random_walk_intro

September 7, 2020

For the last day all that is learned will be put together to complete a small project.

Simulate the random walk of a polymer that is restricted to the 2D plane. The segment length of eachstep in the walk is 1, and the direction for each step is randomly chosen from θ equals 0 to 360^o.

This is an important project. The concepts here are the basis for theories in diffusion, polymers (every protein that exists), Markov Chains, population genetics, etc. This project will focus more on the physical ideas associated with a random walk.

Simulate a polymer with $N = 100$ segments. Start at each simulation at the origin. For each simulation, pick a seed at the beginning, then write each coordinate to a .csv file in tidy format. Plot 5 of these simulation on a graph.

Repeat the simulation and calculate the end-to-end distance, r, for each simulation. Plot the histogram of r^2. Plot the theoretical 2D distribution over your results. Try to carry out enough simulations to get a decent looking histogram, but if the simulations are slow, a rough distribution is fine.

A) Vary the number of steps on the random walk. Using a large number of simulations (say 1000 per set number of steps), find the relationship between number of steps, $N = 100, 200, ...1000$ and the r^2 of end-to-end distance.

B) Now assume that θ is restricted to $\pm\phi$, where ϕ varies from $360, 340, ...$ to $20, 0$ degrees, compute the distribution of r^2 for each of these values. Then find the mean of each r^2 distribution with a different value of ϕ. Plot the relationship of ϕ and r^2.

C) It is a bit harder to visualize, but what if we look into the 3rd dimension? Modify the function to deal with a 3-space coordinate system and calculate the same values of coordinates, x, y, and z displacement, and the net displacement. Plot the same plot from A) with values obtained with this 3D model.

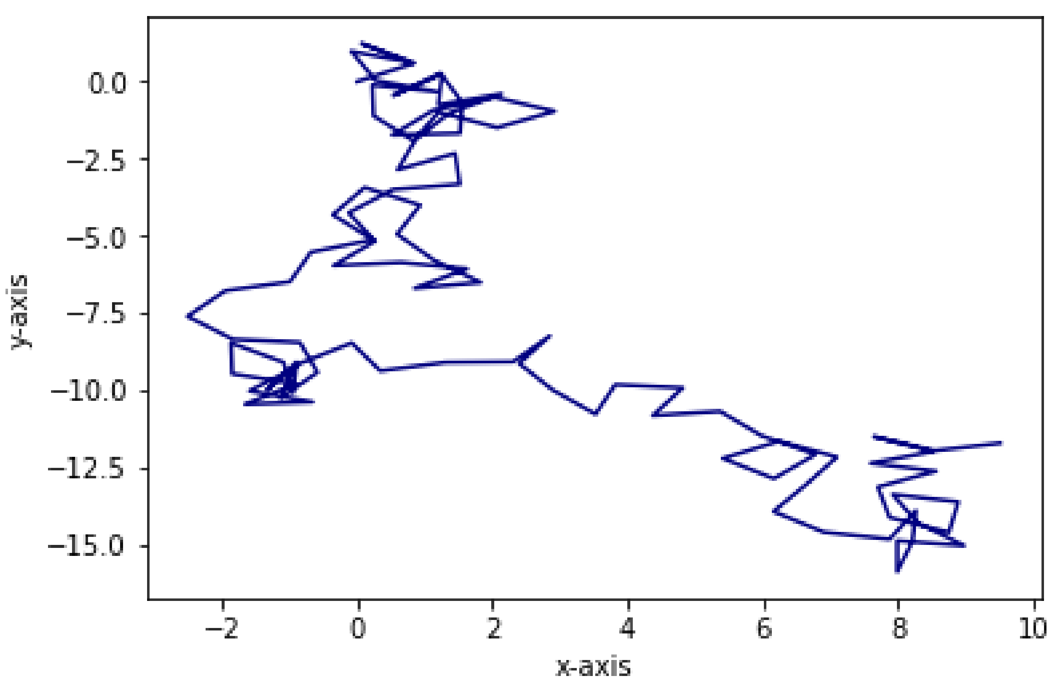
In [59]:
```
# First import packages
import numpy as np
import scipy
import matplotlib.pyplot as plt
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D

import seaborn as sns
import pandas as pd
```

In [60]:
```
# First initiate blank array store data
randwalk_coords = np.zeros((101, 2))

# Iterate over coordinate step (ignore the first step)
for i in range(100):
    # Calculate the new angle
    new_ang = np.random.randint(360)

    # Convert the angle to radians
    new_ang_rad = new_ang / 180 * np.pi

    # Get the new position in Cartesian coordinates
    increment_x = np.cos(new_ang_rad) + randwalk_coords[i, 0]
    increment_y = np.sin(new_ang_rad) + randwalk_coords[i, 1]

    # Assign the new values to the next position
    randwalk_coords[i+1, :] = [increment_x, increment_y]
```

In [13]:
```
randwalk_coords[1]
```

Out[13]:
```
array([0.81915204, 0.57357644])
```

In [15]:
```
# Lets look the new data
_ = plt.plot(randwalk_coords[:, 0], randwalk_coords[:, 1], color='navy')
_ = plt.ylabel('y-axis')
_ = plt.xlabel('x-axis')
```



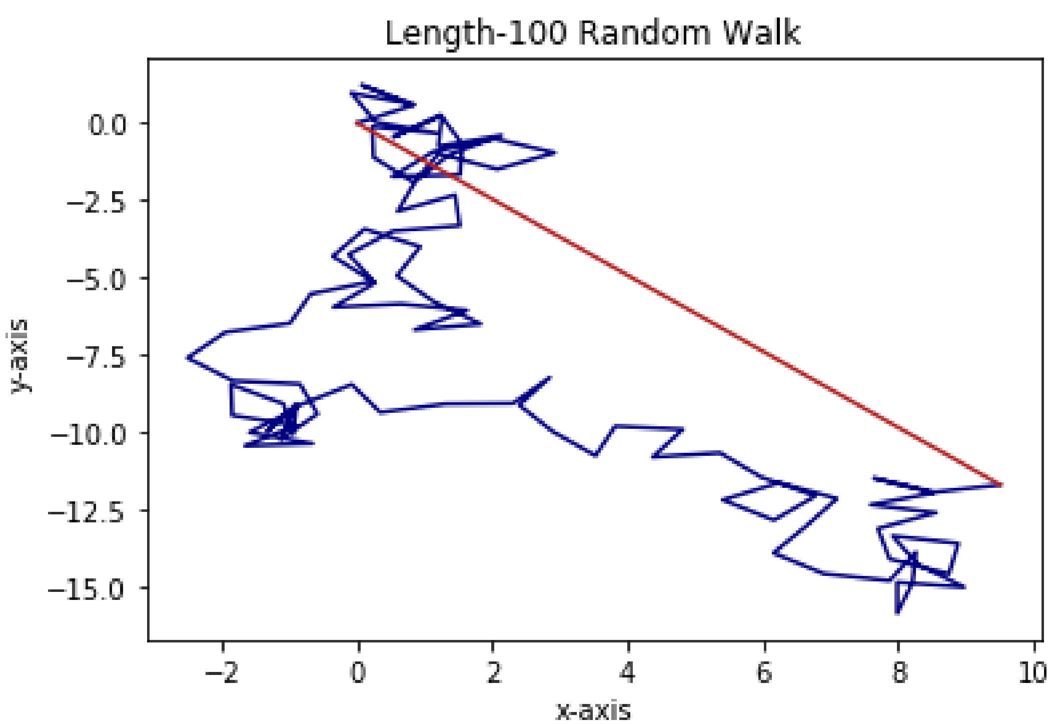
```
In [21]: # Lets find the net displacement

         # Net displacement each x and y coordinate
         first_final_x = [randwalk_coords[0, 0], randwalk_coords[-1, 0]]
         first_final_y = [randwalk_coords[0, 1], randwalk_coords[-1, 1]]

         net_x = first_final_x[1] - first_final_x[0]
         net_y = first_final_y[1] - first_final_y[0]

         print(f'Net displacement in x: {round(net_x, 2)}')
         print(f'Net displacement in y: {round(net_y, 2)}')
```

```
Net displacement in x: 9.5
Net displacement in y: -11.71
```

```
In [22]: # Calculate the total displacement
         net_dist = np.sqrt(net_x ** 2 + net_y ** 2)
         print(f'Net displacement: {round(net_dist, 2)}')
```

```
Net displacement: 15.08
```

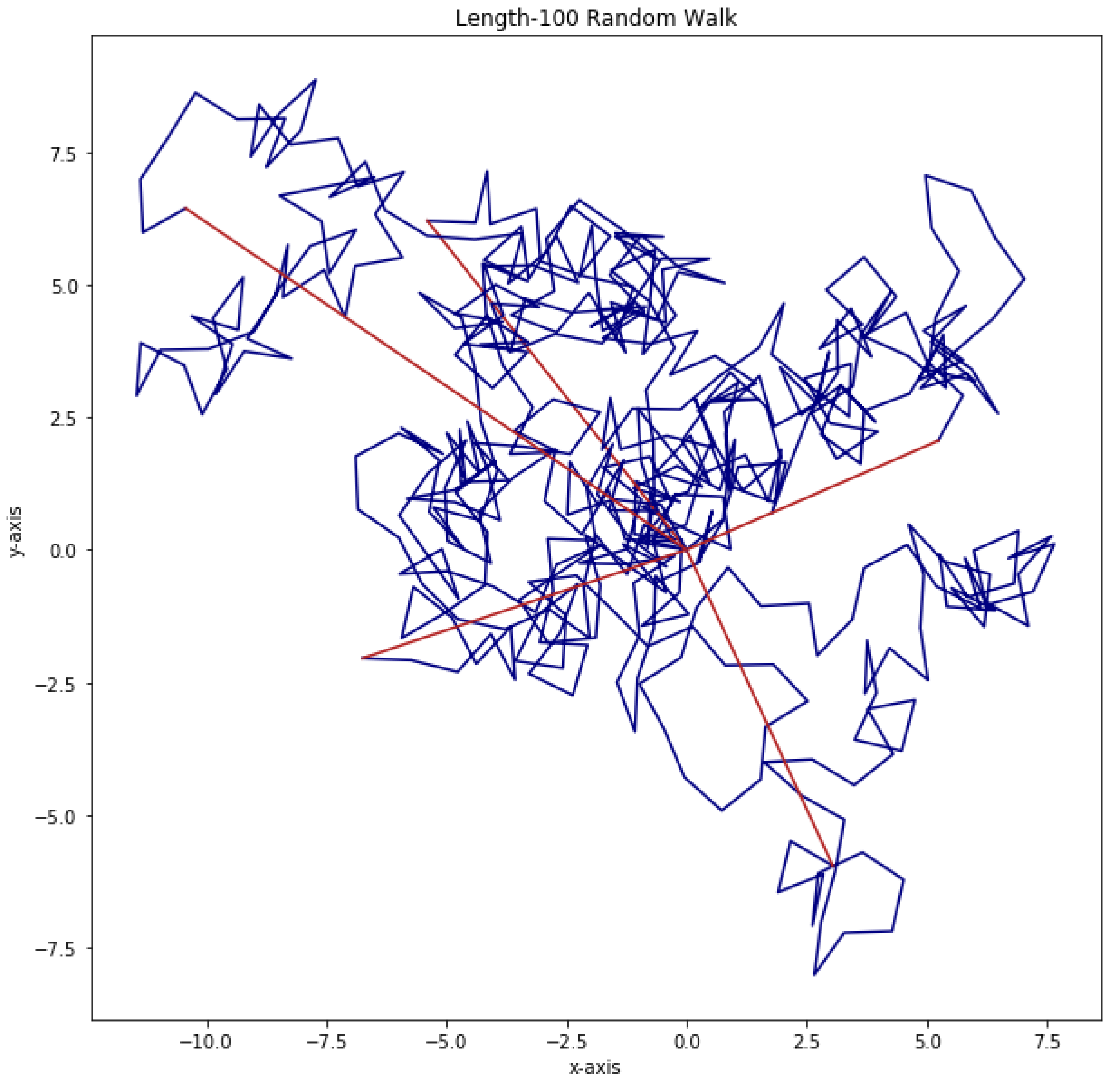
```
In [23]: # Lets look at the total displacement
         _ = plt.plot(randwalk_coords[:, 0], randwalk_coords[:, 1], color='navy')
         _ = plt.plot(first_final_x, first_final_y, color='firebrick')
         _ = plt.ylabel('y-axis')
         _ = plt.xlabel('x-axis')
         _ = plt.title('Length-100 Random Walk')
```



In [24]:
```python
# Define the random walk generator

def randwalk(N=101, rand_seed=42, plot_traj=False):
    """
    For each set of coordinates in a 2xN list of coordinates, compute the next
    set of coordinates using a random number generator.

    param N: number of steps in the random walk
    param rand_seed: seed for the random number generator
    param plot_traj: option of returning a plot of random walk
    """

    # Seed random number generator
    np.random.seed(rand_seed)
    # Initiate the array of coordinates
    randwalk_coords = np.zeros((N, 2))

    # Iterate over coordinate step (ignore the first step)
    for i in range(N-1):
        # Calculate the new angle
        new_ang = np.random.randint(360)

        # Convert the angle to radians
        new_ang_rad = new_ang / 180 * np.pi

        # Get the new position in Cartesian coordinates
        increment_x = np.cos(new_ang_rad) + randwalk_coords[i, 0]
        increment_y = np.sin(new_ang_rad) + randwalk_coords[i, 1]

        # Assign the new values to the next position
        randwalk_coords[i+1, :] = [increment_x, increment_y]

    # Net displacement each x and y coordinate
    first_final_x = [randwalk_coords[0, 0], randwalk_coords[-1, 0]]
    first_final_y = [randwalk_coords[0, 1], randwalk_coords[-1, 1]]

    net_x = first_final_x[1] - first_final_x[0]
    net_y = first_final_y[1] - first_final_y[0]

    # Calculate the total displacement
    net_dist = np.sqrt(net_x ** 2 + net_y ** 2)

    # Plot the random walk
    if plot_traj:
        _ = plt.plot(randwalk_coords[:, 0], randwalk_coords[:, 1], color='navy')
```

```python
            _ = plt.plot(first_final_x, first_final_y, color='firebrick')
            _ = plt.ylabel('y-axis')
            _ = plt.xlabel('x-axis')
            _ = plt.title(f'Length-{N-1} Random Walk')

        return randwalk_coords, net_x, net_y, net_dist
```

In [28]:
```python
# Make the graph big enough to see each trajectory
_ = plt.figure(figsize=(10, 10))

# Define the number of simulations
num_sims = 5
# Store the net displacements
displaces = np.zeros(num_sims)

for j in range(num_sims):
    randwalk_result = randwalk(N=101, rand_seed=j, plot_traj=True)
    displaces[j] = randwalk_result[3]
plt.show()
```




```
In [29]: # Lets look at the net displacement values
         displaces

Out[29]: array([ 8.22878231,  6.70266862,  5.6263533 ,  7.04605217, 12.26058888])

In [30]: # Now lets write out the simulations
         sim_data = randwalk(N=101, rand_seed=0, plot_traj=False)
         temp_df = pd.DataFrame(sim_data[0], columns=['x position', 'y position'])
         temp_df.to_csv('test_random_walk_data.csv', sep=',')

In [31]: # Lets read back the same data
         sim_df = pd.read_csv('test_random_walk_data.csv')
         sim_df.head()
```

Out[31]:
```
   Unnamed: 0  x position  y position
0           0    0.000000    0.000000
1           1   -0.990268    0.139173
2           2   -0.308270    0.870527
3           3   -0.762260    1.761533
4           4   -1.740408    1.553622
```

In [33]:
```
for j in range(num_sims):
    randwalk_result = randwalk(N=101, rand_seed=j, plot_traj=False)
    temp_df = pd.DataFrame(randwalk_result[0], columns=['x position', 'y position'])
    temp_df.to_csv(f'test_random_walk_data_{j}.csv', sep=',')
```

In [34]:
```
# Lets run 100 simulations
num_sims = 100

# Create an array to store the RMSD values
rmsd_values = np.zeros(num_sims)

for j in range(num_sims):
    sim_data = randwalk(N=101, rand_seed=j, plot_traj=False)
    r_value = sim_data[3]
    rmsd_values[j] = r_value
```

In [35]:
```
rmsd_values[:5]
```

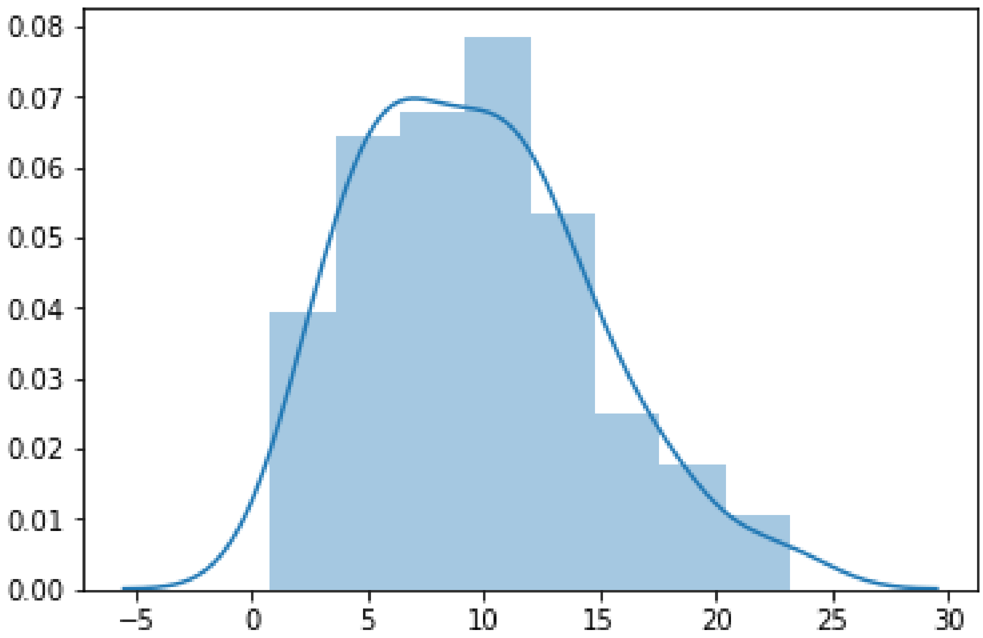
Out[35]:
```
array([ 8.22878231,  6.70266862,  5.6263533 ,  7.04605217, 12.26058888])
```

In [37]:
```
rmsd = np.sqrt(sum([x ** 2 for x in rmsd_values]))
rmsd
```

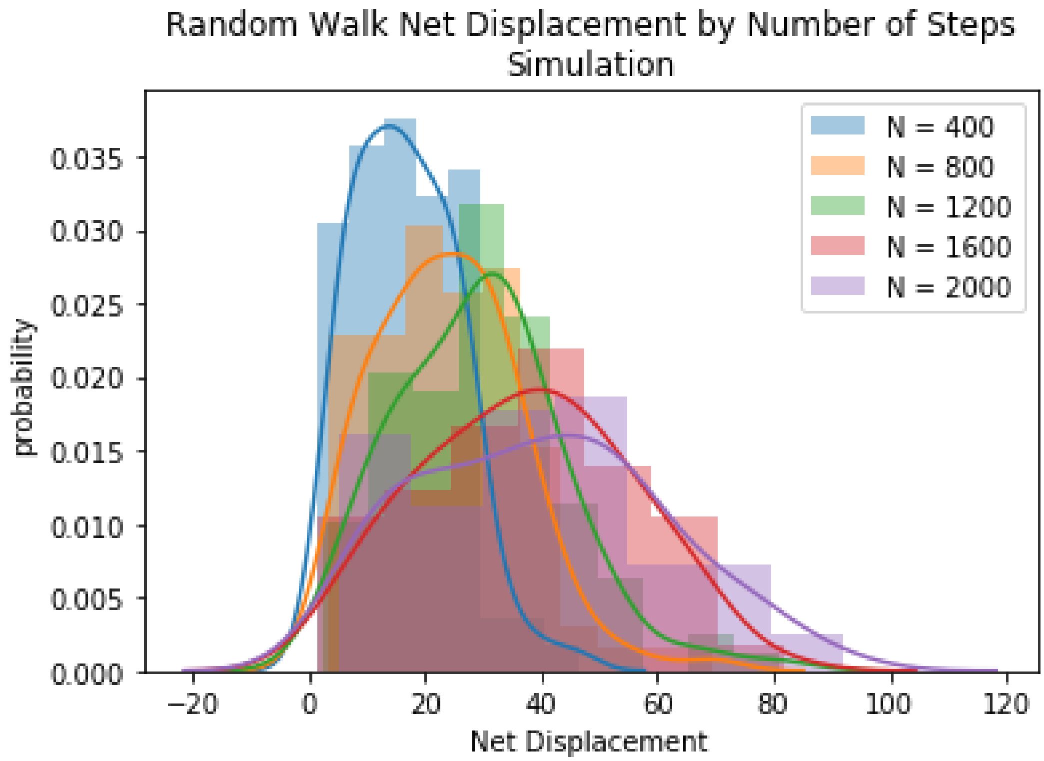
Out[37]:
```
108.16466387755679
```

In [41]:
```
_ = sns.distplot(rmsd_values)
```

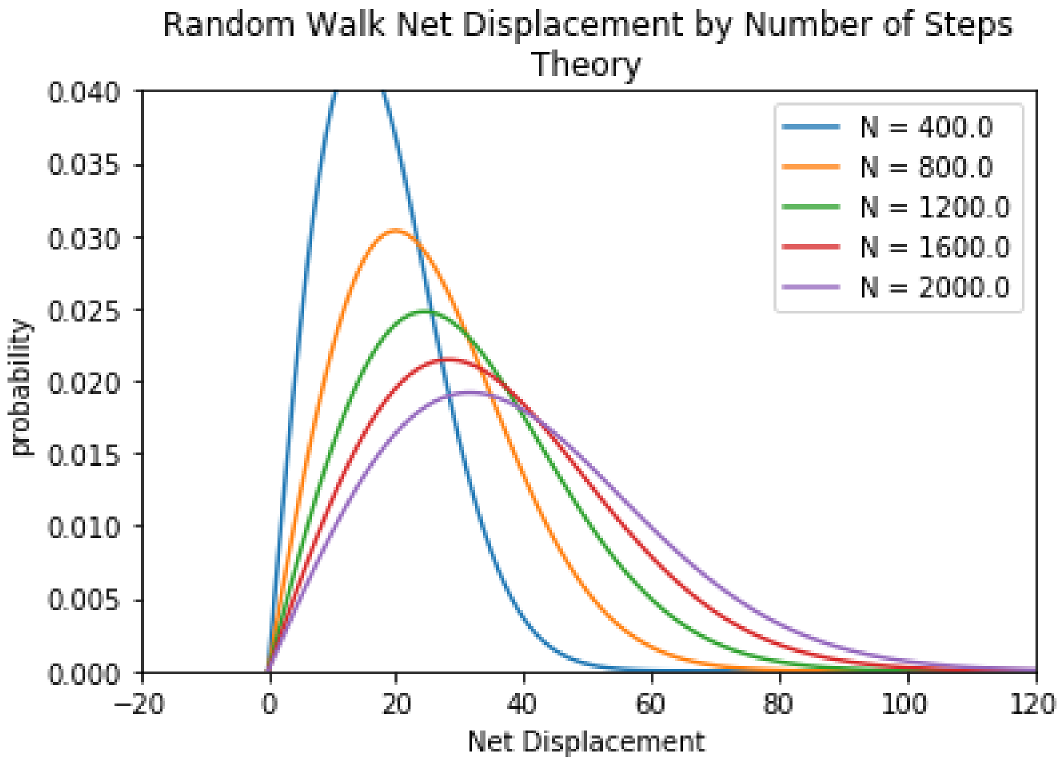
```
/Users/adam/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713: FutureWarning: Us
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```




```
In [42]: # Iterate over different lengths of random walk
         for k in range(5):
             k += 1
             k *= 400

             dist_values = np.zeros(100)
             for j in range(100):
                 sim_data = randwalk(N=k, rand_seed=j, plot_traj=False)
                 dist_values[j] = sim_data[3]
             temp = sns.distplot(dist_values, label=f'N = {k}')

         _ = plt.xlabel('Net Displacement')
         _ = plt.ylabel('probability')
         _ = plt.title('Random Walk Net Displacement by Number of Steps\nSimulation')
         _ = plt.legend()
```



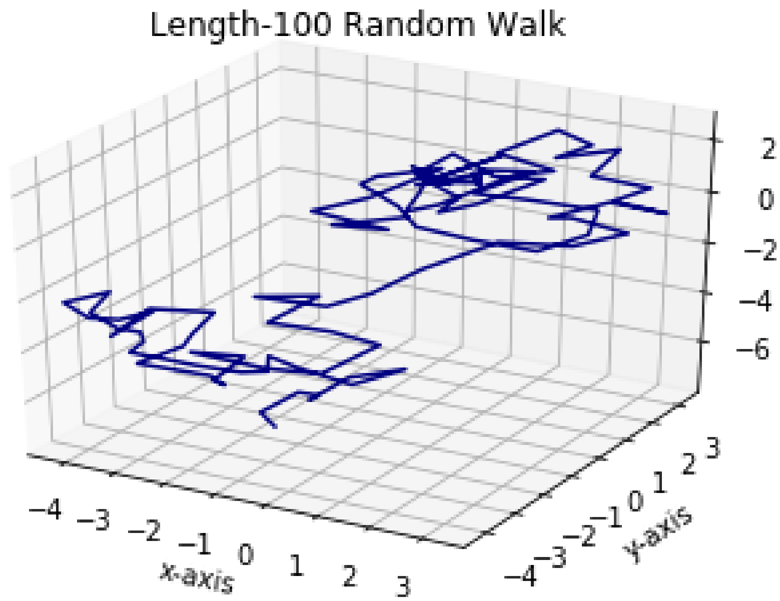
In [45]:
```python
# Theoretical distibution for number of steps
for k in np.linspace(400, 2000, 5):
    x = np.linspace(0, 120, 121)
    _ = plt.plot(x, (x * 2) / (k) * np.exp(-(x ** 2) / (k)), label=f'N = {k}')
_ = plt.xlabel('Net Displacement')
_ = plt.ylabel('probability')
_ = plt.title('Random Walk Net Displacement by Number of Steps\nTheory')
_ = plt.legend()
_ = plt.ylim(0, 0.04)
_ = plt.xlim(-20, 120)
```



In [76]:
```
# Define the random walk generator

def randwalk3d(N=101, rand_seed=42, plot_traj=False):
    """
    For each set of coordinates in a 2xN list of coordinates, compute the next
    set of coordinates using a random number generator.

    param N: number of steps in the random walk
    param rand_seed: seed for the random number generator
    param plot_traj: option of returning a plot of random walk
    """

    # Seed random number generator
    np.random.seed(rand_seed)
    # Initiate the array of coordinates
    randwalk_coords = np.zeros((N, 3))

    # Iterate over coordinate step (ignore the first step)
    for i in range(N-1):
        """# Calculate the new angle
        new_theta = np.random.randint(360)
        new_phi = np.random.randint(180)
```

```
        # Convert the angle to radians
        new_theta_rad = new_ang / 180 * np.pi
        new_phi_rad = new_ang / 180 * np.pi"""

        new_x = 2 * np.random.random() - 1
        new_y = 2 * np.random.random() - 1
        new_z = 2 * np.random.random() - 1

        # Get the new position in Cartesian coordinates
        increment_x = new_y + randwalk_coords[i, 0]
        increment_y = new_y + randwalk_coords[i, 1]
        increment_z = new_z + randwalk_coords[i, 2]

        # Assign the new values to the next position
        randwalk_coords[i+1, :] = [increment_x, increment_y, increment_z]

    # Net displacement each x and y coordinate
    first_final_x = [randwalk_coords[0, 0], randwalk_coords[-1, 0]]
    first_final_y = [randwalk_coords[0, 1], randwalk_coords[-1, 1]]
    first_final_z = [randwalk_coords[0, 2], randwalk_coords[-1, 2]]

    net_x = first_final_x[1] - first_final_x[0]
    net_y = first_final_y[1] - first_final_y[0]
    net_z = first_final_z[1] - first_final_z[0]

    # Calculate the total displacement
    net_dist = np.sqrt(net_x ** 2 + net_y ** 2 + net_z ** 2)

    # Plot the random walk
    if plot_traj:
        fig = plt.figure()
        ax = fig.gca(projection='3d')
        _ = ax.plot(xs=randwalk_coords[:, 0], ys=randwalk_coords[:, 1],
                    zs=randwalk_coords[:, 2], color='navy')
        """_ = ax.plot(xs=first_final_x, ys=first_final_y, zs=first_final_z, color='f
        _ = ax.set_ylabel('y-axis')
        _ = ax.set_xlabel('x-axis')
        _ = ax.set_title(f'Length-{N-1} Random Walk')

    return randwalk_coords, net_x, net_y, net_dist, net_z
```

```
In [77]: temp = randwalk3d(N=101, rand_seed=1137, plot_traj=True)
```




```
In [75]: temp[0][:5]

Out[75]: array([[0.        , 0.        , 0.        ],
                [1.06439418, 1.06439418, 1.94493233],
                [3.7550497 , 3.7550497 , 3.57749351],
                [4.81554195, 4.81554195, 6.5403212 ],
                [7.5225274 , 7.5225274 , 8.87130267]])

In [ ]:
```