษาต่อในระดับปริญญาโทในภาควิชาวิศวกรรม ไฟฟ้า สาขาระบบควบคุม ระหว่างปีการศึกษา พ.ศ. 2533 ถึง พ.ศ. 2535 ได้ทำหน้าที่เป็นผู้ช่วยสอนของห้องปฏิบัติการระบบควบคุม ภาควิชาวิศวกรรมไฟฟ้า