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]:
```
# returns a 2D random walk with a specified number of steps
def randomWalk2D(numSteps):
    # define the possible moves at each step of the random walk
    dirs = matrix( [[0,1],[0,-1],[1,0],[-1,0]] )

    # set up a numpy 2-D array to store the locations visited.
    locations = zero_matrix(numSteps, 2)

    # take steps
    for i in range(1, numSteps):
        r = randrange(4)  # random value 0, 1, 2, or 3
        move = dirs[r]    # choose the direction at index r of dirs
        #print(move)
        locations[i] = locations[i-1] + move

    # output the random walk
    return locations
```

In [4]:
```
rw = randomWalk2D(30)
print( rw )
```

Out[4]:
```
[ 0  0]
[ 1  0]
[ 1  1]
[ 2  1]
[ 1  1]
[ 1  0]
[ 1  1]
[ 2  1]
[ 2  2]
```

```
[ 2  1]
[ 3  1]
[ 2  1]
[ 3  1]
[ 3  0]
[ 4  0]
[ 5  0]
[ 4  0]
[ 3  0]
[ 3 -1]
[ 3 -2]
[ 4 -2]
[ 4 -3]
[ 5 -3]
[ 6 -3]
[ 5 -3]
[ 6 -3]
[ 6 -4]
[ 5 -4]
[ 6 -4]
[ 5 -4]
```

In [7]:

```
for row in rw.rows():
    if row[0] == 1:
        print(row)
```

Out[7]:

```
(1, 0)
(1, 1)
(1, 1)
(1, 0)
(1, 1)
```

In [11]:

```
testMatrix = zero_matrix(2,2)
print(testMatrix)
for row in testMatrix.rows():
    print(row)
    print(row == 0)
    print(row == (0,0))
```

Out[11]:

```
[0 0]
[0 0]
(0, 0)
True
False
(0, 0)
True
False
```

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?

The following function generates a 2D simple symmetric random walk of at most `numSteps` steps. However, if the random walk returns to the origin, the function stops and returns `True`. If the random reaches `numSteps` steps without returning to the origin, the function returns `False`.

In [1]:
```
# generate a 2D random walk and return True if it revisits
the origin within numSteps steps
def doesItReturn(numSteps):
    # define the possible moves at each step of the random
walk
    dirs = matrix([[0,1],[0,-1],[1,0],[-1,0]])

    # set up a vector to store the current location
    # note that this does NOT store the list of locations
visited
    loc = vector([0,0])  # [0,0]

    # take steps
    for i in range(1, numSteps):
        r = randrange(4)
        move = dirs[r]  # direction to move
        loc = loc + move  # next location
        #print(loc)
        if loc[0] == 0 and loc[1] == 0:
            break

    return loc[0] == 0 and loc[1] == 0
```

Testing:

In [4]:
```
doesItReturn(30)
```

Out[4]:
```
True
```

The next function simulates a bunch of random walks and returns the number of them that returned to the origin before a specified number of steps.

In [5]:
```
# simulate numWalks random walks, each with at most
numSteps steps, 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
```

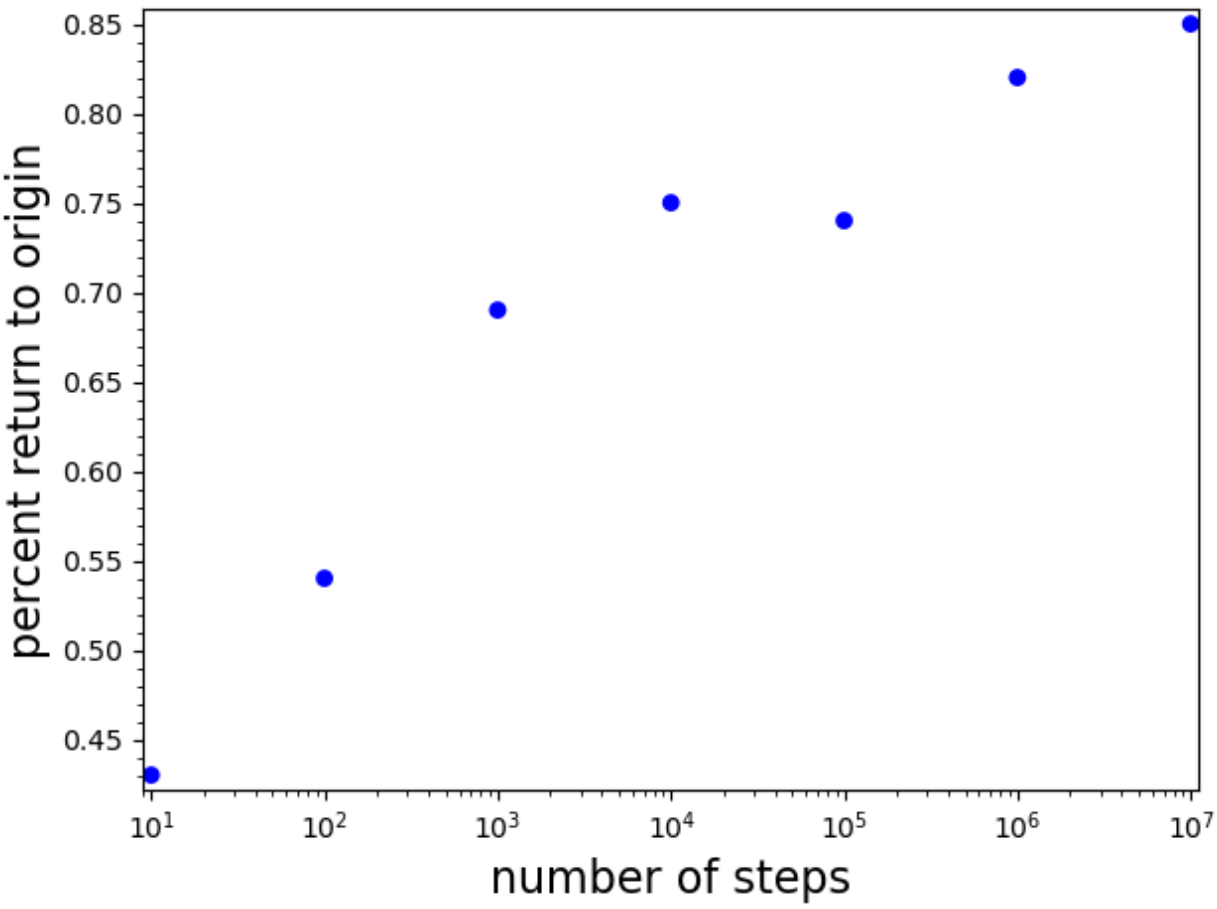
Now compute the percentage of random walks of various lengths that return to the origin and make a plot. *Warning: this takes more than 10 minutes to run!*

In [6]:
```
nvals = [10^k for k in range(1,8)]
print(nvals)
numWalks = 100
pctReturn = [howManyReturn(numWalks, n)/numWalks for n in
nvals]
print(pctReturn)
```

Out[6]:
```
[10, 100, 1000, 10000, 100000, 1000000, 10000000]
[43/100, 27/50, 69/100, 3/4, 37/50, 41/50, 17/20]
```

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

Out[7]:



Interesting. Is the percentage of walks that return to the origin approaching 100% as the number of steps increasing?

In [0]:

In [0]:

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]:

3. What is the probability that a 2D random walk reaches a distance of K from the origin before returning to the origin? How does this depend on K? 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]: