```
/* Code for Adsorption Modeling using Radial Basis Function Networks of ANN */
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<conio.h>
#define NORM(x1,x2) (x1-x2)*(x1-x2)
#define MAX 7
#define C_MAX 9
#define ETA 0.2
#define YES 1
#define NO 0
#define NORMAL(a,b,c) (a-c)/(b-c)

int continue_func(void);
static int pol,mat;
static float ph,eq_time,cont_time,ad_conc1,ad_conc2;
static float min_pol,min_mat,min_ph,min_eq_time,min_cont_time,min_ad_conc1, min_ad_conc2;
static float max_pol,max_mat,max_ph,max_eq_time,max_cont_time,max_ad_conc1, max_ad_conc2;
static float n_pol,n_mat,n_ph,n_eq_time,n_cont_time,n_ad_conc1,n_ad_conc2;
FILE *ftest;

main()
{
   int i,j,k,z,min_index,p,m,n,a,flag;
   int wctr;
   static float bwtemp[C_MAX][C_MAX],bw[C_MAX][C_MAX]; //individual bandwidths
   float radius,var;      //radius,and variance
   static float rsum;    //sum of all the radius
   float  Act_func[C_MAX],Tot_temp[C_MAX],Tot_norm[C_MAX]; // activation functions,individual nor
ms wrt each centre
   static float weight[C_MAX];
   float tmp=0.0;
   float in[MAX],dout,cout,out_diff,prevout_diff;      //input array,desired and current output
   float tempdiff[C_MAX];          //diff between the input and the centre
     // static float err[MAX][MAX];   //error between the two centres
    typedef struct cent_alloc{
    float sum[MAX];
    int count;
    } ncent;
   int cont = YES;
   FILE *fp1,*fp2;
   static ncent new_cent[C_MAX];
   float error=0.0;
   float input[7];
   //the nine arbitrary centres
   float cent[C_MAX][MAX] = { {0.0,0.0,0.0,0.0,0.0,0.0,0.0 },
                  {0.1,0.1,0.1,0.1,0.1,0.1,0.1 },
                  {0.2,0.2,0.2,0.2,0.2,0.2,0.2},
                  {0.3,0.3,0.3,0.3,0.3,0.3,0.3},
                  {0.5,0.5,0.5,0.5,0.5,0.5,0.5},
                  {0.6,0.6,0.6,0.6,0.6,0.6,0.6},
                  {0.7,0.7,0.7,0.7,0.7,0.7,0.7},
                  {0.85,0.85,0.85,0.85,0.85,0.85,0.85},
                  {1.0,1.0,1.0,1.0,1.0,1.0,1.0}
               };
   float perror=999999;
  // FILE *fout=fopen("outfile","w");
  // FILE *fcent = fopen("centfile","w");
   for(n=0;n<C_MAX;n++) {
      weight[n] = rand()%10;
    //printf("wt: %f",weight[n]);
    }

   clrscr();

  //systemclr();
// int i,j;

  clrscr();
  //systemclr();
 if ((ftest = fopen("test.dat", "w")) == NULL)
{
   fprintf(stderr, "Cannot open input file.\n");
   exit(1);
}
if(cont == YES){
```

```
for(i=0;i<10;i++)
 printf(" ");
for(i=0;i<60;i++)
 printf("*");
for(i=0;i<10;i++)
 printf(" ");
printf("\n");
for(i=0;i<34;i++)
 printf(" ");
printf("WELCOME\n");
for(i=0;i<34;i++)
 printf(" ");
printf("-------\n\n");
for(i=0;i<10;i++)
 printf(" ");
printf("PROJECT TITLE:");
printf(" ARTIFICIAL NEURAL NETWORKS FOR MODELLING\n");
for(i=0;i<25;i++)
 printf(" ");
printf("THE REMOVAL OF METALS FROM CONTAMINATED WATER\n\n");
for(i=0;i<35;i++)
 printf(" ");
printf(" By \n\n");
for(i=0;i<25;i++)
 printf(" ");
printf("K.KALYAN CHAKRAVARTHI  96A1PS270\n\n");
for(i=0;i<30;i++)
 printf(" ");
printf("Under the guidance of \n\n");
for(i=0;i<33;i++)
 printf(" ");
printf("DR.B.V.BABU\n\n");
for(i=0;i<10;i++)
 printf(" ");
for(i=0;i<60;i++)
 printf("*");
for(i=0;i<10;i++)
 printf(" ");
printf("\n");

cont = continue_func();
}
while(cont == YES)
{
clrscr();
if ((ftest = fopen("test.dat", "w")) == NULL)
  {
   fprintf(stderr, "Cannot open input file.\n");
   exit(1);
  }
pol =0;
mat =0;
ph =0.0;
eq_time = 0.0;
cont_time = 0.0;
ad_conc1 =0.0;
ad_conc2 =0.0;

printf("\n\n\n");
for(i=0;i<20;i++)
 printf(" ");
printf("Enter the values for the input parameters\n\n");
for(i=0;i<10;i++)
 printf(" ");
printf("POLLUTANT:");
scanf("%d",&pol);
for(i=0;i<10;i++)
 printf(" ");
printf("MATERIAL: ");
scanf("%d",&mat);
for(i=0;i<10;i++)
 printf(" ");
printf("PH:");
scanf("%f",&ph);
for(i=0;i<10;i++)
 printf(" ");
```

```
printf("EQUILIBRIUM TIME(hr):");
scanf("%f",&eq_time);
for(i=0;i<10;i++)
 printf(" ");
printf("CONTACT TIME(min):");
scanf("%f",&cont_time);
for(i=0;i<10;i++)
 printf(" ");
printf("ADSORBATE CONCN(mg/ltr):");
scanf("%f",&ad_conc1);                  //adsorbate concn
for(i=0;i<10;i++)
 printf(" ");
printf("ADSORBENT CONCN(g/ltr):");
scanf("%f",&ad_conc2);          //adsorbent concn

min_pol = 1;
max_pol = 8;
min_mat = 1;
max_mat = 25;
min_ph = 0;
max_ph = 14;

switch(pol){
case 1:
    {
     min_eq_time = 1.5;
     min_cont_time = 30;
     min_ad_conc1 =  2;
     min_ad_conc2 = 0.2;
     max_eq_time = 24;
     max_cont_time = 1440;
     max_ad_conc1 = 150;
     max_ad_conc2 = 150;

     break;
    }
case 2:
{
     min_eq_time = 0;
     min_cont_time =30;
     min_ad_conc1 = 50;
     min_ad_conc2 = 0;
     max_eq_time = 1;
     max_cont_time = 80;
     max_ad_conc1 = 500;
     max_ad_conc2 = 20;

     break;
    }
case 3:
{
     min_eq_time =  0.5;
     min_cont_time = 10;
     min_ad_conc1 =  10;
     min_ad_conc2 =  10;
     max_eq_time =  24;
     max_cont_time = 1440;
     max_ad_conc1 = 200;
     max_ad_conc2 = 100;

     break;
    }
case 4:
{
     min_eq_time = 0.5;
     min_cont_time = 30;
     min_ad_conc1 = 20;
     min_ad_conc2 = 0.1;
     max_eq_time =  24;
     max_cont_time = 1440;
     max_ad_conc1 = 70;
     max_ad_conc2 = 10;

     break;
    }
case 5:
```

```
{
     min_eq_time = 0.5;
     min_cont_time = 20;
     min_ad_conc1 =  10;
     min_ad_conc2 =  10;
     max_eq_time =   24;
     max_cont_time =  1440;
     max_ad_conc1 =   100;
     max_ad_conc2 =   100;

     break;
    }
case 6:
{
     min_eq_time = 0.5;
     min_cont_time = 10;
     min_ad_conc1 = 20;
     min_ad_conc2 = 0.1;
     max_eq_time =  24;
     max_cont_time = 1440;
     max_ad_conc1 = 1000;
     max_ad_conc2 = 40;

     break;
    }
case 7:
{
     min_eq_time = 0;
     min_cont_time = 0;
     min_ad_conc1 = 400;
     min_ad_conc2 = 20;
     max_eq_time = 8;
     max_cont_time = 480;
     max_ad_conc1 =  1500;
     max_ad_conc2 =  30;

     break;
    }
default:
{
     min_eq_time = 3;
     min_cont_time =30;
     min_ad_conc1 =  89.193;
     min_ad_conc2 =  3.33;
     max_eq_time =   4;
     max_cont_time = 240;
     max_ad_conc1 =  1000;
     max_ad_conc2 =   50;

     break;
    }
 }

n_pol =  NORMAL(pol,max_pol,min_pol);
n_mat = NORMAL(mat,max_mat,min_mat);
n_ph = NORMAL(ph,max_ph,min_ph);
n_eq_time = NORMAL(eq_time,max_eq_time,min_eq_time);
n_cont_time = NORMAL(cont_time,max_cont_time,min_cont_time);
n_ad_conc1 = NORMAL(ad_conc1,max_ad_conc1,min_ad_conc1);
n_ad_conc2 = NORMAL(ad_conc2,max_ad_conc2,min_ad_conc2);

fprintf(ftest,"%f %f %f %f %f %f %f",n_pol,n_mat,n_ph,n_eq_time,n_cont_time,n_ad_conc1,n_ad_conc2
);


   if((fp1 = fopen("input.dat","r")) == NULL)
   {
    fprintf(stderr, "Error opening file input.dat.\n");
    exit(1);
   }
   if((fp2 = fopen("output.dat","r")) == NULL)
   {
    fprintf(stderr, "Error opening file output.dat.\n");
    exit(1);
   }
   for(flag =0;flag<300;flag++){
```

```
   rewind(fp1);
   for(z=0;z<440;z++)
   {
     fscanf(fp1,"%f %f %f %f %f %f %f", in,in+1,in+2, in+3, in+4, in+5, in+6);

     for(p=0;p<C_MAX;p++)
      tempdiff[p] = 0.0;
     for(p=0;p<C_MAX;p++)
     {
      for(i=0;i<MAX;i++)
      {
    tempdiff[p] += (in[i] - cent[p][i])*(in[i]-cent[p][i]);
      }


     }


      tmp = tempdiff[0];
      min_index =0;
      for(k=0;k<C_MAX-1;k++) {
    if (tmp > tempdiff[k+1])
    {
      tmp =tempdiff[k+1];
      min_index = k+1;
    }

      }

     for(k=0;k<MAX;k++)
     new_cent[min_index].sum[k] += in[k];
     new_cent[min_index].count++;
     error +=tmp;
    //printf("\tsum[0]: %f",new_cent.sum[0]);
   }

   //there should be break here if the error is less than the limiting value
   for(i=0;i<C_MAX;i++)
    for(j=0;j<MAX;j++)
    {
       cent[i][j] = new_cent[i].sum[j]/new_cent[i].count;
      // fprintf(fcent,"cent[%d][%d]: %f\t error=%f perror = %f flag=%d\n",i,j,cent[i][j],error,p
error,flag);
    }
    if(error<50 || fabs(error-perror) < 0.00005) break;
    perror=error;
    error=0.0;
    memset(&new_cent,0,sizeof(ncent)*10);

   }

  //To calculate bandwidth of the receptive field
  for(p=0;p<C_MAX;p++)
  {
    for(m=0;m<C_MAX;m++)
     {
    for(n=0;n<MAX;n++)
    {
       bwtemp[p][m] += NORM(cent[p][n],cent[m][n]);
    }

     }
  }

      for(i=0;i<C_MAX;i++)
    for(j=0;j<C_MAX;j++)
    {
      bw[i][j] = sqrt(bwtemp[i][j]);    // norm of the diff between two centers
      rsum += bw[i][j];             //sum of all the radius
    }
      radius = rsum/100;        //mean radius
      var = radius*radius;
     // printf("%f\t%f\n",radius,var);      //variance = square of radius

    /* to calculate the functions, current outputs and henceforth to
    calculate the weights using the desired outputs and current outputs */
```

```
      for(wctr=0;wctr<1000;wctr++)
      {
       rewind(fp1);
       rewind(fp2);

       for(z=0;z<440;z++)
       {
    fscanf(fp1,"%f %f %f %f %f %f %f", in,in+1,in+2, in+3, in+4, in+5, in+6);
    //printf("the values are %f, %f, %f,%f,%f,%f,%f\n",in[0],in[1],in[2],in[3],in[4],in[5],in[6])
;
    fscanf(fp2,"%f",&dout);
    //printf("the value is %f\n",out);

     //activation functions are calculated
    for(i=0;i<C_MAX;i++)
    {
      Tot_norm[i] = 0.0;
      Tot_temp[i] = 0.0;
      Act_func[i] =0.0;
    }
    for(m=0;m<C_MAX;m++)
    {
     for(n=0;n<MAX;n++)
     {
       Tot_norm[m] += NORM(in[n],cent[m][n]);
     }
     Tot_temp[m] = Tot_norm[m]/(2*var);
     Act_func[m] = exp(-Tot_temp[m]);
    }

    //calculate the current output
     cout = 0.0;
     for(i=0;i<100;i++)
     {
     for(j=0;j<C_MAX;j++)
          cout += Act_func[j]*weight[j];
      //break if dout-cout is less than the certain value
     for(p=0;p<C_MAX;p++)
         weight[p] += ETA*(dout-cout)*Act_func[p];

     if(fabs(dout-cout)<0.001) break;
     cout = 0.0;
     }
      out_diff += fabs(dout-cout);   //sum of diff bet desired and current outputs for all the in
puts
     // fprintf(fout,"\noutdiff: %f  prevdiff: %f",out_diff,prevout_diff);
    }
    //fprintf(fout,"\noutdiff: %f  prevdiff: %f",out_diff,prevout_diff);
    if(out_diff < 0.01 || fabs(out_diff-prevout_diff)<0.000001) break;
     prevout_diff = out_diff;
     out_diff =0.0;
      }
     // printf("wts: %f %f %f %f %f %f %f %f %f  %d %d", weight[0],weight[1],weight[2],weight[3],
weight[4],weight[5],weight[6],weight[7],weight[8],wctr,z);


      /*     //calculate the current output
     cout = 0.0;
     for(j=0;j<C_MAX;j++)
          cout += Act_func[j]*weight[j];

     for(p=0;p<C_MAX;p++)
         weight[p] += ETA*(dout-cout)*Act_func[p];

     // fprintf(fout,"\ncout: %f  dout: %f",cout,dout);
     if(fabs(dout-cout)<0.000001) break;
    }

     // printf("wts: %f %f %f %f %f %f %f %d %d\n", weight[0],weight[1],weight[2],weight[3],weigh
t[4],weight[5],weight[6],wctr,z);

     /* for(i=0;i<C_MAX;i++)
       for(p=0;p<MAX;p++)
      {
       fprintf(fcent,"cent[%d][%d]: %f\n",i,p, cent[i][p]);
      } */
```

```
fprintf(ftest,"%f %f %f %f %f %f %f",n_pol,n_mat,n_ph,n_eq_time,n_cont_time,n_ad_conc1,n_ad_conc2
);


input[0] = n_pol;
input[1] = n_mat;
input[2] = n_ph;
input[3] = n_eq_time;
input[4] = n_cont_time;
input[5] = n_ad_conc1;
input[6] = n_ad_conc2;
   for(i=0;i<C_MAX;i++)
    {
      Tot_norm[i] = 0.0;
      Tot_temp[i] = 0.0;
      Act_func[i] =0.0;
    }
    for(m=0;m<C_MAX;m++)
    {
     for(n=0;n<MAX;n++)
     {
       Tot_norm[m] += NORM(input[n],cent[m][n]);
     }
     Tot_temp[m] = Tot_norm[m]/(2*var);
     Act_func[m] = exp(-Tot_temp[m]);
    }
    //calculate the current output
     cout = 0.0;
     for(j=0;j<C_MAX;j++)
          cout += Act_func[j]*weight[j];
     printf("\ncout: %f", cout);

cont = continue_func();

}
    fclose(ftest);
    fclose(fp1);
    fclose(fp2);

    return 0;
}

int continue_func(void)
{
  int ch;
  printf("\n\nDo you want to continue? (Y)es/(N)o: ");

  fflush(stdin);
  ch=getchar();
  while(ch != 'n' && ch != 'N' && ch != 'y'&& ch != 'Y')
  {
   printf("\n%c is invalid!", ch);
   printf("\n\nPlease enter \'N\' to Quit or \'Y\' to Continue: ");
   fflush(stdin);
   ch = getchar();
  }
  fflush(stdin);
  if(ch == 'n' || ch == 'N')
    return(NO);
  else
    return(YES);
  }
```