TFY4230
Exercise 2

Vegard Stenhjem Hagen

September 14, 2012

Numpy arrays

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

import numpy as np

a100 = np.random.randint(1,7,100)
print 'b) First element of a100: {0}'.format(a100[0])
print 'c) Last element of a100: {}'.format(a100[-1])
print 'd) First three elements of a100: {}'.format(a100[:3])
print 'e) Last three elements of a100: {}'.format(a100[-3:])
print 'f) Third to fifth element of a100: {}'.format(a100[2:5])
print 'g) Every odd element of a100: {}\n'.format(a100[1::2])

a12x12 = 5*np.random.random_sample((12, 12)) + 5
print 'i) Sum of all elements in a12x12: {}'.format(np.sum(a12x12))
print 'j) Sum of all elements in the first column of a12x12: {}'\
  .format(np.sum(a12x12[:,0]))
print 'k) Sum of every third element in the last column of a12x12: {}'\
  .format(np.sum(a12x12[::3,-1]))
```

Random 2D walks

Random 2D walk simulation

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

from __future__ import division
import sys
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

def randomWalk(numberOfSteps):
  #Simulates a random walk of n steps
  steps = np.random.randint(1,7,(numberOfSteps,2)) -7./2
  walk = np.zeros((numberOfSteps,2))
  walk[:,0] = np.cumsum(steps[:,0])
  walk[:,1] = np.cumsum(steps[:,1])
  return walk
```

Plot random walks

```
def plotWalk(numberOfWalks,numberOfSteps=5000):
  #Plot m walks of n steps in the same graph.
  plt.figure(100)
  plt.xlabel(r'$x$-position')
  plt.ylabel(r'$y$-position')
  plt.title(r'Random walk')
  for i in xrange(numberOfWalks):
    walk = randomWalk(numberOfSteps)
    plt.plot(walk[:,0],walk[:,1])
```

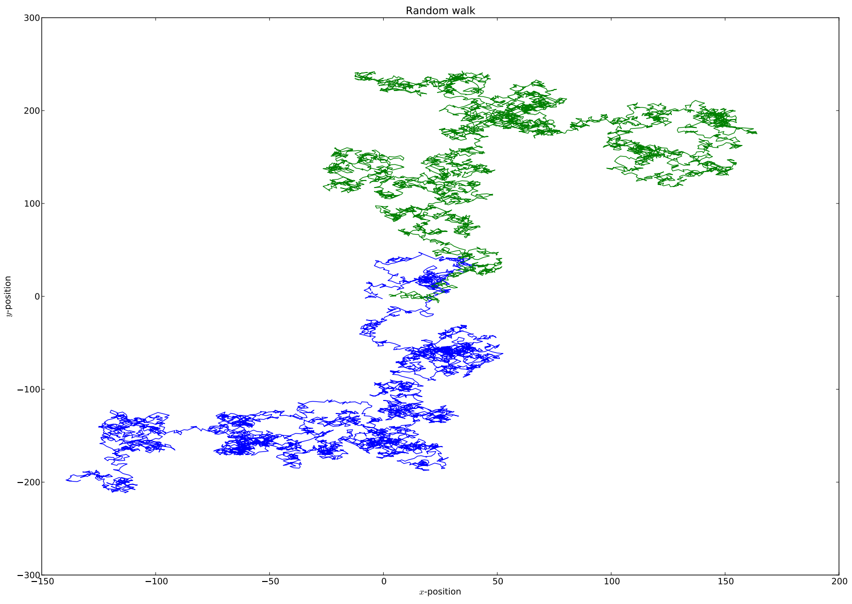

Figure 1: Graph of two random walks of 5000 steps each.

Figure 1 shows the graph of two random walks of 5000 steps each. A number of simulations were run before this pair of random walks were chosen. The reason for selecting this particular pair is purely aesthetic as there was little overlap in the walk-paths.

Gather statistics of random walks and plot it

```
def statWalk(numberOfWalks,numberOfSteps=100):
  #Simulate m walks of n steps and plot the maximum
  #absolute x-value, and the endpoint x-value
  absXmax = np.zeros(numberOfWalks)
  endpointX = np.zeros(numberOfWalks)
  walk = np.zeros((numberOfSteps,2))
  for i in xrange(numberOfWalks):
    #Simulate the walks
    walk = randomWalk(numberOfSteps)
    #Extract the maximum absolute x-value
    absXmax[i] = np.max(np.abs(walk[:,0]))
    #Extract the endpoint x-value
    endpointX[i] = walk[-1,0]

  nbins_max = int(np.max(absXmax)) + 1
  histogram_max = np.zeros(nbins_max)
```

```
for n in absXmax:
  #Fill the maximum absolute x-value bins
  bin = int(n)
  histogram_max[bin] += 1

plt.figure(1)
plt.subplot(221)
plt.title(r'Absolute maximum x-value of random walks')
plt.xlabel(r'$x$-position')
plt.plot(histogram_max/numberOfWalks)

plt.subplot(223)
plt.title(r'Log absolute maximum x-value of random walks')
plt.xlabel(r'$x$-position')
plt.semilogy(histogram_max)

nbins_end = int(2*np.max(abs(endpointX)))
origin = (nbins_end-1)/2
histogram_end = np.zeros(nbins_end)

for n in endpointX:
#Fill the endpoint x-value bins
  bin = origin + int(n)
  histogram_end[bin] += 1

plt.subplot(222)
plt.title(r'Endpoint x-value of random walks')
plt.xlabel(r'$x$-position')
x = np.linspace(np.min(endpointX),np.max(endpointX),nbins_end)
plt.plot(x,histogram_end/numberOfWalks)

plt.subplot(224)
plt.title(r'Log endpoint x-value of random walks')
plt.xlabel(r'$x$-position')
plt.semilogy(x,histogram_end)
```

The left side of figure 2 shows a histogram of the maximum absolute x-value of 100000 random walks of 100 steps, while the right side shows a histogram of the endpoint x-value of the same walks. The top graphs are plotted with normal axes, while the bottom graphs are plottet with a logarithmic second axis. The histogram of the maximum x-values resembles a poisson distribution. The histogram of the endpoint x-values highly resembles a gaussian distribution with expactation value 0, which is to be expected by the law of large numbers.

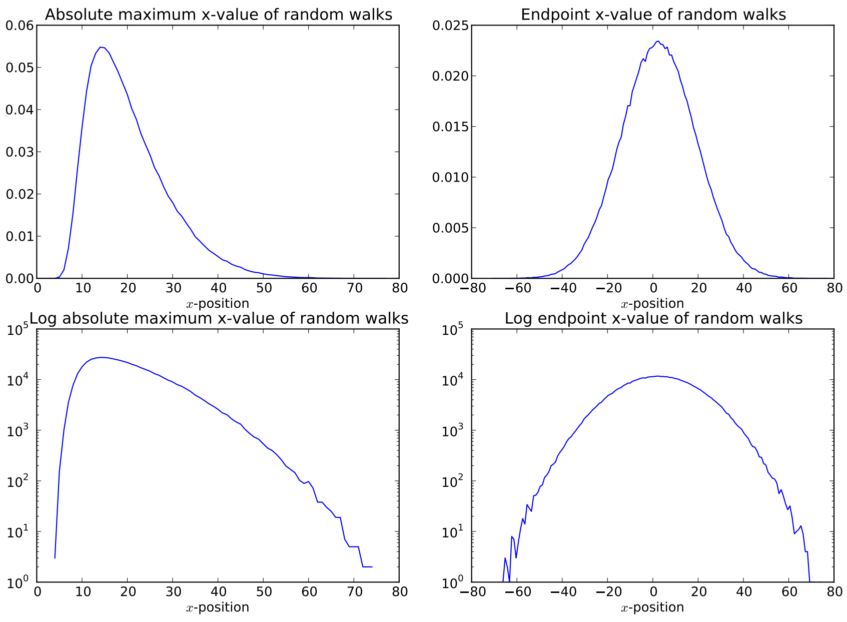

Figure 2: Graphs showing statistical data of 100000 random walks of 100 steps.