Kernel: SageMath 10.7

Two-Dimensional Random Walks

MATH 242 Modern Computational Math

Today we will start to understand properties of two-dimensional (2D) random walks.

Each position of a 2D random walk is an ordered pair of integers. Thus, a list of positions of a 2D random walk can be stored as a $N \times 2$ matrix, where N is the number of steps in the random walk.

Before we generate 2D random walks, we should practice using matrices in Sage.

In [21]:
```
mat = matrix( [[1,2,3], [4,5,6], [7,8,9]] )
print(mat)
```
Out[21]:
```
[1 2 3]
[4 5 6]
[7 8 9]
```

In [5]:
```
mat[:2]
```
Out[5]:
```
[1 2 3]
[4 5 6]
```

In [22]:
```
mat[:,0]
```
Out[22]:
```
[1]
[4]
[7]
```

In [9]:
```
print( mat.dimensions() )
print( mat.nrows() )
print( mat.ncols() )
```
Out[9]:
```
(3, 3)
3
3
```

In [10]:
```
type( mat )
```

Out[10]:
```
<class
'sage.matrix.matrix_integer_dense.Matrix_integer_dense'>
```

In [11]:
```
zero_matrix(5, 3)
```

Out[11]:
```
[0 0 0]
[0 0 0]
[0 0 0]
[0 0 0]
[0 0 0]
```

Now we can generate a 2D random walk.

In [14]:
```
# define the possible moves at each step of the random walk
dirs = matrix( [[0,1],[0,-1],[1,0],[-1,0]] )

# define the number of steps to take
numSteps = 50

# 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]    # direction to move
    locations[i] = locations[i-1] + move


# output the random walk
print(locations)
```

Out[14]:
```
[ 0  0]
[ 1  0]
[ 1 -1]
[ 1  0]
[ 0  0]
[ 1  0]
[ 1 -1]
[ 2 -1]
[ 2 -2]
[ 2 -3]
[ 2 -4]
[ 3 -4]
[ 3 -5]
[ 2 -5]
[ 2 -6]
[ 1 -6]
[ 1 -7]
```

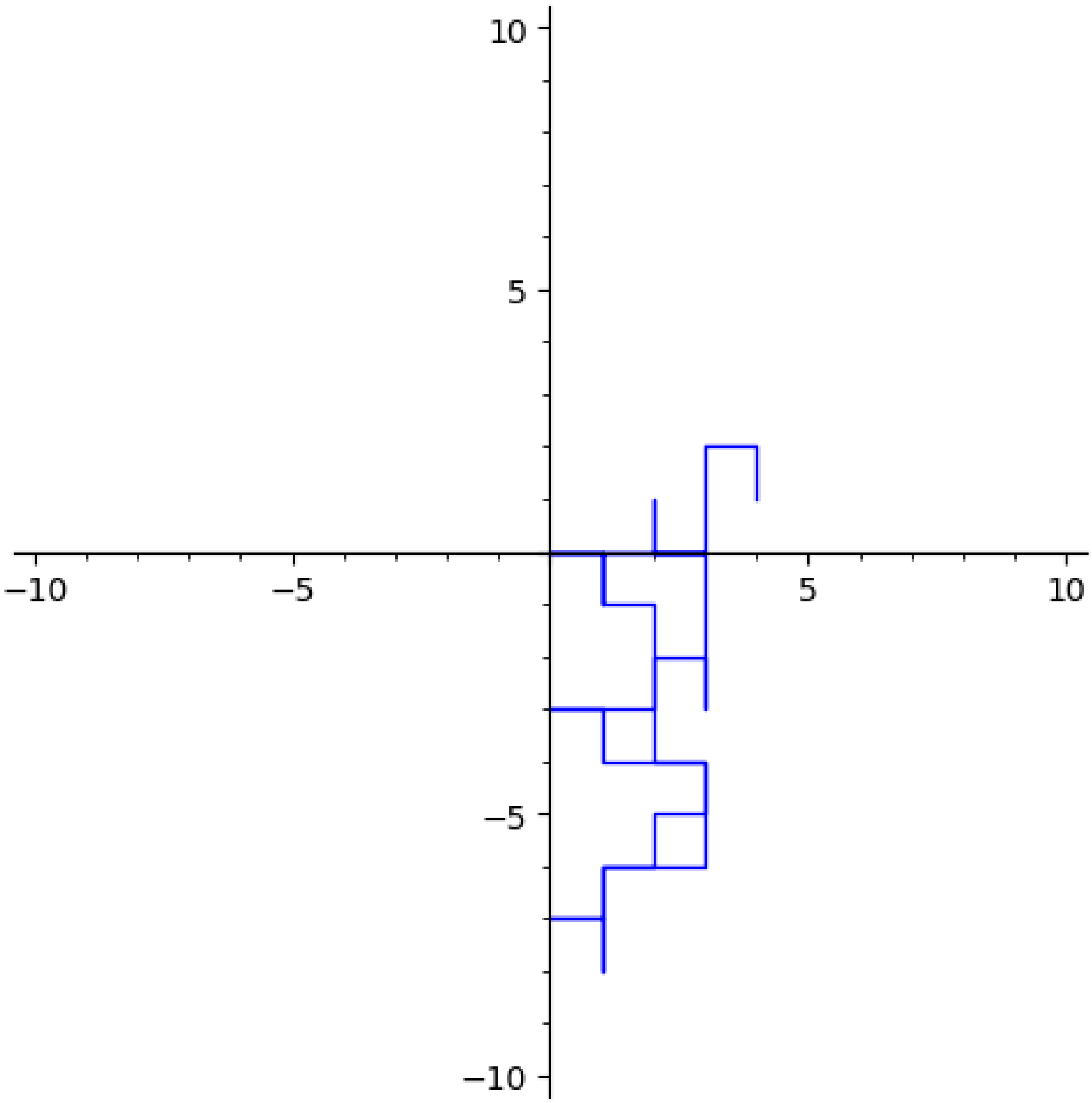
```
[ 1 -8]
[ 1 -7]
[ 0 -7]
[ 1 -7]
[ 1 -6]
[ 2 -6]
[ 3 -6]
[ 3 -5]
[ 3 -4]
[ 2 -4]
[ 1 -4]
[ 1 -3]
[ 0 -3]
[ 1 -3]
[ 2 -3]
[ 2 -2]
[ 3 -2]
[ 3 -3]
[ 3 -2]
[ 3 -1]
[ 3  0]
[ 2  0]
[ 1  0]
[ 2  0]
[ 3  0]
[ 2  0]
[ 2  1]
[ 2  0]
[ 3  0]
[ 3  1]
[ 3  2]
[ 4  2]
[ 4  1]
```

We can also plot the random walk. Here, we will make a 2D plot.

In [15]:

```
list_plot(locations, plotjoined=True, xmin=-10, xmax=10,
ymin=-10, ymax=10, figsize=[5,5])
```

Out[15]:



Now let's put our 2D random walk code into a function, so we can easily generate lots of 2D random walks.

In [1]:

```
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]    # direction to move
        locations[i] = locations[i-1] + move
```

```python
    # output the random walk
    return locations
```

In [20]:
```python
print(randomWalk2D(50))
```

Out[20]:
```
[  0   0]
[  1   0]
[  0   0]
[  1   0]
[  0   0]
[  0  -1]
[ -1  -1]
[ -1   0]
[  0   0]
[  0  -1]
[  0  -2]
[  0  -3]
[  0  -4]
[ -1  -4]
[ -1  -5]
[ -1  -6]
[ -1  -7]
[  0  -7]
[  0  -8]
[  0  -9]
[  0  -8]
[ -1  -8]
[  0  -8]
[  1  -8]
[  1  -9]
[  2  -9]
[  2 -10]
[  1 -10]
[  0 -10]
[  0  -9]
[  1  -9]
[  1  -8]
[  1  -9]
[  0  -9]
[  0 -10]
[  1 -10]
[  1  -9]
[  0  -9]
[  1  -9]
[  0  -9]
[  0  -8]
[  0  -9]
[  1  -9]
[  2  -9]
[  1  -9]
```

```
[  2  -9]
[  2  -8]
[  2  -7]
[  2  -6]
[  2  -5]
```

Diameter of a 2D Random Walk

Define the *width* of a 2D random walk to be the maximum difference between any two x-coordinates attained by the walk. Similarly, define the *height* of the walk to be the maximum difference between any two y-coordinates. Define the *diameter* of the random walk to be the maximum of its width and height.

What is the average diameter of a 2D random walk after N steps? How does this depend on N?

First, a function that takes a 2D random walk (as a list of locations) and returns its diameter.

In [2]:

```
# function that takes a 2D random walk (as a list of
locations) and returns its diameter
def diameter2D(walk):
    xcoords = walk[:,0]
    width = max(xcoords) - min(xcoords)

    ycoords = walk[:,1]
    height = max(ycoords) - min(ycoords)

    return max(width[0], height[0])
```

Testing:

In [3]:

```
aWalk = randomWalk2D(30)
print(aWalk)
diameter2D(aWalk)
```

Out[3]:

```
[ 0  0]
[ 1  0]
[ 0  0]
[-1  0]
[ 0  0]
[ 0 -1]
[ 0  0]
[ 0 -1]
```

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

```
5
```

Function that computes the average diameter of many random walks, all with the same number of steps:

In [4]:
```
# function that computes the diameter of a specified number of random walks, all with the same number of steps
def avgDiameter2D(numSteps, numWalks):
    diams = [diameter2D(randomWalk2D(numSteps)) for _ in range(numWalks)]
    return sum(diams)/numWalks
```

Now compute the average diameter of 2D random walks for a selection of numbers of steps.

In [5]:
```
nvals = range(10,500,10)
avgDiams = [avgDiameter2D(n,100) for n in nvals]
print(avgDiams)
```

Out[5]:
```
[311/100, 481/100, 291/50, 189/25, 413/50, 183/20, 254/25,
1103/100, 58/5, 1257/100, 1231/100, 663/50, 348/25, 357/25,
1509/100, 1533/100, 388/25, 1683/100, 427/25, 1701/100,
917/50, 899/50, 466/25, 989/50, 199/10, 1987/100, 1067/50,
2129/100, 211/10, 546/25, 22, 449/20, 473/20, 463/20,
```

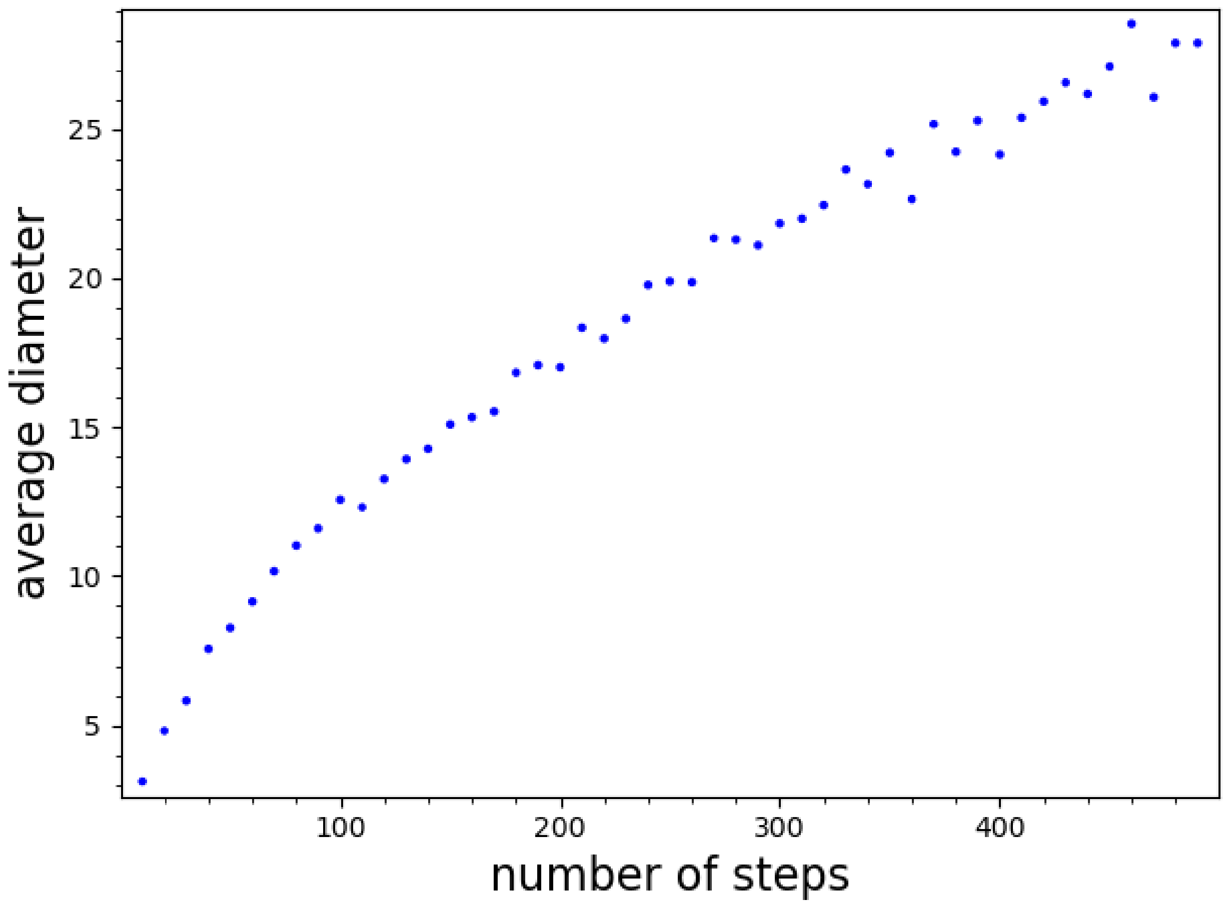
```
2421/100, 453/20, 2517/100, 606/25, 2529/100, 483/20,
2539/100, 2593/100, 664/25, 1309/50, 2711/100, 2853/100,
2607/100, 2789/100, 2789/100]
```

Make a plot of the average diameters we just computed:

In [7]:

```
list_plot( list(zip(nvals, avgDiams)), axes_labels=["number
of steps","average diameter"], frame=True)
```

Out[7]:

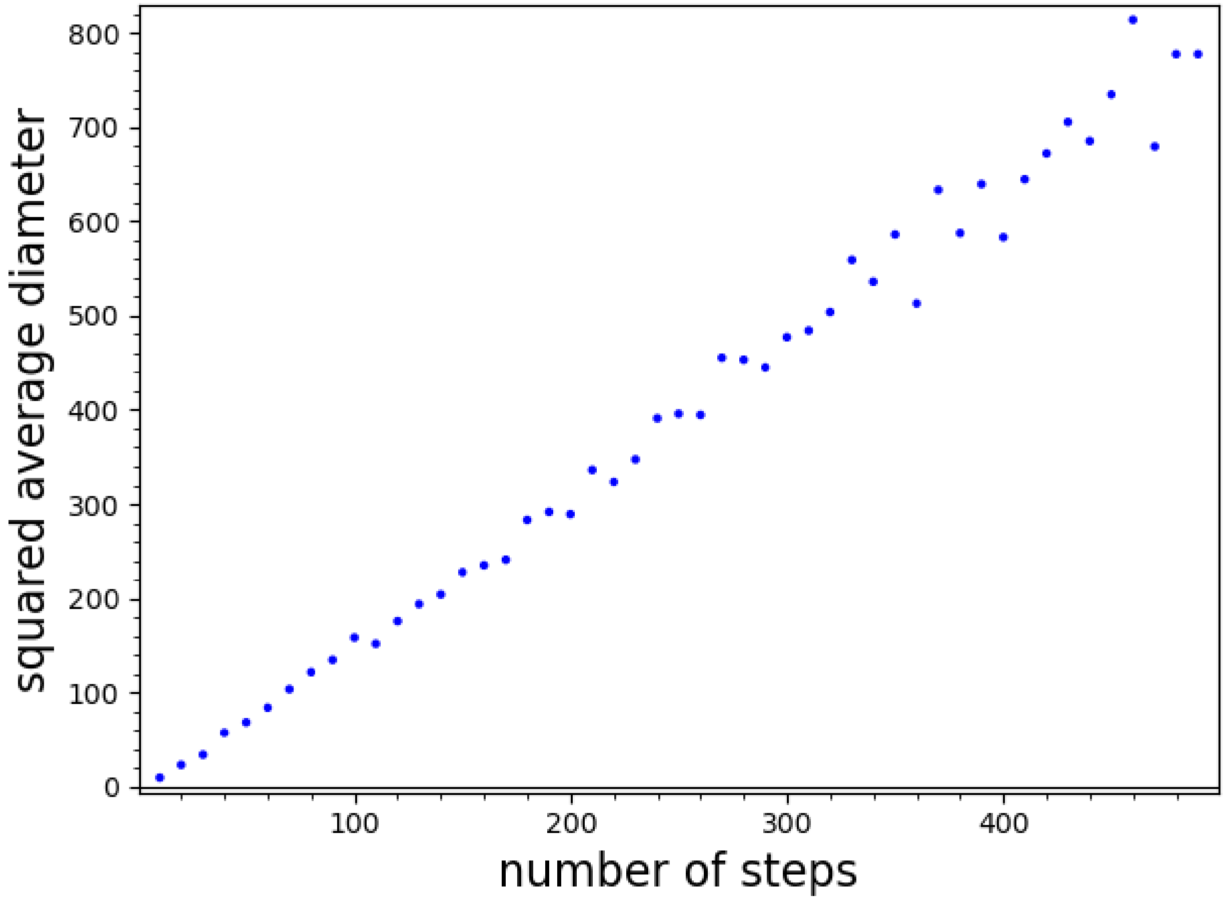


The previous plot looks like it might have a square-root shape, so let's plot the average diameters squared. Since this plot is nearly linear, the previous plot is proportional to a square root function.

In [9]:

```
sqDiams = [d^2 for d in avgDiams]
list_plot( list(zip(nvals, sqDiams)), axes_labels=["number
of steps","squared average diameter"], frame=True)
```

Out[9]:

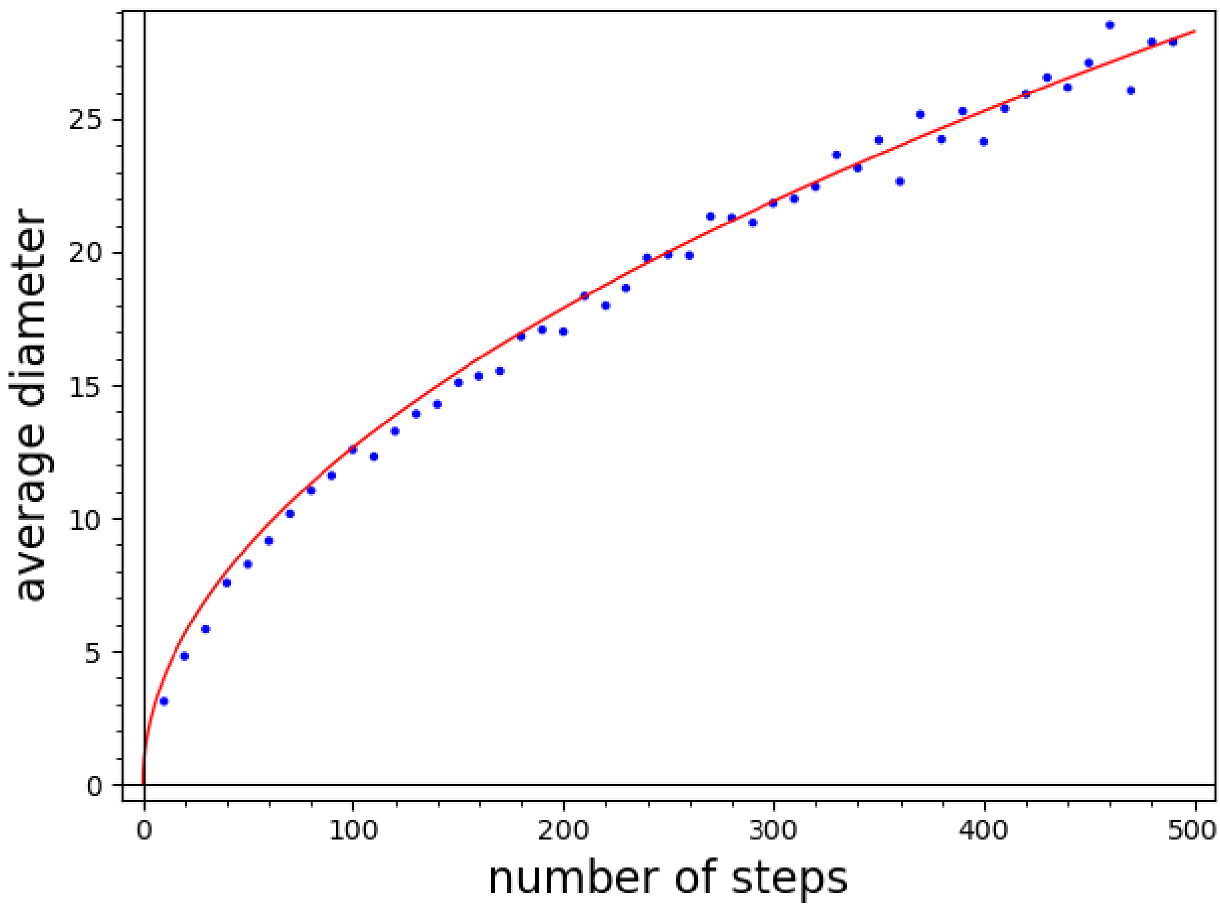


Since the slope of the linear plot is approximately 1.6, we conclude that the average diameter after n steps is approximately $\sqrt{1.6n}$. The following plot shows the avereage diameters along with this square root function.

In [8]:

```
list_plot( list(zip(nvals, avgDiams)), axes_labels=["number of steps","average diameter"], frame=True) + plot(sqrt(1.6*x), (x, 0, 500), color="red")
```

Out[8]:



In [0]:

Four Quadrants

How likely is it that a 2D random walk visits all four quadrants of the plane within N steps? How does this depend on N?

In [0]:

In [0]:

In [0]:

Distinct Points

How many distinct points does a 2D random walk visit, on average, in N steps? How does this depend on N?

```
In [0]:
```

```
In [0]:
```

```
In [0]:
```