Chapter 31-13: Parabolic PDEs Using the Monte-Carlo Method.

It is well known that Monte Carlo methods are preferable for solving large sparse systems, such as those arising from approximations of partial differential equations. Such methods are good for diagonally dominant systems in which the rate of the convergence is high. One of the most important advantages of Monte Carlo methods is that they can be used to evaluate only one component of the solution or some linear form of the solution. This advantage is of great practical interest, since the most important problems in the applied sciences are formulated as problems of evaluating linear or nonlinear forms of the solution. In this case, it is not necessary to perform all computational work which is needed to obtain a complete solution. In addition, even though Monte Carlo methods do not yield more accurate solutions than direct or iterative numerical methods for solving systems of linear algebraic equations, they are more efficient for large n. Also, numerical experiments show that the Monte Carlo method is preferable when one needs to have a coarse estimation of the solution.

An important advance in the generality of Monte Carlo methods came in 2022 with the publishing of the paper *Grid-Free Monte Carlo for PDEs with Spatially Varying Coefficients* by Sawhney, Seyb, Jarosz, and Crane. The paper is available at https://arxiv.org/abs/2201.13240 (https://arxiv.org/abs/2201.13240) and additional accompanying materials are also available.

Defining Parabolic PDE's

•The general form for a second order linear PDE with two independent variables (x, y) and one dependent variable (u) is

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D = 0$$

•Recall the criteria for an equation of this type to be considered parabolic

$$B^2 - 4AC = 0$$

•For example, examine the heat-conduction equation given by

$$\alpha\frac{\partial^2 T}{\partial x^2} = \frac{\partial T}{\partial t}$$

where $A = \alpha, B = 0, C = 0$ and $D = -1$

then

$$B^2 - 4A\,C = 0 - 4(\alpha)(0) = 0$$

thus allowing the classification of the heat equation as parabolic.

With the finite difference implicit method solve the heat problem

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial t} + x - t$$

with initial condition:

$$u(0, x) = \sin(x)$$

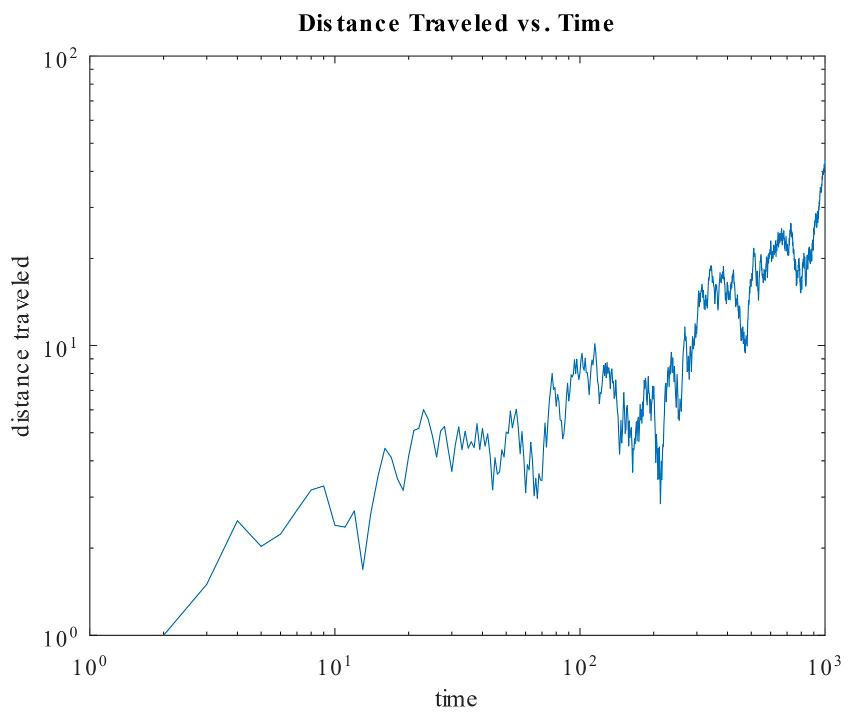
and boundary conditions:

$$u(t, 0) = e^t,\ u(t, 1) = e^t \sin 1$$

The following Octave code is attributed to Jake Blanchard | blanchard@engr.wisc.edu (mailto:blanchard@engr.wisc.edu).

In [ ]:
```
clear all

steplen=1;
startx=0;
starty=0;
nsteps=1000;
angle=2*pi*rand(nsteps,1);
```

```
dx=steplen*cos(angle);
dy=steplen*sin(angle);
x(1)=startx;
y(1)=starty;
for i=2:nsteps
    x(i)=x(i-1)+dx(i-1);
    y(i)=y(i-1)+dy(i-1);
end
plot(x,y,0,0,'ro','LineWidth',2)
xlabel('x')
ylabel('y')
title('Random Walk Path')
distance=sqrt(x.^2+y.^2);
t=1:numel(x);
figure;
loglog(t,distance)
xlabel('time')
ylabel('distance traveled')
title('Distance Traveled vs. Time')
disp('Press any key to continue')

pause

clear all
squaresidelength=2;
area=squaresidelength.^2;
samples=1000
x=squaresidelength*(-0.5+rand(samples,1));
y=squaresidelength*(-0.5+rand(samples,1));
outside=floor(2*sqrt(x.^2+y.^2)/squaresidelength);
circarea=(1-sum(outside)/samples)*area
ins=find((x.^2+y.^2)<=squaresidelength^2/4);
outs=find((x.^2+y.^2)>squaresidelength^2/4);
plot(x(ins),y(ins),'+')
axis equal
hold on
plot(x(outs),y(outs),'r+')
hold off
samples=1000000
x=squaresidelength*(-0.5+rand(samples,1));
y=squaresidelength*(-0.5+rand(samples,1));
outside=floor(2*sqrt(x.^2+y.^2)/squaresidelength);
circarea=(1-sum(outside)/samples)*area
disp('Press any key to continue')
pause
a=2
b=7
randnumbers=a+(b-a)*rand(5,1)
mean(randnumbers)

```



With the Monte Carlo method solve the diffusion equation

$$u_t = \nabla^2 u$$

for $u(x, y, t)$ with the boundary conditions:

$$u(x, 0, t) = u(x, 1, t) = 0$$

$$u(0, y, t) = u(1, y, t) = 0$$

$$u(x, y, 0) = 10$$

The exact solution to the above equations and conditions is:

$$u(x,y,t) \ = \ \frac{160}{\pi^2} \sum_{n,m=1}^{\infty} \frac{e^{-[(2n-1)^2 + (2m-1)^2]\pi^2 t}}{(2m-1)(2n-1)} \sin[(2m-1)\pi x] \sin[(2n-1)\pi y]$$

which is obtained by separation of variables. It is numerically determined that $u(0.6, 0.6, 0.05) \ \simeq \ 5.42$. The following code gives a way to numerically obtain this value in Octave:

In [ ]:
```
numberParticles = 1000 % Number of particles simulated
maxTime = 0.05 % Time at which answer is desired
positionInitial = [0.6 0.6 ] % Particle starting position
valueInitial = 10; % Value of u(x,y,0)
valueBoundary = 0; % Value on unit square edges
valueSum = 0; % Stored values
DeltaSpace = 0.05 % Spatial step
DeltaTime = DeltaSpace**2/4 % Time step
maxNumberTimeSteps = floor(maxTime/DeltaTime) % Maximal number of time steps

for particleCounter = 1:numberParticles % loop for each particle
  position = positionInitial; % particle position
  particleStepCounter = 0; % particle step counter

  particleMoving = true; % particle has not hit boundary
  while (particleMoving)
    % particle update
    if ( rand() < 0.5 ) vec=[0 1]; else vec=[1 0]; endif % step in x or y
    if ( rand() < 0.5 ) vec=-vec; endif % step -1 or +1
    position += DeltaSpace*vec; % update position
    particleStepCounter += 1;
    % Determine if the particle has reached a boundary
    if ( any( ([1 1 -1 -1].*[position position] + [0 0 1 1]) <= 0 ) )
      valueSum += valueBoundary;
      particleMoving = false;
    end
    % Determine if the maximum number of steps have been reached
    if ( particleStepCounter > maxNumberTimeSteps )
      valueSum += valueInitial;
      particleMoving = false;
    end
  endwhile
end
solutionValue = valueSum/numberParticles

```

Supposedly, by either increasing the number of steps, or decreasing the length of the spatial step, the accuracy should converge to the sought value. However, whether it relates to the random number assignment in Octave or has some other cause, the above code does not seem to converge reliably.