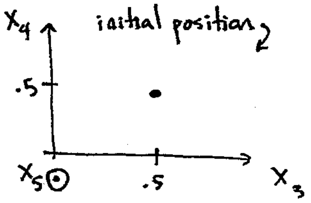
```c
/* ------------------------------------------------------------------------------- */

/* Metropolis algorithm, generic illustration
   [Note: One point particle in 3d cube. You will not get "good
   statistics" with one particle, of course. Illustration only!
   Use as template for Project 5.] */

/* Header files */
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
/* #include "f2c.h" */
/* #include "slatec.h" */

/* Global constants */           # of dimensions; 3 meimenta, 3 position
#define N 6 ←
#define M 1000
#define L 50 ←  reduces statistical dependence

/* Aliases*/
#define RAN (rand()/(float)RAND_MAX)

/* Prototypes */
float p(float *x, int n);   /* probability distribution function */

/* ------------------------------------------------------------------------------- */
int main (void)
{
/*      declarations */
        unsigned int seed;
        const int n=N;
        int i,ii,j,k;
        float x[N],xx[N],r,acpt,eps, ekin,ekin1,ekin2,epot,epot1,epot2;

        FILE *mtout;

        seed=305;
        srand(seed);

    eps=0.2;

/*      Initial X, counters */
        x[0]=0.0; }
        x[1]=0.0; } momentum    initial conditions.
        x[2]=0.0; }
        x[3]=0.5; }
        x[4]=0.5; } space
        x[5]=0.0; }
        i=0;
        j=0;
        acpt=0.0;

/*      Initialize observables, counters */
        ekin1=0.0;
        ekin2=0.0;
        epot1=0.0;
        epot2=0.0;
        k=0;
```

This code is for only one particle

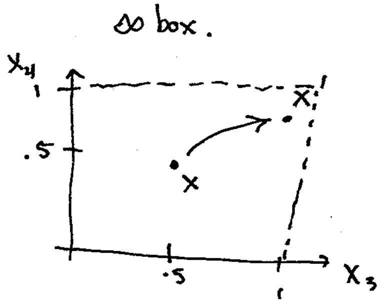
I need to change it for many interacting particles.



```
/*      For the record */
        mtout=fopen("metropolis.out","w");

/*      Initial energies */
        ekin=(powf(x[0],2.0)+powf(x[1],2.0)+powf(x[2],2.))/2.0;
        epot=x[5];
        fprintf(mtout,"%6i%15.7e%15.7e\n",i,ekin,epot);

/*      Limit length of chain to M */

        A:
        if (i>=M) goto C;

/*              Trial X'  */
/*              momenta   */
```

eps=small # (set to .2)

```
                xx[0]=x[0]+(RAN-.5)*eps;
                xx[1]=x[1]+(RAN-.5)*eps;
                xx[2]=x[2]+(RAN-.5)*eps;
/*              positions */
                xx[3]=x[3]+(RAN-.5)*eps;
                xx[4]=x[4]+(RAN-.5)*eps;
                xx[5]=x[5]+(RAN-.5)*eps;

/*              Metropolis update */
                if ( p(xx,n) >= p(x,n) )
/*                      Accept */
                        goto B;
                else {
                        r=RAN;
                        if ( p(xx,n) > r*p(x,n) )
/*                              Accept */
                                goto B;
                        else {
/*                              Reject */
                                j++;
                                goto A;
                        }
                }

                B:

/*              Next element of chain */
                i++;
                for (ii=0; ii<n; ii++) x[ii]=xx[ii];
                acpt=(float)i/(float)(i+j);

/*              Energies */
                ekin=(powf(x[0],2.0)+powf(x[1],2.0)+powf(x[2],2.0))/2.0;
                epot=x[5];

/*              For the record */
                fprintf(mtout,"%6i%15.7e%15.7e\n",i,ekin,epot);

                if (L*(int)(i/L)==i) {  /* mod function */
/*                      accumulate 1st and 2nd moments */
                        k++;
                        ekin1=ekin1+ekin;
```



```
117                         ekin2=ekin2+powf(ekin,2.0);
118                         epot1=epot1+epot;
119                         epot2=epot2+powf(epot,2.0);
120                         printf("%6i%7.4f%4i%14.6e\n",i,acpt,k,(ekin1+epot1)/(float)k);
121                 }
122                 goto A;
123 
124                 C:
125 
126 /*              Final results */
127         ekin1=ekin1/(float)k;
128         ekin2=ekin2/(float)k;
129         ekin2=sqrt(ekin2-powf(ekin1,2.0))/sqrt((float)k);
130         epot1=epot1/(float)k;
131         epot2=epot2/(float)k;
132         epot2=sqrt(epot2-powf(epot1,2.0))/sqrt((float)k);
133                 printf("%6i%7.4f%4i%14.6e%12.4e%14.6e%12.4e\n",i,acpt,k,ekin1,ekin2,epot1,epot2);
134 
135         return 0;
136 
137 }
138 
139 /* ------------------------------------------------------------------------------ */
140 
141 /*      Probability distribution function, def */
142 
143 float p(float *x, int n)
144 {
145                 float s, beta=4.0;
146                 int i;
147 
148 /*              outside box */
149                 s=0.0;
150                 for (i=3; i<5; i++) if (x[i]<0.0 || x[i]>1.0) return s;
151                 if (x[5]<0.0) return s;
152 
153 /*              Boltzmann */
154                 s=exp(-beta*(powf(x[0],2.0)+powf(x[1],2.0)+powf(x[2],2.0))/2.0+x[5]);
155 
156                 return s;
157 }
158 
159 /* ------------------------------------------------------------------------------ */
```

line 68-124 is looking @ his outline

if they are interact
then ∃ an
interacti
potent

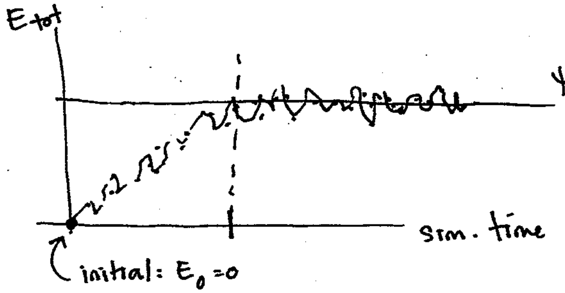
this line reads: $s = e^{-\beta\left[\overbrace{\tfrac{1}{2}(p_x^2+p_y^2+p_z^2)}^{\text{The K.E}} + \overset{\text{the P.E. w/ mg=1}}{z}\right]}$

actually 1/2m w/ m=1.

so this is

$s = e^{-\beta E_{tot}}$, the Boltzmann factor.

make a plot of $E_{tot}$ vs. simulation time



the average energy is const but ∃ fluctuations about that average.

(the kinetic energy is going to be similar in shape, but not @ same time.)

# Markov Chain Process

## Implementation

```
0   Generate initial config $X_i$ , $p(X_i)$ , $i=0$

  1   Generate a (new) trial config $X'$, $p(X')$

    2   if  $p(X') \geq p(X_i)$
            accept, $i=i+1$ , $X_i = X'$
            goto 1
        else
            gen  $r \in [0,1)$ uniform
            if  $p(X')/p(X_i) > r$
                accept, $i=i+1$, $X_i = X'$
                goto 1
            else
                reject
                goto 1
            endif
        endif
```