Physics 40: Laboratory Ten

Thursday, April 30, 2020

Today's Goals: More on analytic solution of diffusion equation;
From random numbers to random walks.

[0] The analytic solution to the diffusion equation is

$$\rho(x,t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} dk \int_{-\infty}^{+\infty} dx' \, \rho(x',0) \; e^{ik(x-x')} \, e^{-Dk^2 t}$$

Here $\rho(x', t = 0)$ is the starting density at $t = 0$. I put the derivation of this formula on the board on Tuesday. It involves some sophisticated mathematics- separation of variables, taking linear combinations of a family of solutions, Fourier integrals and their inverses $\cdots$. Indeed, doing the $\int dk$ leads you to an understanding of 'Green's functions,' a very powerful method to express the general solution of partial differential equations.

If we plug in a 'delta function' initial condition $\rho(x', t = 0) = \delta(x')$ (this just means all the particles are located in a very small region of space at the beginning of the evolution) then

$$\rho(x,t) = \frac{1}{\sqrt{4\pi Dt}} e^{-x^2/4Dt}$$

We are now going to see that your C code gets the same solution. It is really quite amazing that a simple program can duplicate all that high powered mathematics!

[1] The following code computes the analytic solution to the diffusion equation. As usual with codes I supply, you should study it carefully and make sure you understand the logic behind its structure.

```
#include <stdio.h>
#include <math.h>
int main(void)
{
    FILE * fileout;
    int j,Nx;
    double D,rho,dx,x,t,pi=3.1415926;

    fileout=fopen("madeyemoody.txt","w");

    printf("\nEnter dx   ");
    scanf("%lf",&dx);
    printf("\nEnter Nx   ");
    scanf("%d",&Nx);
    printf("\nEnter D,t   ");
    scanf("%lf %lf",&D,&t);

    for (j=-Nx; j<Nx+1; j=j+1)
    {
        x=dx*j;
        rho = exp(-(x*x)/(4.*D*t)) /sqrt(4.*pi*D*t);
        fprintf(fileout,"\n %12.6lf %12.6lf  ",5.+x,rho);
    }

    fclose(fileout);
    printf("\n");
    return 0;
}
```

[HW5-7] Run the code in [1] for $D = 0.02$ and $t = 10$. You can choose what you like for Nx and dx: they just control the resolution and range of x values for which your analytic calculation is done. Plot this analytic solution and check it agrees with your numeric solution from Tuesday.

[HW5-8] Modify your diffusion code so that instead of starting all the 'smoke' in box 500, the smoke instead starts in boxes 400 and 600. Run your new code for $D = 0.02, dt = 0.0001, dx = 0.01, N = 100000$. Make a plot of rho[x] versus x. (Does it matter what values you give to rho[400] and rho[600]?)

[2] The following code executes a random walk. Type it in and compile it. As usual with codes I supply, you should study it carefully and make sure you understand the logic behind its structure.

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

int main()
   {
   int i,N,x;
   unsigned int seed;
   double R;
   printf("\nEnter the number of steps and seed   ");
   printf("\n");
   scanf("%i %u",&N,&seed);
   srand(seed);

   x=0;
   for(i=0;i<N;i++)
   {
      R=(double)rand()/RAND_MAX;
      if (R<0.5)
      x=x+1;
      else
      x=x-1;
   }
   printf("Final location is    ");
   printf("%d",x);
   printf("\n ");

}
```

[HW5-8]
Run your code with $N = 100$ and ten different seeds. What final positions do you get?
Run your code with $N = 10000$ and ten different seeds. What final positions do you get?
Run your code with $N = 1000000$ and ten different seeds. What final positions do you get?
You should find that (as expected!) longer walks (larger N) allow you to get farther from the origin. Does it look to you as if the distance from the origin is growing linearly with N? More slowly? More rapidly? Is there a way you could get more precise information about how the distance depends on N?