Lab 5: Random Processes

Solving Laplaces equation using Random Walks or How to approximate interior points in a grid if we know the boundary

In this lab we want to see how we can use Monte Carlo simulation to approximate the solution to the second order PDE $\Delta u(x,y) = u_{xx} + u_{yy} = 0$ on a given domain such as $[0,1] \times [0,1]$. Of course we need to also specify boundary conditions. Typically, one would use a technique such as finite difference methods to approximate the solution of a PDE. To solve the problem we overlay the domain with a grid. If one uses finite difference techniques, then a difference equation is written at each interior node where the difference equation is obtained by replacing the derivatives in the original PDE with difference quotients. To use Random Walks to approximate the solution we take a different approach. At each interior node where we want an approximate solution we take M random walks (use $M = 100$). Each random walk is stopped when the walk reaches the boundary. We then record the value of the boundary condition at the point where the random walk terminates. We repeat this process M times and then average the values to find the approximation at that node. We then repeat the process at the next node.

1. In this problem we want to write a routine to perform a single random walk in two dimensions. If this was all we were going to use the code for, we might assume that the object always starts at (0,0) and have a tiled square of $(2n+1) \times (2n+1)$ "tiles of length one and then we could move one tile to the north, south, east or west; the random walk would continue until it reaches the boundary, i.e., when x or y is n. Because we will use this routine to approximate the solution to Laplaces equation we will write the code in a slightly different way. As described above, we think of overlaying our domain which we assume is a square region (say $[x_l, x_r] \times [y_b, y_t]$) with a uniform grid with spacing $\Delta x = (x_r - x_l)/(n+1)$ where $n+1$ is the number of cells in each direction. Then we can take a step of length Δx to the north, south, east or west. We also want the capability to start the random walk at each cell point, i.e., $(x_l + i\delta x, y_b + j\delta x)$. Thus we should input the boundaries of our rectangle x_l, x_r, y_b, y_t;and the number of cells in each direction, and the starting point (i,j) whose coordinates are $(x_l + i\delta, y_b + j\delta)$. For this problem you need to output the number of steps, and the path so you can plot it. Write your code and plot the path of a random walk starting from 4 different interior locations in your grid. Use the box $[0,1] \times [0,1]$ with 10 cells in each direction.

2. We now want to use our code from #1 to approximate the surface (the solution to our PDE)

$$-\Delta u = 0 \text{ on } \Omega = (0,1) \times (0,1)$$
$$u = e^y \cos x \text{ on the boundary of } \Omega$$

   With our brownian motion path we need to do little to get an approximation, once our path hits the boundary we calculate the u, and add u to the sum for the point i,j, at the end we divide by the number of steps to get an average for that point:

$$p_{i,j} = \frac{1}{M} \sum u(i',j') \tag{1}$$

   where the i', j' are the last element in our path starting from i,j.

3. Calculate the sum of squared differences at each grid point i,j with the analytical result which is $u(x,y) = e^y \cos x$; change the grid size from using $n = 10$ to $n = 20$ to $n = 50$, does the approximation become more accurate, show in a table