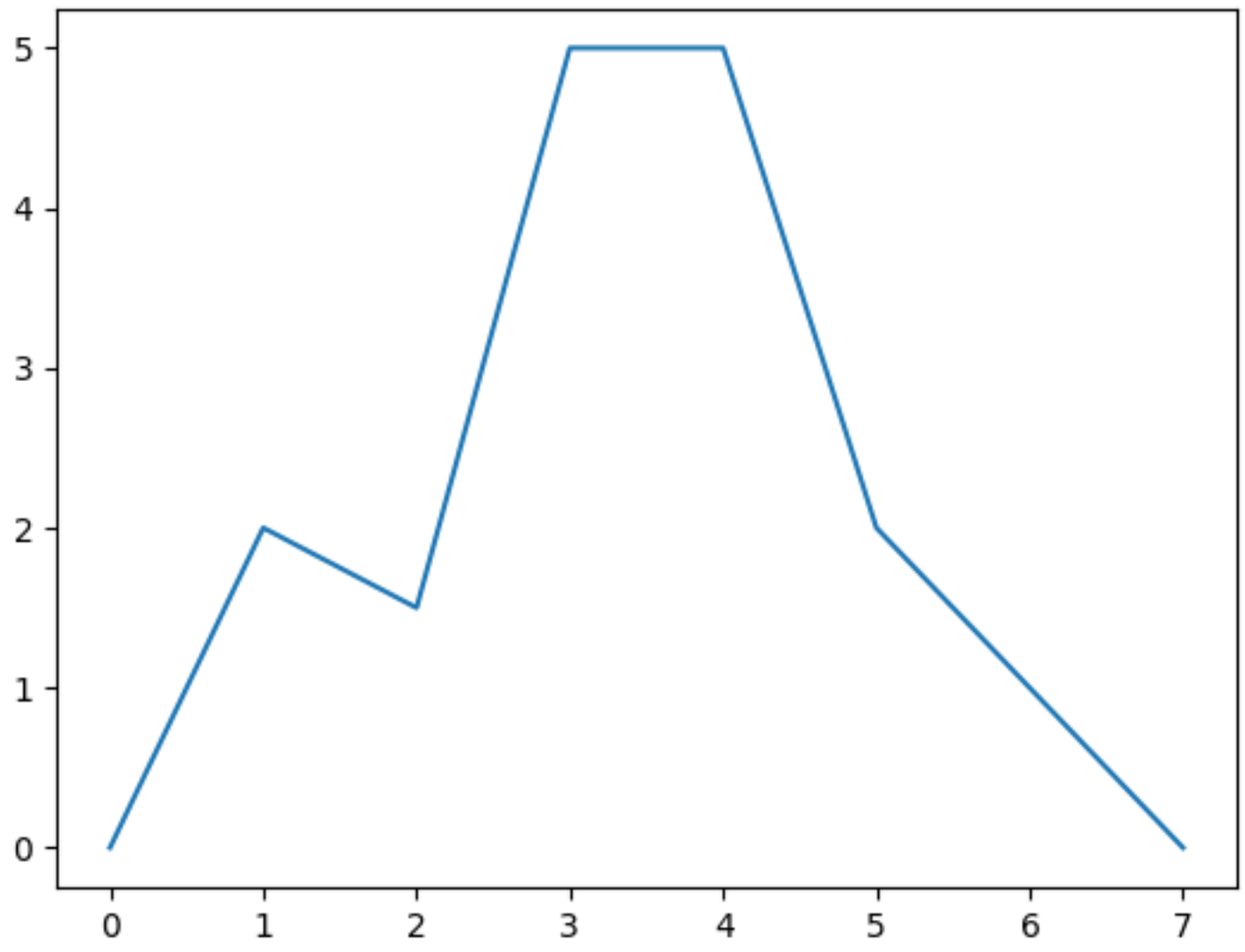
Introduction to Python

Lecture 4

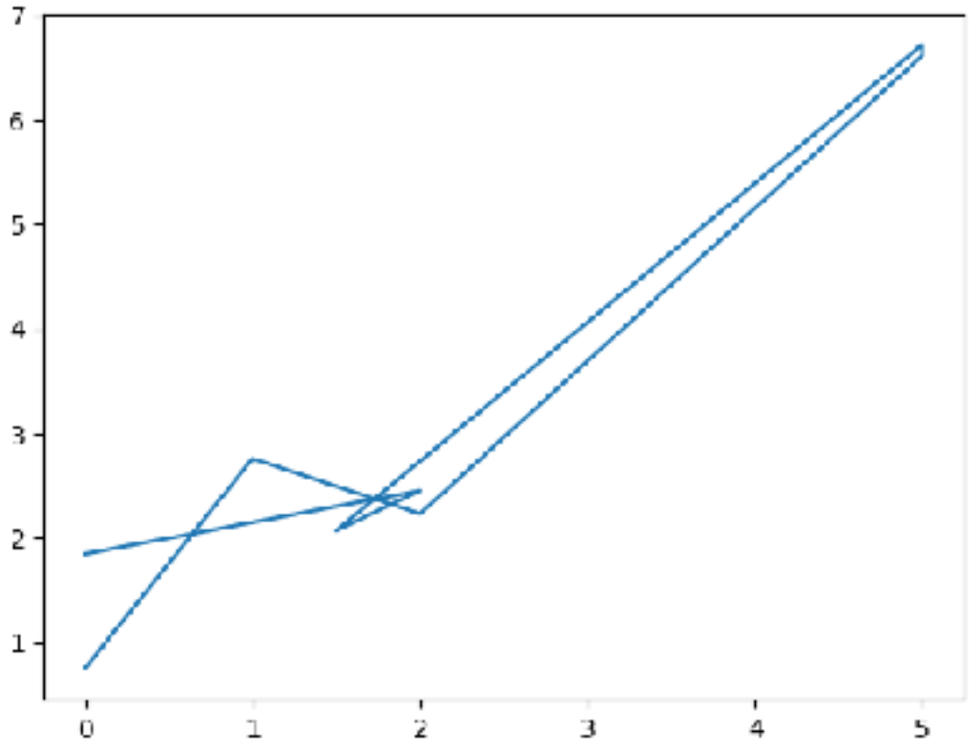
Arul Lakshminarayan, 13/10/17

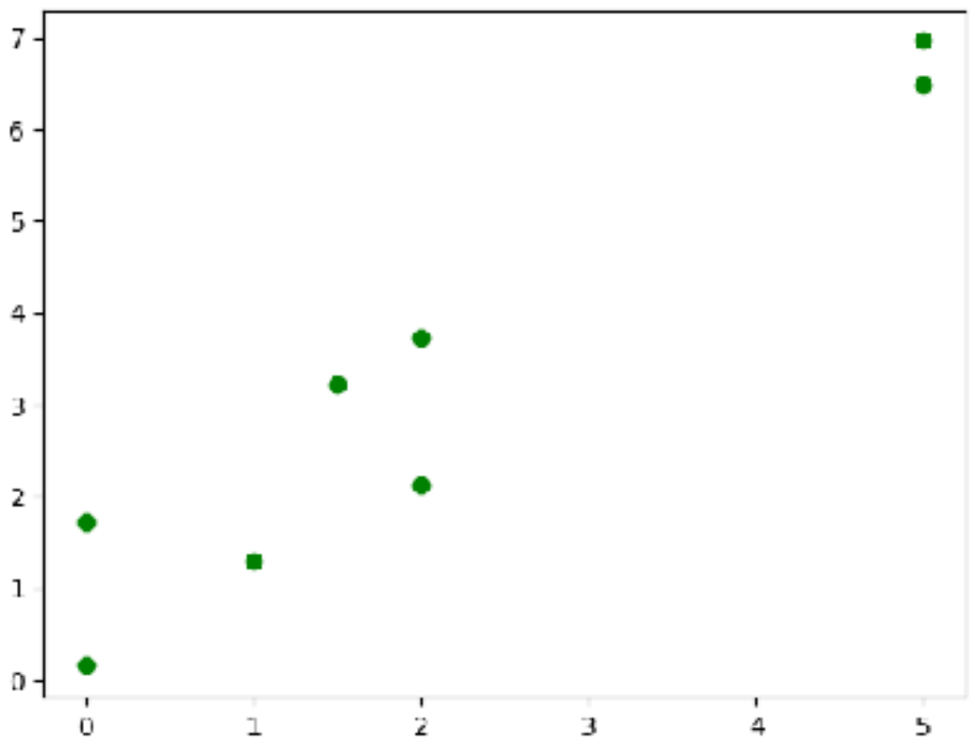
Basic plotting

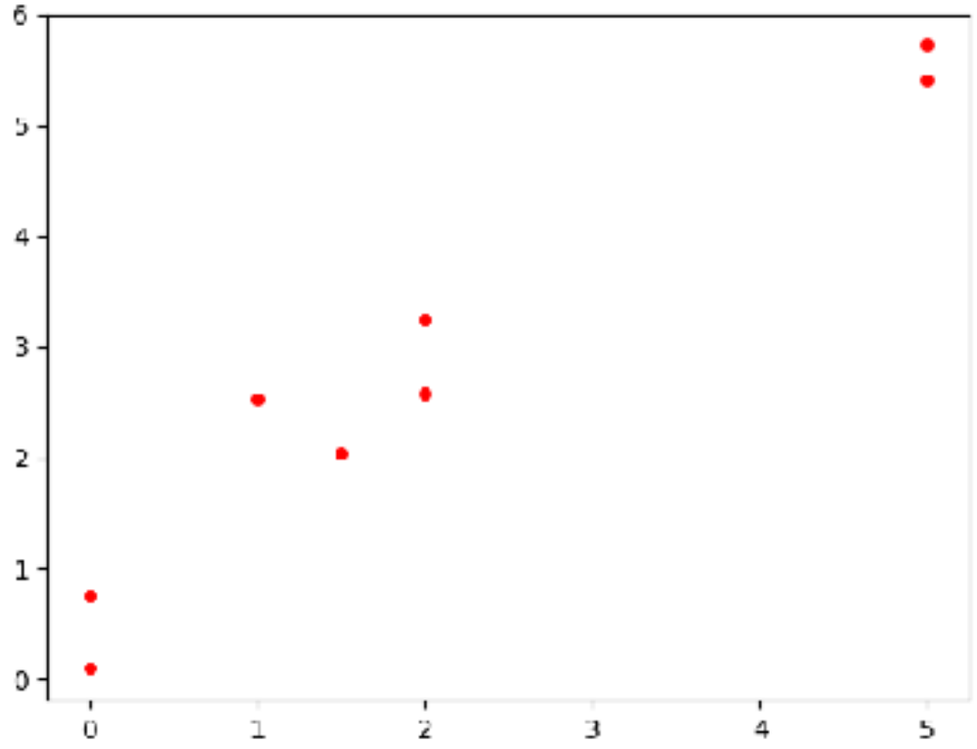
```
import pylab as plt
import numpy as np

a=[.0,2.0,1.5,5,5,2,1,0]
plt.plot(a)
plt.show()
```



```
import pylab as plt
import numpy as np

x=[.0,2.0,1.5,5,5,2,1,0]
y=x+2*np.random.random(len(x))
plt.plot(x,y)
#plt.plot(x,y,"go")
#plt.plot(x,y,"ro",markersize=4)
plt.show()
```







Example: A single spin in a magnetic field

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \qquad \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

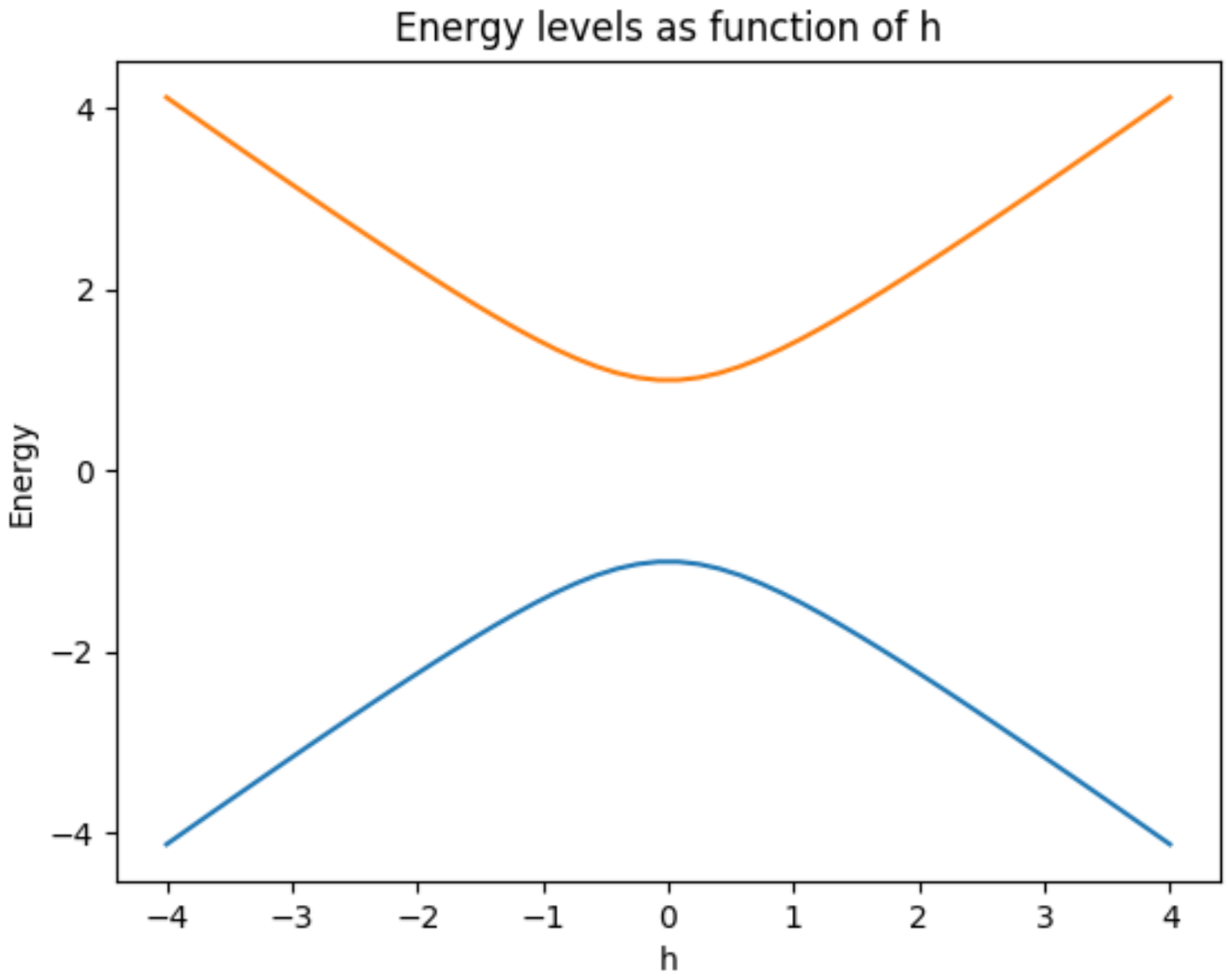
$$H = \sigma_z + h\,\sigma_x \quad = \begin{pmatrix} 1 & h \\ h & -1 \end{pmatrix}$$

The EIGENVALUEs of H are the energy levels

```
import numpy as np
import pylab as plt

sigmax=np.array([[0,1],[1,0]])
sigmaz=np.array([[1,0],[0,-1]])
evallist=[]
x=[]
y=[]
z=[]

for h in np.linspace(-4,4,50):
    hamil=sigmaz+h*sigmax
    eval1=np.linalg.eigvalsh(hamil)
    x.append(h)
    y.append(eval1[0])
    z.append(eval1[1])
    evallist.append([h,eval1[0],eval1[1]])
np.savetxt('evalszsx.dat',evallist,fmt="%2.5f")
plt.plot(x,y)
plt.plot(x,z)
plt.xlabel('h')
plt.ylabel('Energy')
plt.title('Energy levels as function of h')
plt.show()
```

Python — more evalszsx.dat — 80×24

```
-4.00000 -4.12311 4.12311
-3.83673 -3.96491 3.96491
-3.67347 -3.80715 3.80715
-3.51020 -3.64987 3.64987
-3.34694 -3.49314 3.49314
-3.18367 -3.33703 3.33703
-3.02041 -3.18165 3.18165
-2.85714 -3.02709 3.02709
-2.69388 -2.87350 2.87350
-2.53061 -2.72103 2.72103
-2.36735 -2.56989 2.56989
-2.20408 -2.42033 2.42033
-2.04082 -2.27265 2.27265
-1.87755 -2.12725 2.12725
-1.71429 -1.98463 1.98463
-1.55102 -1.84544 1.84544
-1.38776 -1.71052 1.71052
-1.22449 -1.58094 1.58094
-1.06122 -1.45815 1.45815
-0.89796 -1.34400 1.34400
-0.73469 -1.24088 1.24088
-0.57143 -1.15175 1.15175
-0.40816 -1.08009 1.08009
evalszsx.dat
```


Energy levels as function of h
Energy
h
4
2
0
−2
−4
−4
−3
−2
−1
0
1
2
3
4

Doing it in 2 steps

```
!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Oct  6 11:44:00 2017
Hamiltonian: \sigma_z+ h* \sigma_x
@author: arul
"""
import numpy as np
import pylab as plt

sigmax=np.array([[0,1],[1,0]])
sigmaz=np.array([[1,0],[0,-1]])
evallist=[]

for h in np.linspace(-4,4,50):
    hamil=sigmaz+h*sigmax
    eval1=np.linalg.eigvalsh(hamil)
    evallist.append([h,eval1[0],eval1[1]])
np.savetxt('evalszsx_1.dat',evallist,fmt="%2.5f")
```

Creates data in the form of 3 columns containing h, eval(0), and eval(1).

Saves the data in a file called ‘evalszsx_1.dat’

Note the format “%2.5f”

loadtxt

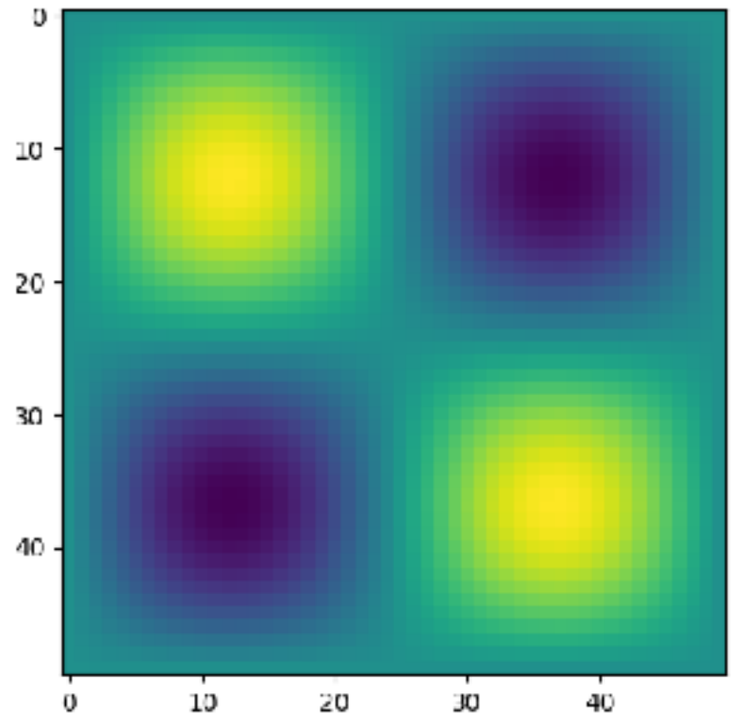
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Oct  6 11:44:00 2017
Hamiltonian: \sigma_z+ h* \sigma_x
@author: arul
"""
import numpy as np
import pylab as plt



x,y,z=np.loadtxt('evalszsx_1.dat',unpack='True')
plt.plot(x,y)
plt.plot(x,z)
plt.xlabel('h')
plt.ylabel('Energy')
plt.title('Energy levels as function of h')
plt.show()
```

Without “unpacking”

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Oct  6 11:44:00 2017
Hamiltonian: \sigma_z+ h* \sigma_x
@author: arul
"""
import numpy as np
import pylab as plt


data=np.loadtxt('evalszsx_1.dat')
plt.plot(data[:,0],data[:,1])
plt.plot(data[:,0],data[:,2])
plt.xlabel('h')
plt.ylabel('Energy')
plt.title('Energy levels as function of h')
plt.show()
```

2-D density plot

```
##Plotting sin(x)*sin(y) as a density plot
##
import pylab as plt
import numpy as np

x=np.linspace(0,2*np.pi) #Defualt is 50 values
z=np.array([np.sin(xx)*np.sin(yy) for xx in x for yy in x])
z=z.reshape((50,50))

#z=[]
#for i in x:
#    for j in x:
#        z.append(np.sin(i)*np.sin(j))
#z=np.array(z).reshape((50,50))

plt.imshow(z)
plt.show()
```

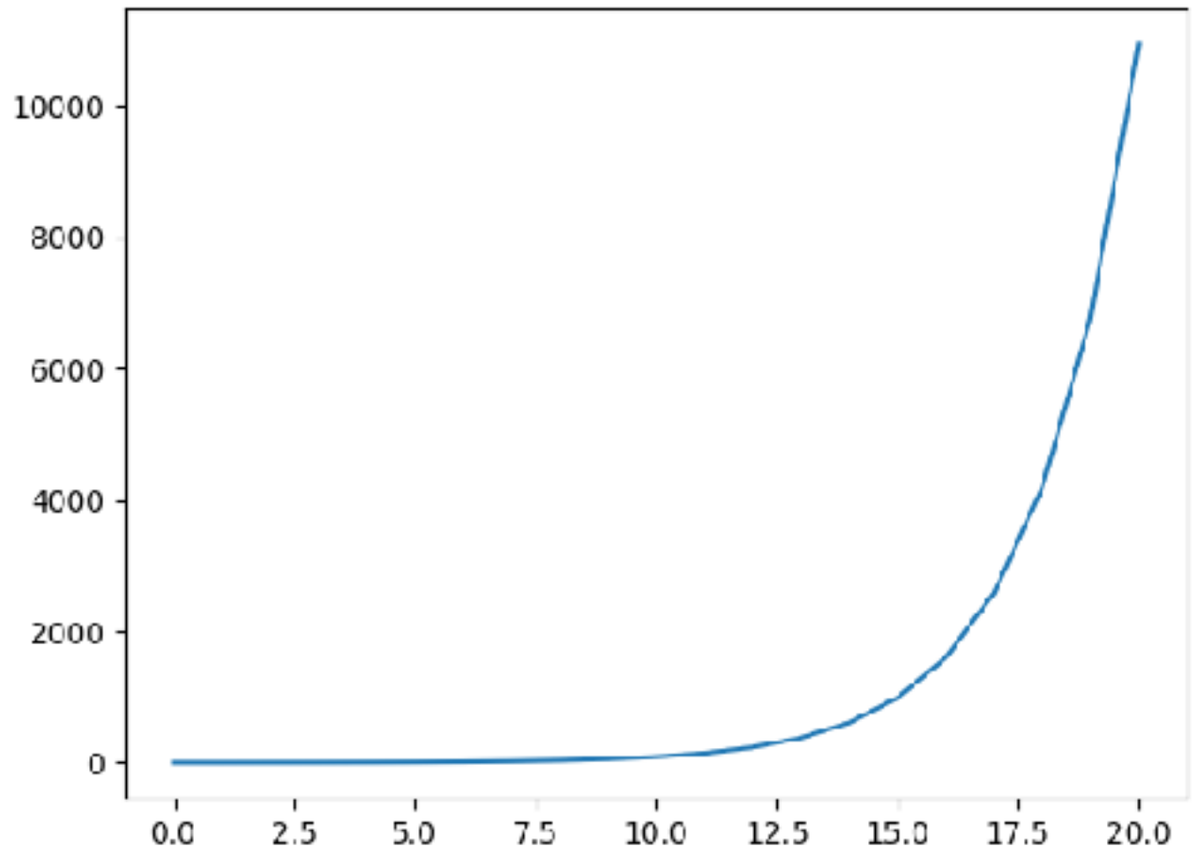


Saving figures directly

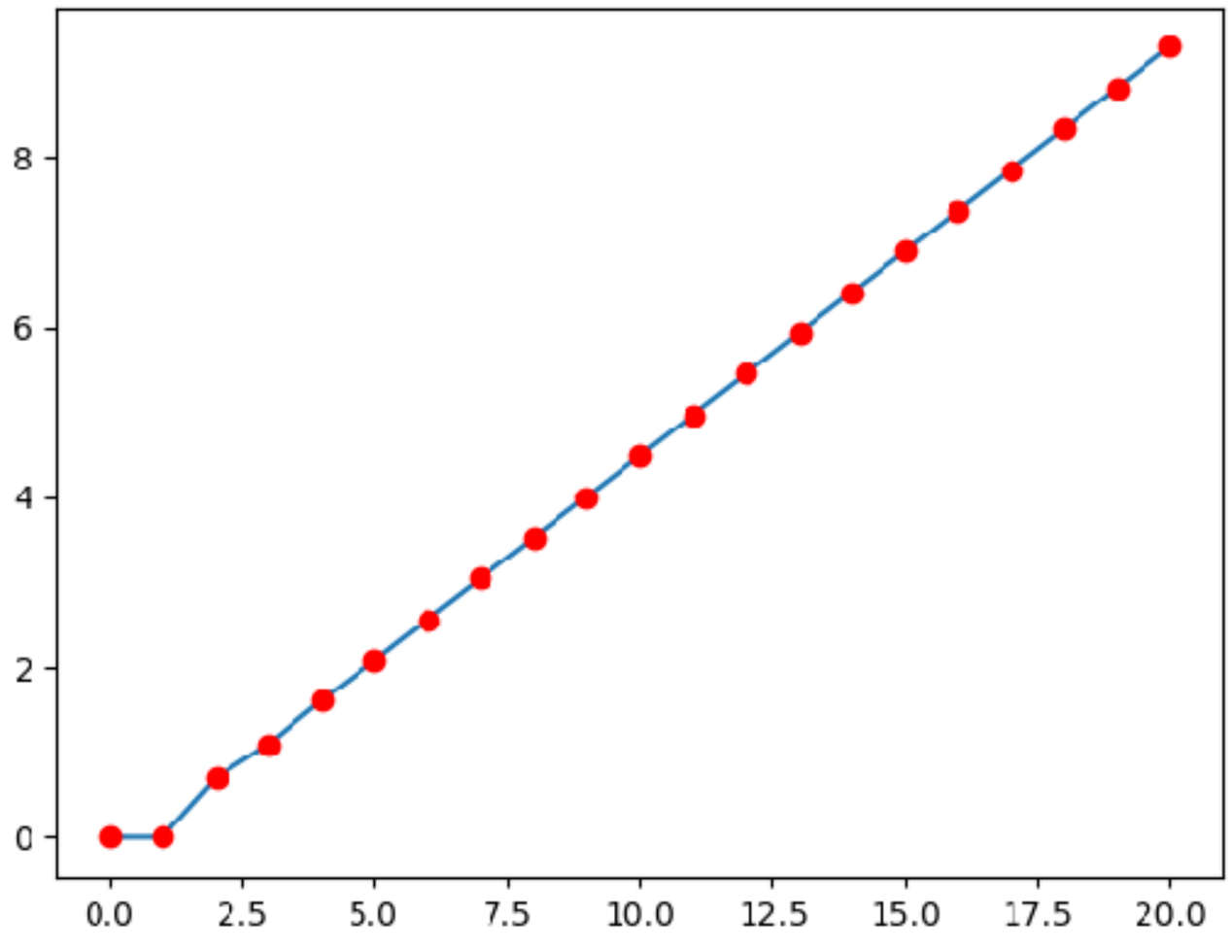
```
import pylab as plt
import numpy as np

x=np.linspace(0,2*np.pi)
#y=x+2*np.random.random(len(x))
plt.plot(x,np.sin(x))
plt.plot(x,np.cos(x),"ro")
plt.xlabel('x')
plt.ylabel('sin(x), cos(x)')
plt.savefig('foo.pdf')
#plt.show()
```

Revisiting Fibonacci

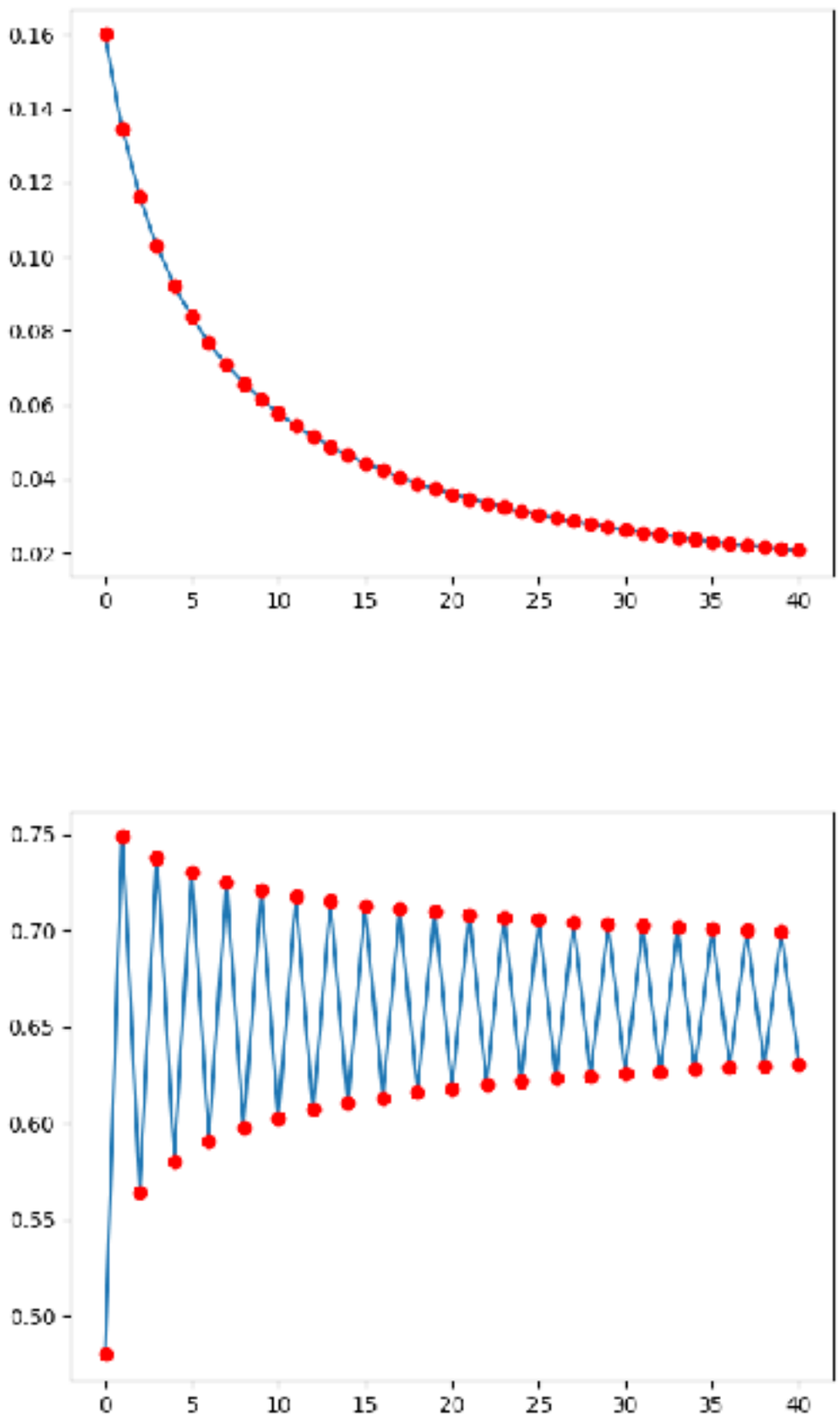
```
# Fibonacci numbers module, also plotting it

import numpy as np
import pylab as plt

def fib2(n):   # return Fibonacci series up to n
    result = []
    a, b = 0, 1
    for i in range(n+1):
        result.append((i,b))
        a, b = b, a+b
    data=np.array(result)
    plt.plot(data[:,0],data[:,1])
    plt.show()
```

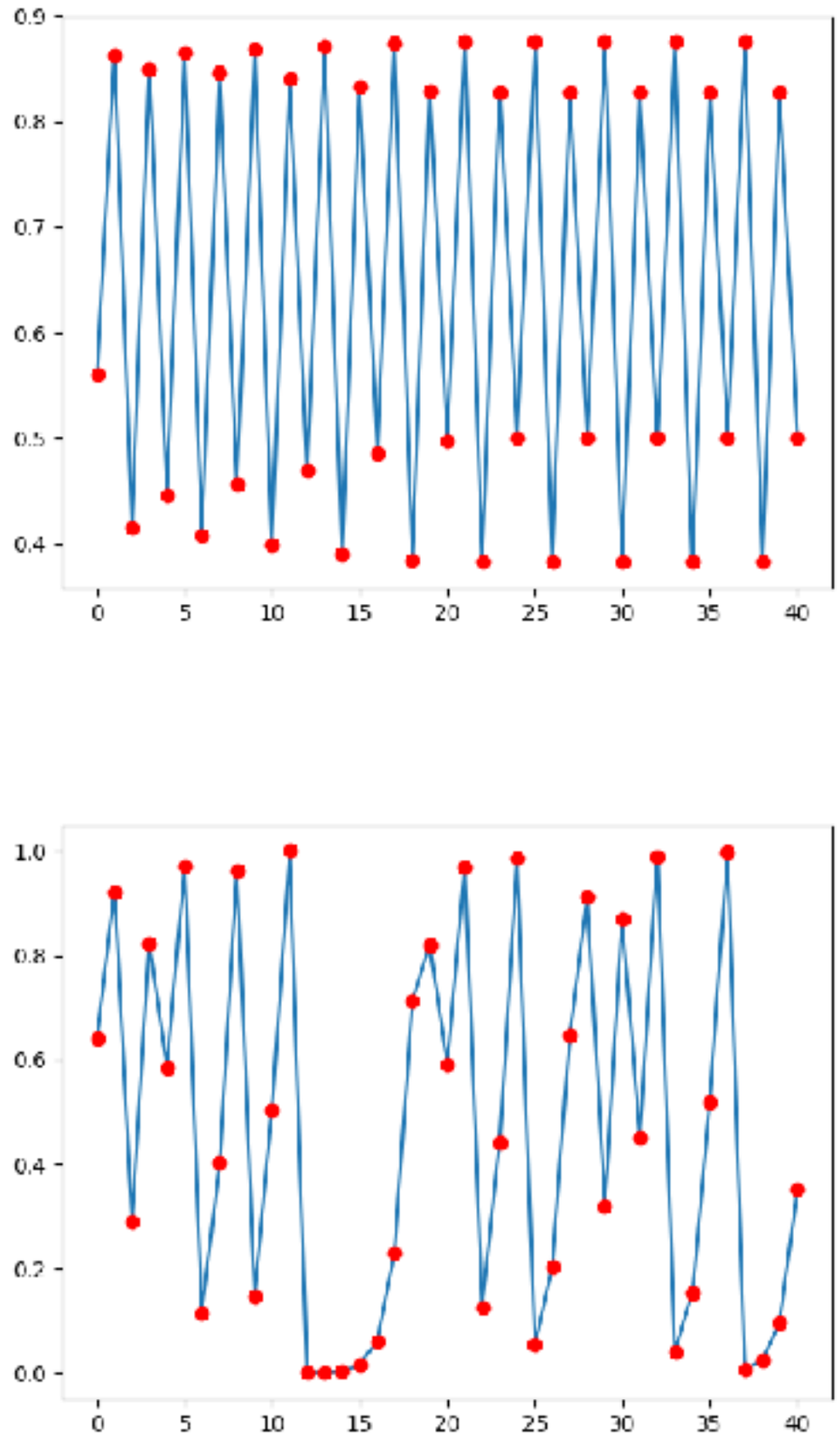


Revisiting Fibonacci: Exponential growth



Revisiting Logistic map: r=1, 3, 3.5, 4





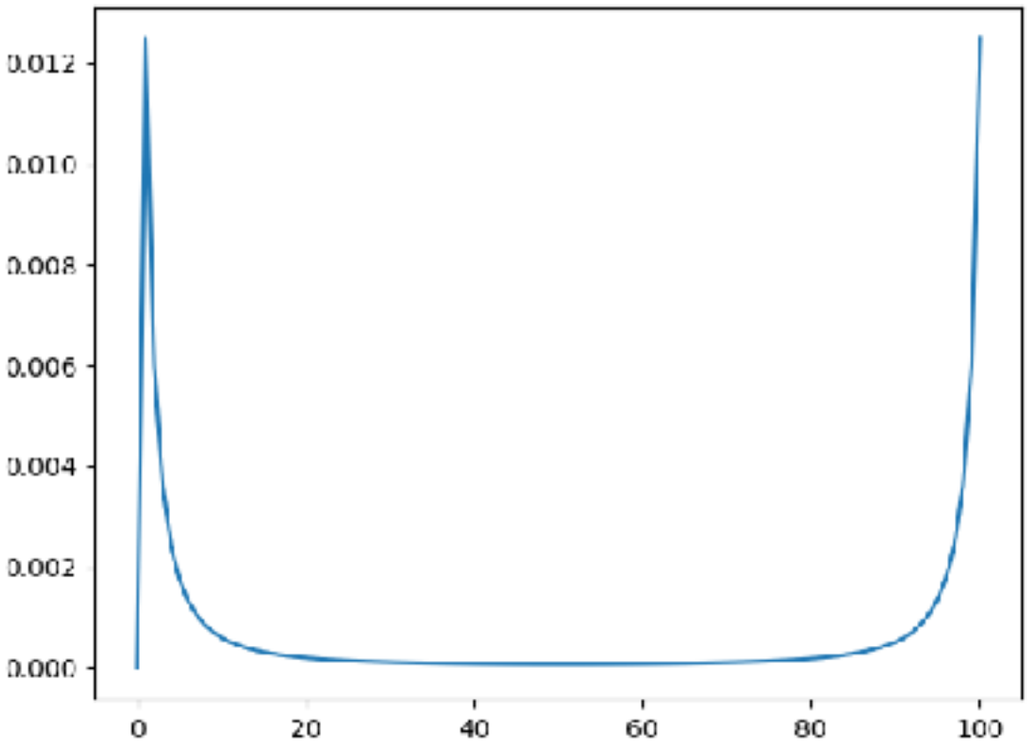
Fourier Transform DFT, and FFT

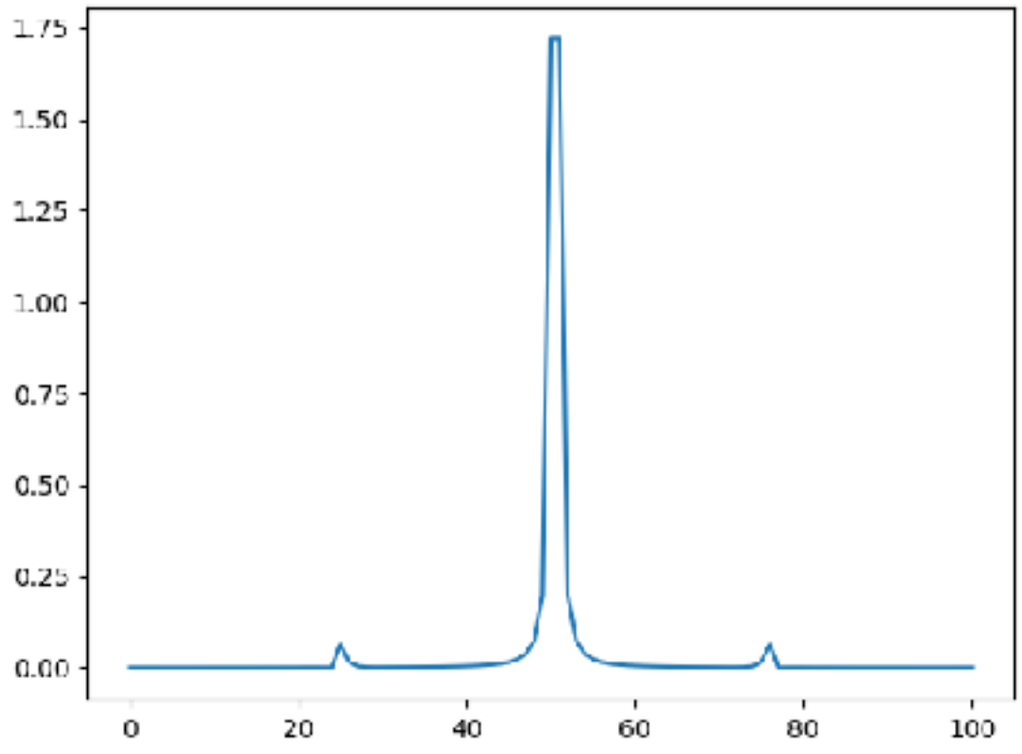
$$A_k = \sum_{m=0}^{n-1} a_m \exp\left[-2\pi i \frac{mk}{n}\right], \quad k = 0, \ldots, n-1.$$

$$a_m = \frac{1}{n} \sum_{k=0}^{n-1} A_k \exp\left[2\pi i \frac{mk}{n}\right], \quad m = 0, \ldots, n-1.$$

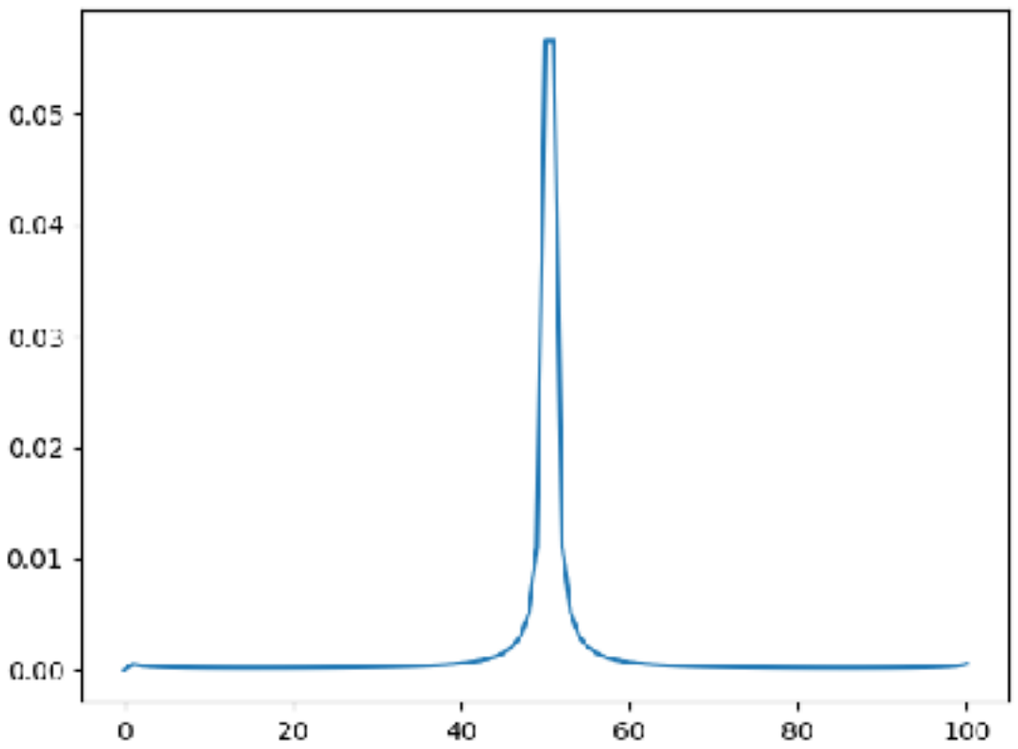
FFT of logistic map data

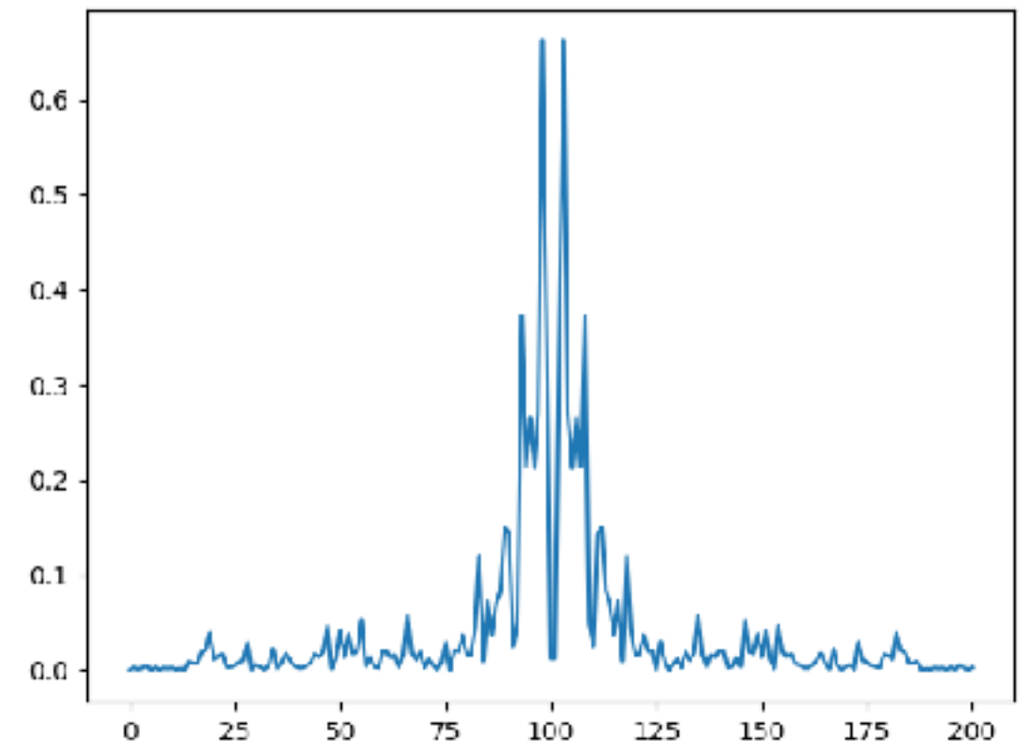
```
# FFT of logistic map
import numpy as np
import pylab as plt
def logistic_mapfft(r,x,n):
    xn=[]
    for i in range(n+1):
        #print(x,end=',')
        x=r*x*(1-x)
        xn.append(x)
    xna=np.array(xn)
    xn=xna-sum(xna)/len(xn) ##Remove "DC" component.
    xnf=np.fft.fft(xn,norm='ortho')
    plt.plot(np.abs(xnf)**2)
    #plt.plot(xn,"ro")
    plt.show()
```


0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
20
40
60
80
100

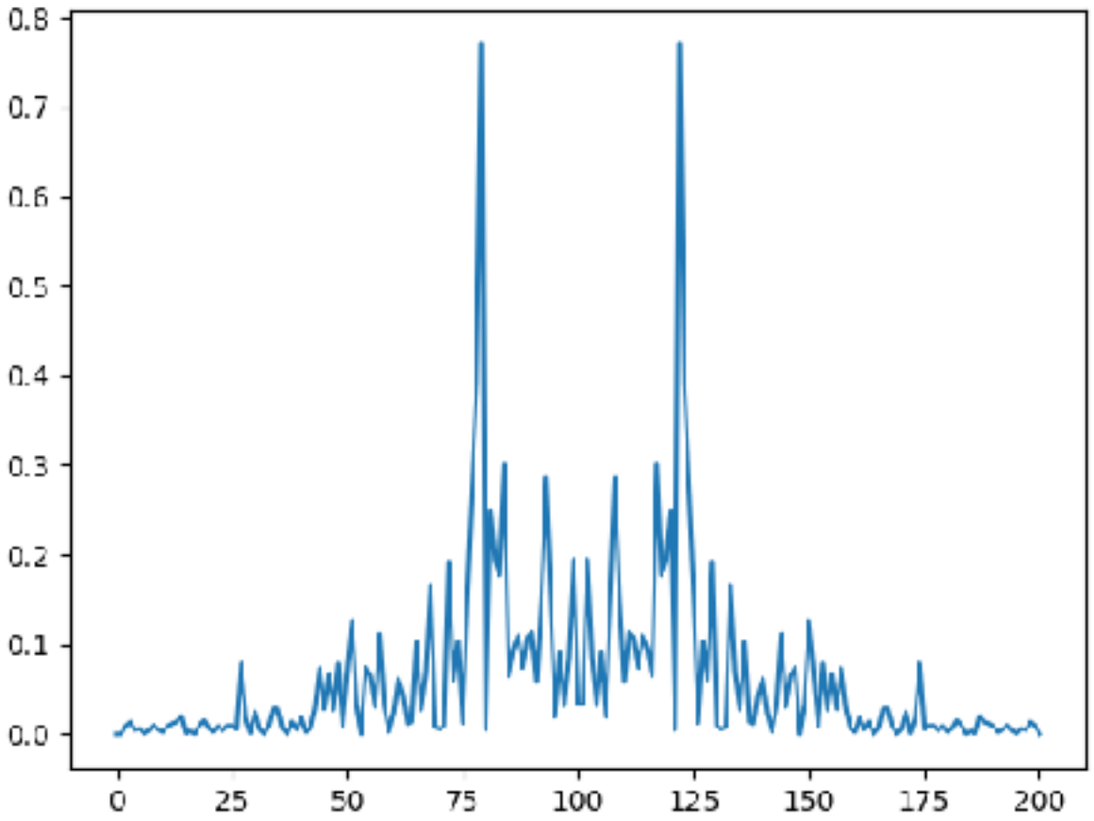

1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
20
40
60
80
100

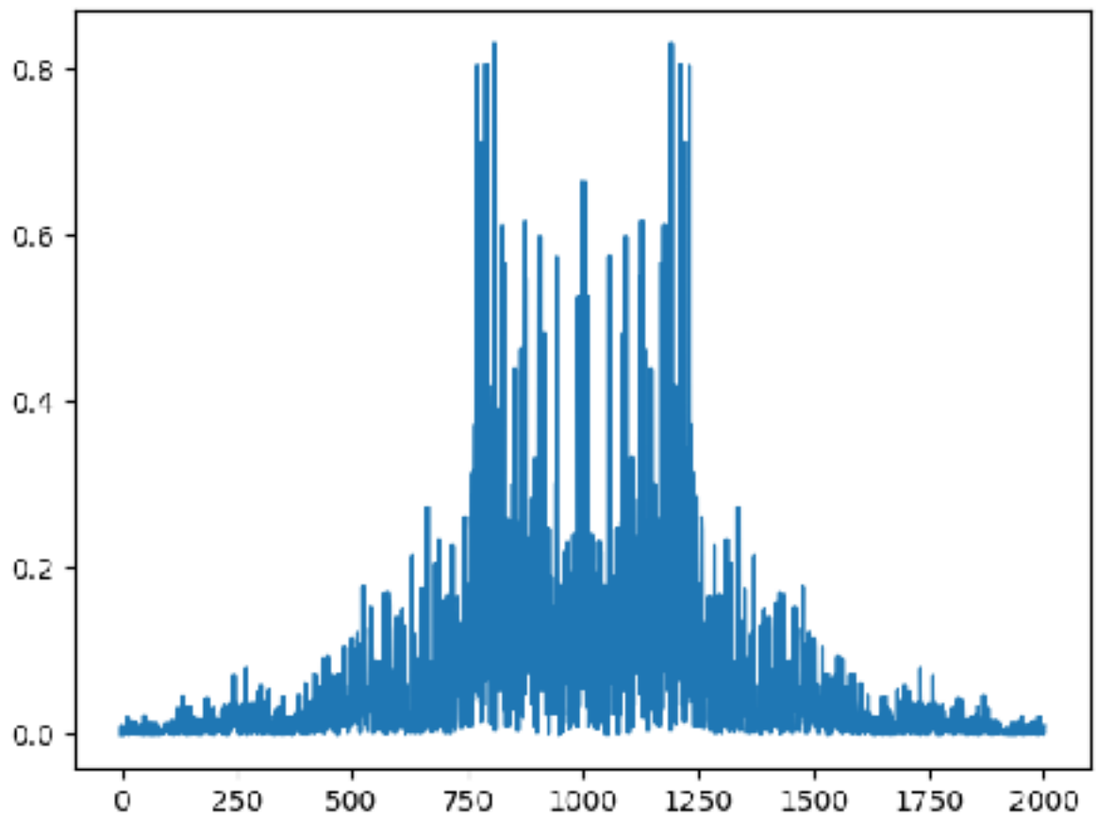
r=3, 3.5, 3.6, 3.7


0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100

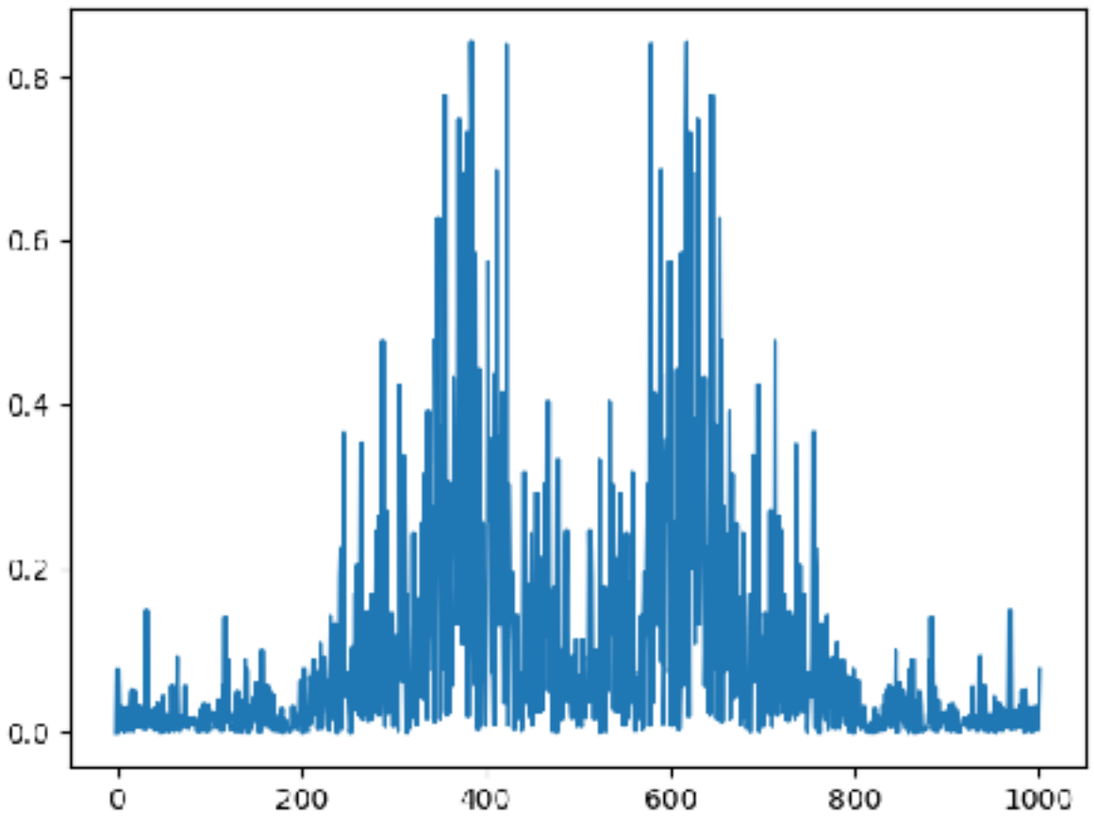

0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
25
50
75
100
125
150
175
200

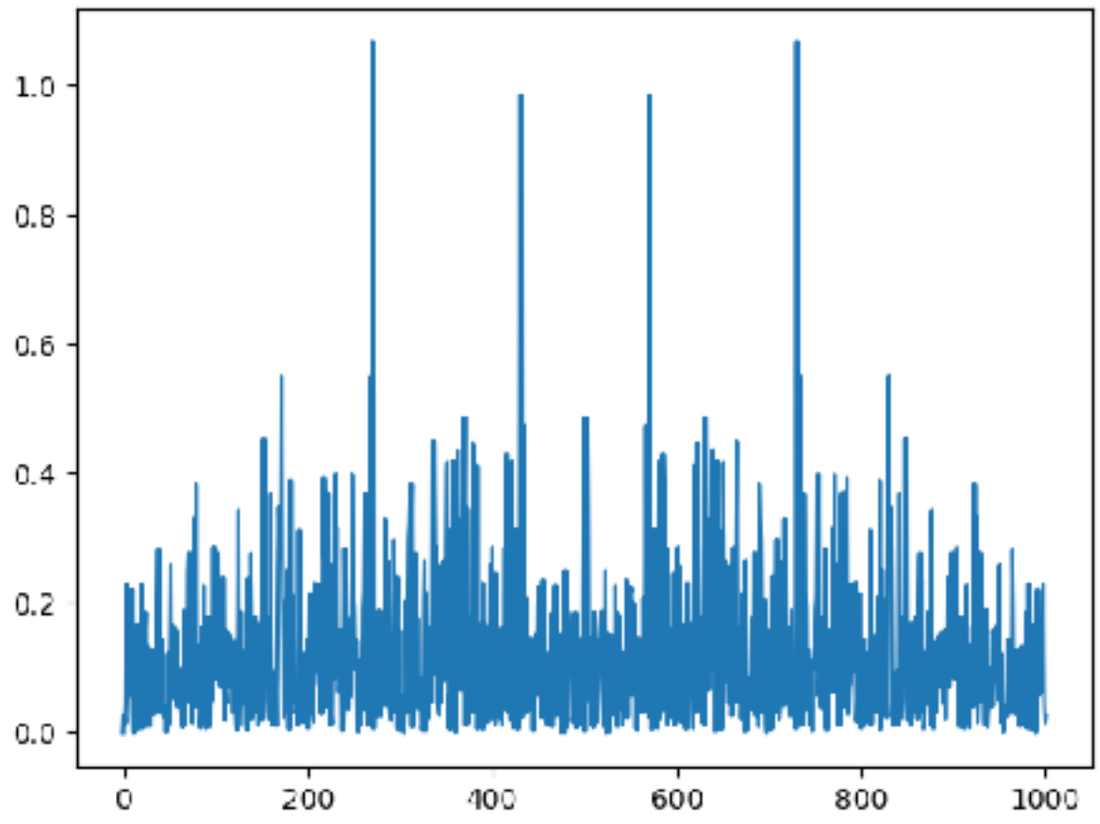
r=3.8 for n=200, 2000

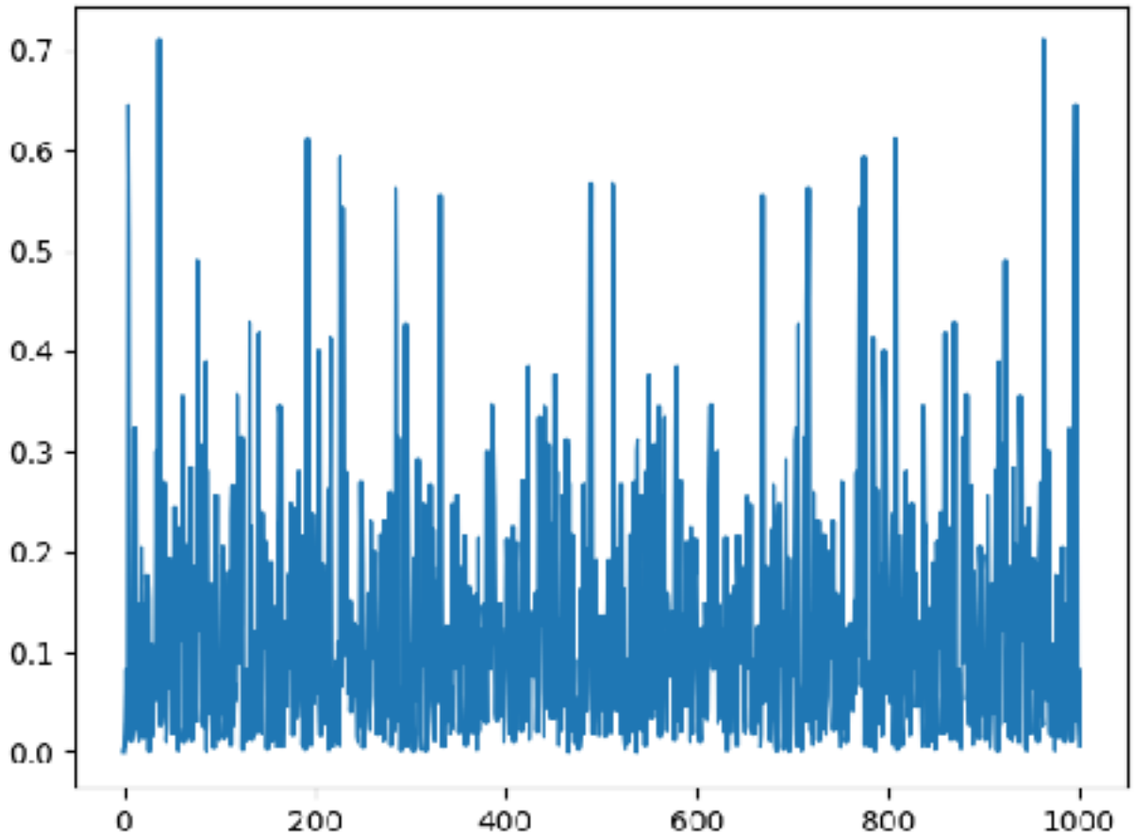




r=3.9, r=4







r=4, another orbit of length 1000

Random unif [0,1}