```
#include <formatio.h>
#include <utility.h>
#include <analysis.h>
#include <ansi_c.h>
#include <cvirte.h>
#include <userint.h>
#include "TDS-MKC.h"

static int L,L2;  /* lattice dimension */
static int Lattice[500000];  /* Lattice two-dimensional array*/
static int Site[500000];  /*Sites of different desorption energy: "1" - E_des, "2" - E_des2,
"3" - E_des3 */
static long int a, N_init, N_2init, N_3init,  N,  N_1, N_2,  N_3, pt, aa, ff; /* initial and
current number of atoms; number of points */
static long int x, y, NN, a_limit;
static double E_des, E_dif, E_des2, Theta2, E_des3, Theta3; /* desorption and diffusion
energies */
static double E_nn, E_nnn, E_nnnn, scale_factor; /* lateral interaction energy between
nearest-neighbour atoms */
static double Nu; /* pre-exponential factor */
static double beta; /* heating rate */
static double T; /* temperature */
static double T_init, T_fin; /* initial and final temperatures */
static double R_sum, R_sum1, R_sum2, R_sum3, R_sum_des, R_sum_diff, Pk, epsilon,
E_mean, tt, tau, Th, k, Theta0, R_sum_check;
static double Rk[500000], Rk_diff[500000][6], E[500], TDS_T[25000],
TDS_rate[25000][4], E_plot [25000], E_tot[25000], Theta[25000];


double Energy (long int mm);
int number_NN (long int ii);
int number_NNN (long int ii);
int number_NNNN (long int ii);
double Energy_total (int Lat[500000]);
double Energy_hop (int ii, int jj);


int main (int panel, int control, int event,
                  void *callbackData, int eventData1, int eventData2)
{
          long int i,ii,ij,j,i2,j2,i3, j3, b, flag, i_max, i_min, j_max, j_min;
          double rn, r1, r2, E1, E2, P, Gamma_i1, Gamma_i2, R_max, R_min;
          div_t raninit, divres;
```

```
/* Main algorithm */

/* 1. Populating randomly LxL lattice with N_init atoms */

L2=L*L;
for (i=1;i<=L2;i++)      /*initializing the arrays*/
          {
           Lattice[i]=0;
           Site[i]=1;
           Rk[i]=0;
           }
for (a=0;a<500;a++)
          E[a]=0;
for (a=0;a<25000;a++)
          {
           TDS_T[a]=0;
           TDS_rate[a][0]=0;
           TDS_rate[a][1]=0;
           TDS_rate[a][2]=0;
           TDS_rate[a][3]=0;
          }

N_init=(int)(Theta0*L2);  /* initial number of molecules */
N_2init=(int)(Theta2*N_init);  /* fraction of molecules with desorption energy E_des2 */
N_3init=(int)(Theta3*N_init);  /* fraction of molecules with desorption energy E_des3 */
N_2=N_2init; /* counters */
N_3=N_3init;
srand(time(NULL)); /*initializing random number generator*/
a=1;
i=0;
j=0;
while (a<=N_init)  /* populating the lattice */
          {
          i=(int)(((double)rand()/((double)(RAND_MAX)+(double)(1)))*L2)+1;
          if (Lattice[i]==0)
                    {
                    Lattice[i]=1;
                    a++;
                    }
          }

a=1; /*initialization */
while (N_2>0)   /*marking the desorption sites 2*/
          {
```

```
        i=(int)(((double)rand()/((double)(RAND_MAX)+(double)(1)))*L2)+1; /* random
                site i in the region (0..L2] */
        if ((Lattice[i]==1)&&(Site[i]==1))
                {
                b=(int)((double)rand()/((double)(RAND_MAX)+(double)(1))); /* random
                        number [0;1) */
                if (b<0.5)
                        {
                         Site[i]=2; /*site with desorption energy E_des2 */
                         N_2--;
                         }
                }
        }
 while (N_3>0)   /*marking the desorption sites 3*/
        {
        i=(int)(((double)rand()/((double)(RAND_MAX)+(double)(1)))*L2)+1; /* random
                site i in the region (0..L2] */
        if ((Lattice[i]==1)&&(Site[i]==1))
                {
                b=(int)((double)rand()/((double)(RAND_MAX)+(double)(1))); /* random
                        number [0;1) */
                if (b<0.5)
                        {
                         Site[i]=3; /* site with desorption energy E_des3 */
                         N_3--;
                         }
                }
        }
N=0;
N_1=0;
N_2=0;
N_3=0;

for (i=1;i<=L2;i++)  /* counting the number of sites of each configuration */
        {
        if (Lattice[i]==1)
        N++;
        switch (Site[i])
                {
                case 1:
                        N_1++;
                         break;
                case 2:
                         N_2++;
                         break;
                case 3:
```

```
                                N_3++;
                                break;
                        }
                }

k=8.3144;

T=T_init; /* initial temperature value */
tt=0;
pt=0;
R_sum=0;
ff=0;

while ((T<=T_fin)&&(N>0)&&(ff==0)) /* starting temperature ramp */
        {
        R_sum1=0;
        R_sum2=0;
        R_sum3=0;

        /* 2. Rate of desorption for each molecule */

        for (i=1; i<=L2; i++)
                {
                if (Lattice[i]==1) /* site is populated */
                        {
                        Rk[i]=Nu*exp((-Energy[i])/(k*T));
                        switch (Site[i])

                                case 1: /* site 1 */
                                        R_sum1=R_sum1+Rk[i];
                                        break;

                                case 2:  /* site 2 */
                                        R_sum2=R_sum2+Rk[i];
                                        break;

                                case 3:  /* site 3 */
                                        R_sum3=R_sum3+Rk[i];
                                        break;
                        }
                        else /* site is empty */
                                Rk[i]=0;
                } /* for */

Sum1D(Rk, L2+1, &R_sum_des); /* summing all desorption rates*/
```

```
/* 3. Rates of diffusion for each molecule */

for (i=1; i<=L2; i++)
          {
          divres=div(i,L);   /* hopping left */
          if (divres.rem==1)
                    j=i+L-1;
                     else
                              j=i-1;
          if ((Lattice[i]==1)&&(Lattice[j]==0))
                    Rk_diff[i][0]=Nu*exp(-Energy_hop(i,j)/(k*T));
                     else
                              Rk_diff[i][0]=0;
          }


for (i=1; i<=L2; i++)     /* hopping right */
          {
          divres=div(i,L);
          if (divres.rem==0)
                    j=i-L+1;
                    else
                              j=i+1;
          if ((Lattice[i]==1)&&(Lattice[j]==0))
                    Rk_diff[i][1]=Nu*exp(-Energy_hop(i,j)/(k*T));
                    else
                              Rk_diff[i][1]=0;
          }

for (i=1; i<=L2; i++)     /* hopping up */
          {
          if (i<=L)
                    j=L2-L+i;
                    else
                              j=i-L;
          if ((Lattice[i]==1)&&(Lattice[j]==0))
                    Rk_diff[i][2]=Nu*exp(-Energy_hop(i,j)/(k*T));
                    else
                              Rk_diff[i][2]=0;
          }


for (i=1; i<=L2; i++)     /* hopping down */
          {
          if (i>(L2-L))
                    j=i+L-L2;
```

```
                else
                        j=i+L;
        if ((Lattice[i]==1)&&(Lattice[j]==0))
                Rk_diff[i][3]=Nu*exp(-Energy_hop(i,j)/(k*T));
                else
                        Rk_diff[i][3]=0;
        }


for (i=1; i<=L2; i++)           /* hopping up-left */
        {
        divres=div(i,L);
        if (i<=L)
                j=L2-(i-1)*L;
                else
                         if ((divres.rem==1)&&(i!=L2-L+1))
                                j=L2-divres.quot;
                                else
                                        j=i-L-1;
        if ((Lattice[i]==1)&&(Lattice[j]==0))
                Rk_diff[i][4]=Nu*exp(-Energy_hop(i,j)/(k*T));
                else
                        Rk_diff[i][4]=0;
        }


for (i=1; i<=L2; i++)           /* hopping down-right */
        {
        divres=div(i,L);
        if (divres.rem==0)
                j=L+1-divres.quot;
                else
                        if (i>(L2-L))
                                j=L*(L2-i)+1;
                                else
                                        j=i+L+1;
        if ((Lattice[i]==1)&&(Lattice[j]==0))
                Rk_diff[i][5]=Nu*exp(-Energy_hop(i,j)/(k*T));
                else
                        Rk_diff[i][5]=0;
        }


Sum2D(Rk_diff, L2+1, 5, &R_sum_diff); /* summing all diffusion rates*/
R_sum=R_sum_des+R_sum_diff; /* total sum of diffusion and desorption rates */
```

```
/* 4. Process selection (desorption or diffusion) */

raninit=div(pt,100);  /* initializing the random generator after desorbing each 100
          molecules */
if (raninit.rem==0)
          srand(time(NULL));

r1=(double)rand()/((double)(RAND_MAX)+(double)(1)); /* two random
          numbers [0..1) */

r2=(double)rand()/((double)(RAND_MAX)+(double)(1));

flag=0;
ij=0;
Gamma_i1=R_sum*r1;
for (i=1; ((i<=L2)&&(flag==0)); i++)
          for (j=0; j<=5; j++)
                    {
                     Gamma_i2=Gamma_i1-Rk_diff[i][j]; /* subtraction of diffusion
                              rates for each molecule one by one*/
                     if (Gamma_i2>=0)
                              Gamma_i1=Gamma_i2;
                              else                                   /* hopping event */
                                        {
                                        divres=div(i,L);
                                        if (j==0)                 /* hopping left */
                                                  if (divres.rem==1)
                                                            ij=i+L-1;
                                                            else
                                                                      ij=i-1;

                                        if (j==1)                 /*  hopping right */
                                                  if (divres.rem==0)
                                                            ij=i-L+1;
                                                            else
                                                                      ij=i+1;

                                        if (j==2)                 /* hopping up */
                                                  if (i<=L)
                                                            ij=L2-L+i;
                                                            else
                                                                      ij=i-L;

                                        if (j==3)                 /* hopping down */
                                                  if (i>(L2-L))
                                                            ij=i+L-L2;
```

```
        else
            ij=i+L;


if (j==4)           /* hopping up-left */
    if (i<=L)
        ij=L2-(i-1)*L;
        else
            if (divres.rem==1)
                ij=L2-divres.quot;
                else
                    ij=i-L-1;


if (j==5)           /* hopping down-right */
    if (divres.rem==0)
        ij=L+1-divres.quot;
        else
            if (i>(L2-L))
                ij=L*(L2-i)+1;
                else
                    ij=i+L+1;



if (ij!=0)
    {
    Lattice[ij]=1;
    Lattice[i]=0;
    flag=1;
    switch (Site[i])
        {
        case 1:
            N_1--;
            break;
        case 2:
            N_2--;
            break;
        case 3:
            N_3--;
            break;
        }
    switch (Site[ij])
        {
        case 1:
            N_1++;
```

```
                                        break;
                                case 2:
                                        N_2++;
                                        break;
                                case 3:
                                        N_3++;
                                        break;
                                }

                        }
                }

        for (i=1; ((i<=L2)&&(flag==0)); i++)
                {
                Gamma_i2=Gamma_i1-Rk[i]; /* subtraction of desorption
                        rates for each molecule */
                if (Gamma_i2>=0)
                        Gamma_i1=Gamma_i2;
                        else
                                if (Lattice[i]==1)      /* desorption event */
                                        {
                                        Lattice[i]=0;
                                        pt++;
                                        N--;
                                        switch (Site[i])
                                                {
                                                 case 1:
                                                        N_1--;
                                                         break;

                                                case 2:
                                                        N_2--;
                                                        break;

                                                case 3:
                                                        N_3--;
                                                        break;
                                                 }
                        Theta[pt]=(double)(N)/(double)(L2);
                                /* coverage */
                        E_tot[pt]=Energy_total(Lattice)/(double)(N);
                                /* energy per molecule */
                        tau=-log(r2)/R_sum_des;         /* time increment */
                        tt=tt+tau;
                        TDS_T[pt]=T;
                        /* temperature */
```

```
                                        TDS_rate[pt][0]=R_sum_des;
                                        TDS_rate[pt][1]=R_sum1;
                                        TDS_rate[pt][2]=R_sum2;
                                        TDS_rate[pt][3]=R_sum3;
                                        /* desorption rates */
                                        T=T+beta*tau;         /* temperature increment */
                                        flag=1;
                                        }
                                }
                        }
        } /*end while ((T<=T_fin)&&(N>0)) */

}   /* main */


double Energy (long int mm)   /* energy of a particlurar site [mm] */
        {
        double Ei;
        switch (Site[mm])
                {
                case 1:  /* site 1 */
        Ei=E_des+number_NN(mm)*E_nn+number_NNN(mm)*E_nnn+number_NNNN
(mm)*E_nnnn;
                        break;
                case 2:  /* site 2 */
        Ei=E_des2+number_NN(mm)*E_nn+number_NNN(mm)*E_nnn+number_NNN
N(mm)*E_nnnn;
                        break;
                case 3:  /* site 3 */
        Ei=E_des3+number_NN(mm)*E_nn+number_NNN(mm)*E_nnn+number_NNN
N(mm)*E_nnnn;
                        break;
                }
        return(Ei);
                                        }
int number_NN (long int ii)  /* number of occupied nearest neighbours (NN) for a site [ii]
*/

        {
         long int aa,bb,cc,dd,ee,ff,nn;
         div_t divresult;

         nn=0;
         aa=0;  /* left NN */
         bb=0; /* right NN */
         cc=0;  /* upper NN */
```

```
 dd=0;  /* down NN */
 ee=0;  /* upper-left NN */
 ff=0;   /* down-right NN */

divresult=div(ii,L);

if (divresult.rem==1)   /* First column */
            aa=ii+L-1;
            else
                        aa=ii-1;
if (divresult.rem==0)   /* Last column */
            bb=ii-L+1;
            else
                        bb=ii+1;
 if (ii<=L)                  /* First row */
            cc=ii+L2-L;
            else
                         cc=ii-L;
if (ii>(L2-L))              /* Last row */
            dd=ii+L-L2;
            else
                        dd=ii+L;
if (ii<L)
            ee=L2-(ii-1)*L;
            else
                        if ((divresult.rem==1)&&(ii!=L2-L+1))
                                    ee=L2-divresult.quot;
                                    else
                                                if ((ii!=L)&&(ii!=L2-L+1))
                                                            ee=ii-L-1;

if ((divresult.rem==0)&&(ii!=L))
            ff=L+1-divresult.quot;
            else
                        if ((ii>(L2-L+1)))
                                    ff=L*(L2-ii)+1;
                                    else
                                                ff=ii+L+1;


nn=Lattice[aa]+Lattice[bb]+Lattice[cc]+Lattice[dd]+Lattice[ee]+Lattice[ff];

return(nn);
}
```

```
int number_NNN (long int ii)  /* number of occupied next-to-nearest neighbours (NNN)
for a site [ii] */

        {
         long int aa,bb,cc,dd,ee,ff,ix,iy,nnn;
         long int ax,ay,bx,by,cx,cy,dx,dy,ex,ey,fx,fy;
         div_t divresult;
         nnn=0;
         ax=0;
         ay=0;
         bx=0;
         by=0;
         cx=0;
         cy=0;
         dx=0;
         dy=0;
         ex=0;
         ey=0;
         fx=0;
         fy=0;
         aa=0;
         bb=0;
         cc=0;
         dd=0;
         ee=0;
         ff=0;

        divresult=div(ii,L);
        if (divresult.rem==0) /* converting to Cartesian coordinates */
                {
                 ix=L;
                 iy=divresult.quot;
                }
                 else
                        {
                         ix=divresult.rem;
                         iy=divresult.quot+1;
                         }

        if ((ix<=2)&&(iy<(L-1))) /* direction 1 */
                {
                if (L%2==0)
                        ay=iy+(int)((double)(L)/(double)(2))-ix;
                        else
                                ay=iy+(int)((double)(L+1)/(double)(2))-ix;
                if (ay<=L)
                        if ((ix+L)%2==0)
                                ax=L;
                                else
                                        ax=L-1;
                        else
                                {
                                ay=L;
                                ax=2*(L-iy)+1;
                                }
                }

        if ((ix<(L-3))&&(iy==1)&&(ix>2))
                {
                 if ((ix+L)%2==0)
                        ax=L;
                        else
                                ax=L-1;
                                if (ix%2==0)
                                        if (L%2==0)
                                                ay=(int)((double)(L-ix)/(double)(2))+1;
                                                else
                                                        ay=(int)((double)(L-ix-
1)/(double)(2))+1;
                                        else
                                        if (L%2==0)
                                                ay=(int)((double)(L-ix-1)/(double)(2))+1;
                                                else
                                                        ay=(int)((double)(L-
ix)/(double)(2))+1;
                }

        if ((ix>2)&&(iy>1))
                {
                ax=ix-2;
                ay=iy-1;
                }


        if ((iy==1)&&(ix<(L-1))) /* direction 2 */
                if (L%2==0)
                        if ((2*(L-ix)+1)>=(L-1))
                                {
                                 bx=ix+(int)((double)(L)/(double)(2))-1;
                                 by=L-1;
                                 }
```

```
                    else
                            {
                             by=2*(L-ix)+1;
                             bx=L;
                             }
          else
                  if ((2*(L-ix)+1)>=L)
                          {
                           bx=ix+(int)((double)(L+1)/(double)(2))-1;
                           by=L;
                           }
                           else
                                   {
                                    by=2*(L-ix)+1;
                                    bx=L;
                                    }



if ((iy==2)&&(ix<(L-1)))
        if (L%2==0)
                if ((2*(L-ix)+2)>=L)
                        {
                        bx=ix+(int)((double)(L)/(double)(2))-1;
                        by=L;
                        }
                        else
                                {
                                 by=2*(L-ix)+2;
                                 bx=L;
                                }

                else
                        if ((2*(L-ix)+2)>=(L-1))
                                {
                                 bx=ix+(int)((double)(L-1)/(double)(2))-1;
                                 by=L-1;
                                 }
                                 else
                                        {
                                         by=2*(L-ix)+2;
                                         bx=L;
                                        }


if ((iy>2)&&(ix==1)&&(iy<(L-4)))
```

```
            {
             if (L%2==0)
                        if (iy%2==0)
                                   {
                                   by=L;
                                   bx=L-(int)((double)(iy+2)/(double)(2))-2;
                                   }
                                   else
                                              {
                                              by=L-1;
                                              bx=L-(int)((double)(iy+3)/(double)(2))-2;
                                              }
                        else
                                   if (iy%2==0)
                                              {
                                              by=L-1;
                                              bx=L-(int)((double)(iy+2)/(double)(2))-3;
                                              }
                                              else
                                                         {
                                                          by=L;
                                                          bx=L-(int)((double)(iy+3)/(double)(2))-2;
                                                         }
            }


if ((iy>2)&&(ix>1))
            {
            by=iy-2;
            bx=ix-1;
            }


if ((iy==1)&&(ix>2)) /* direction 3 */
            {
             cx=1;
             cy=ix;
            }

if ((ix==L)&&(iy>1)&&(iy<(L-2)))
            {
            cy=L;
            cx=iy;
            }

if ((iy>1)&&(ix<L))
```

```
        {
        cx=ix+1;
        cy=iy-1;
        }

if ((iy==L)&&(ix<(L-1))) /* direction 4 */
        {
        dx=L;
        dy=ix;
        }

if ((ix==1)&&(iy>2)&&(iy<L))
        {
        dy=1;
        dx=iy;
        }

if ((ix>1)&&(iy<L))
        {
        dx=ix-1;
        dy=iy+1;
        }

 if ((ix>=(L-1))&&(iy>2))   /* direction 5 */
        {
        if (L%2==0)
                ey=iy-(int)((double)(L)/(double)(2))+1;
                else
                        ey=iy-(int)((double)(L+1)/(double)(2))+1;

        if (ey>=1)
                if (ix%2==0)
                        ex=2;
                        else
                                ex=1;
                else
                        {
                         ey=1;
                         ex=L-2*ey+2;
                        }
        }

if ((ix<(L-1))&&(iy==L)&&(ix>4))
        {
        if (ix%2!=0)
                {
                ex=1;
                ey=L-(int)((double)(ix-1)/(double)(2));
                }
                else
                        {
                        ex=2;
                        ey=L-(int)((double)(ix-2)/(double)(2));
                        }
        }

if ((iy<L)&&(ix<(L-1)))
        {
        ex=ix+2;
        ey=iy+1;
        }

 if ((iy==L)&&(ix>2)) /* direction 6 */
        if (L%2==0)
                if ((L-2*ix+2)<=2)
                        {
                        fx=ix-(int)((double)(L)/(double)(2))+1;
                        fy=2;
                        }
                        else
                                {
                                 fy=L-2*ix+2;
                                 fx=1;
                                }
                else
                        if ((L-2*ix+2)<=1)
                                {
                                 fx=ix-(int)((double)(L+1)/(double)(2))+1;
                                 fy=1;
                                }
                                 else
                                        {
                                        fy=L-2*ix+2;
                                        fx=1;
                                        }



if ((iy==(L-1))&&(ix>2))
        if (L%2==0)
                if ((L-2*ix+1)<=2)
                        {
```

```
                        fx=ix-(int)((double)(L)/(double)(2))+1;
                        fy=1;
                        }
                        else
                                {
                                 fy=L-2*ix+1;
                                 fx=1;
                                 }

                else
                        if ((L-2*ix+1)<=1)
                                {
                                 fx=ix-(int)((double)(L-1)/(double)(2))+1;
                                 fy=2;
                                 }
                                 else
                                         {
                                         fy=(L-2*ix+1);
                                         fx=1;
                                         }


if ((iy<(L-1))&&(ix==L)&&(iy>4))
        {
        if (iy%2==0)
                {
                fy=2;
                fx=L-(int)((double)(iy-2)/(double)(2));
                }
                else
                        {
                        fy=1;
                        fx=L-(int)((double)(iy-1)/(double)(2));
                        }
        }

if ((ix<L)&&(iy<(L-1)))
        {
        fx=ix+1;
        fy=iy+2;
        }
if (ax*ay>0)
        aa=ax+(ay-1)*L;
if (bx*by>0)
        bb=bx+(by-1)*L;
if (cx*cy>0)
```

```
                cc=cx+(cy-1)*L;
        if (dx*dy>0)
                dd=dx+(dy-1)*L;
        if (ex*ey>0)
                ee=ex+(ey-1)*L;
        if (fx*fy>0)
                ff=fx+(fy-1)*L;

nnn=Lattice[aa]+Lattice[bb]+Lattice[cc]+Lattice[dd]+Lattice[ee]+Lattice[ff];

return(nnn);
}

int number_NNNN (long int ii)  /* number of occupied next-to-next-to-nearest
        neighbours  (NNNN) for a site [ii] */

        {
        long int aa,bb,cc,dd,ee,ff,ix,iy,nnnn;
        long int ax,ay,bx,by,cx,cy,dx,dy,ex,ey,fx,fy;
        div_t divresult;

        nnnn=0;
        divresult=div(ii,L);

        if (divresult.rem==0)
                ix=L;
                else
                        ix=divresult.rem;
        iy=divresult.quot+1;
        ax=0;
        ay=0;
        bx=0;
        by=0;
        cx=0;
        cy=0;
        dx=0;
        dy=0;
        ex=0;
        ey=0;
        fx=0;
        fy=0;
        aa=0;
        bb=0;
        cc=0;
        dd=0;
        ee=0;
```

```
ff=0;

ax=ix+2; /* right */
ay=iy;
if (ax>L)
        ax=ax-L;

bx=ix-2; /* left */
by=iy;
if (bx<1)
        bx=L+bx;

cx=ix; /* down */
cy=iy+2;
if (cy>L)
        cy=cy-L;

dx=ix; /* up */
dy=iy-2;
if (dy<1)
        dy=L+dy;

if ((iy==1)&&(ix>1)&&(ix<(L-3))) /* up-left */
        {
        ey=L-ix;
        ex=L-1;
        }
        else
                if ((iy==2)&&(ix>1)&&(ix<(L-3)))
                        {
                        ey=L-ix+2;
                        ex=L;
                        }
                        else
                                if ((ix==1)&&(iy<(L-3))&&(iy>1))
                                        {
                                         ex=L-iy;
                                         ey=L-1;
                                        }
                                        else
                                                if ((ix==2)&&(iy<(L-3))&&(iy>1))
                                                        {
                                                        ex=L-iy+2;
                                                        ey=L;
                                                        }
```

```
if ((ix>2)&&(iy>2))
        {
        ex=ix-2;
        ey=iy-2;
        }

if ((iy==L)&&(ix>4)) /* down-right */
        {
        fy=L-ix+2;
        fx=2;
        }
        else
                if ((iy==(L-1))&&(ix<L)&&(ix>3))
                        {
                        fy=L-ix;
                        fx=1;
                        }
                        else
                                if ((ix==L)&&(iy>4))
                                        {
                                         fx=L-iy+2;
                                         fy=2;
                                        }
                                        else
                                                if ((ix==(L-1))&&(iy<L)&&(iy>3))
                                                        {
                                                         fx=L-iy;
                                                         fy=1;
                                                        }


if ((ix<(L-1))&&(iy<(L-1)))
        {
        fx=ix+2;
        fy=iy+2;
        }

if (ax*ay>0)
        aa=ax+(ay-1)*L;
if (bx*by>0)
        bb=bx+(by-1)*L;
if (cx*cy>0)
        cc=cx+(cy-1)*L;
if (dx*dy>0)
        dd=dx+(dy-1)*L;
if (ex*ey>0)
```

```
                ee=ex+(ey-1)*L;
        if (fx*fy>0)
                ff=fx+(fy-1)*L;

        nnnn=Lattice[aa]+Lattice[bb]+Lattice[cc]+Lattice[dd]+Lattice[ee]+Lattice[ff];

        return(nnnn);
        }

double Energy_total (int Lat[500000])  /* total energy of the lattice */
        {
         long int aaa;
         double Ee;

        Ee=0.0;
        for (aaa=1; aaa<=L2; aaa++)
                if (Lat[aaa]==1)
                        Ee=Ee+Energy(aaa);

        return(Ee);
        }

double Energy_hop (int ii,int jj)  /* activation barrier of hopping from i to j */
        {
        double Ehop;
        Ehop=E_dif-(Energy(jj)-Energy(ii))/2+pow((Energy(jj)-
Energy(ii)),2)/(16*E_dif);

         return(Ehop);
        }
```

Further contact information:
Uwe Uwe Burghaus, Associate Professor - Surface Chemistry
Department of Chemistry and Molecular Biology
NDSU Dept. 2735, PO Box 6050, Fargo, ND 58108-6050

For DHL, FedEx and UPS deliveries, please use:
Department of Chemistry and Molecular Biology, NDSU Ladd Hall 208
1231 Albrecht Blvd, Fargo, North Dakota 58102, USA

Phone 701-231-9742   FAX 701-231-8831
Email Uwe.Burghaus@ndsu.edu   www.ndsu.edu/chemistry
http://www.uweburghaus.de