Kernel: SageMath 10.7

2D Random Walks

MATH 242 Modern Computational Math

Today we will explore how frequently two-dimensional simple random walks return to the origin.

First, here is a copy of our 2D random walk function from last time.

In [1]:
```
# function: randomWalk 2D
# input: number of steps
# output: a 2D random walk, specified as a list of positions
def randomWalk2D(numSteps):
    # store the possible directions
    dirs = [[0,1],[0,-1],[1,0],[-1,0]]

    # set up a list to store the locations visited.
    positions = [[0,0]] * (numSteps+1)
    #print(positions)

    # take steps
    for i in range(1, numSteps+1):
        r = randrange(4)
        move = dirs[r]
        positions[i] = [positions[i-1][0] + move[0], positions[i-1][1] + move[1]]

    # return the random walk as a list of locations
    return positions



# testing
rw = randomWalk2D(20)
print(rw)
```

Out[1]:
```
[[0, 0], [1, 0], [0, 0], [1, 0], [0, 0], [1, 0], [2, 0], [1, 0], [0, 0], [-1, 0],
[-1, -1], [-1, -2], [-1, -1], [0, -1], [0, -2], [1, -2], [2, -2], [2, -3], [2, -4],
[2, -3], [3, -3]]
```

As before, there are many different questions we can ask about how often a 2D random walk returns to the origin.

1. What proportion of 2D random walks return to the origin at least once in their first N steps? How does this proportion depend on N? How does this compare with 1D random walks?

In [2]:
```
test = [0,0]
print(test == [0,0])
```

Out[2]:
```
True
```

In [8]:
```
# Generate a 2D random walk and return True if it revisits the origin within
numSteps steps.
# This code stores only the current location, not the list of locations.
def doesItReturn(numSteps):
    rw = randomWalk2D(numSteps)
```

```
    #print(rw)
    return [0,0] in rw[1:]

# testing
doesItReturn(20)
```

Out[8]: True

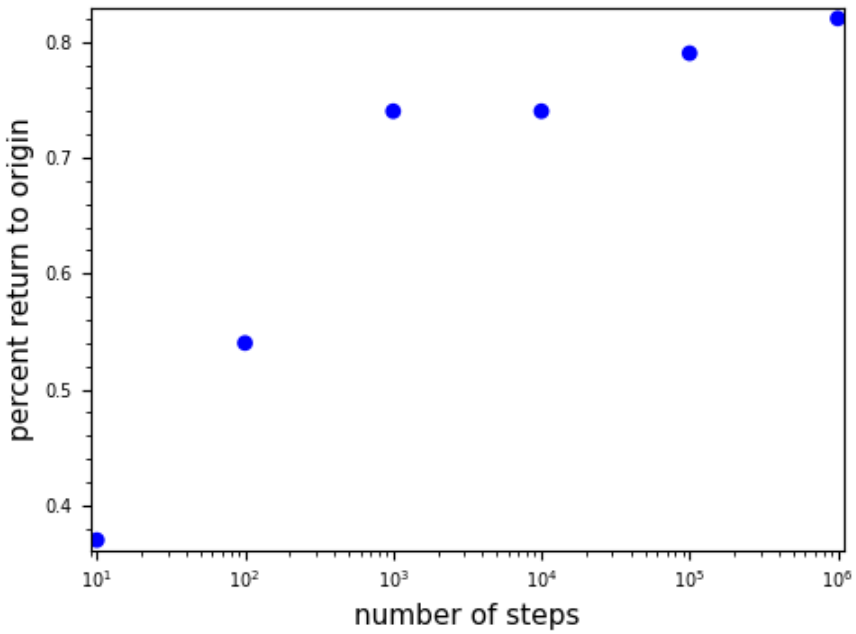
In [9]:
```
# simulate a bunch of walks and count how many return to the origin
def howManyReturn(numWalks, numSteps):
    vals = [doesItReturn(numSteps) for _ in range(numWalks)]
    return sum(vals)  # number of True values
```

In [10]:
```
nvals = [10^k for k in range(1,7)]
print(nvals)
numWalks = 100
pctReturn = [howManyReturn(numWalks, n)/numWalks for n in nvals]
print(pctReturn)
```

Out[10]:
```
[10, 100, 1000, 10000, 100000, 1000000]
[37/100, 27/50, 37/50, 37/50, 79/100, 41/50]
```

In [11]:
```
list_plot( list(zip(nvals, pctReturn)), axes_labels=["number of steps","percent
return to origin"], scale="semilogx", frame=True, size=40, figsize=5, fontsize=7)
```

Out[11]:



2. On average, how many times does a 2D random walk return to the origin in its first N steps? How does this depend on N? How does this compare with 1D random walks?

In [0]:

In [0]:

In [0]:

In [0]:

3. What is the probability that a 2D random walk reaches a distance of M from the origin without returning to the origin? How does this depend on M? How does this compare with 1D random walks?

In [0]:

In [0]:

In [0]:

4. We previously learned that simple symmetric 1D random walks return to the origin with probability 1. What do you think is the probability that a simple symmetric 2D random walk returns to the origin?

In [0]: