chaos_prelab

October 13, 2022

[29]:

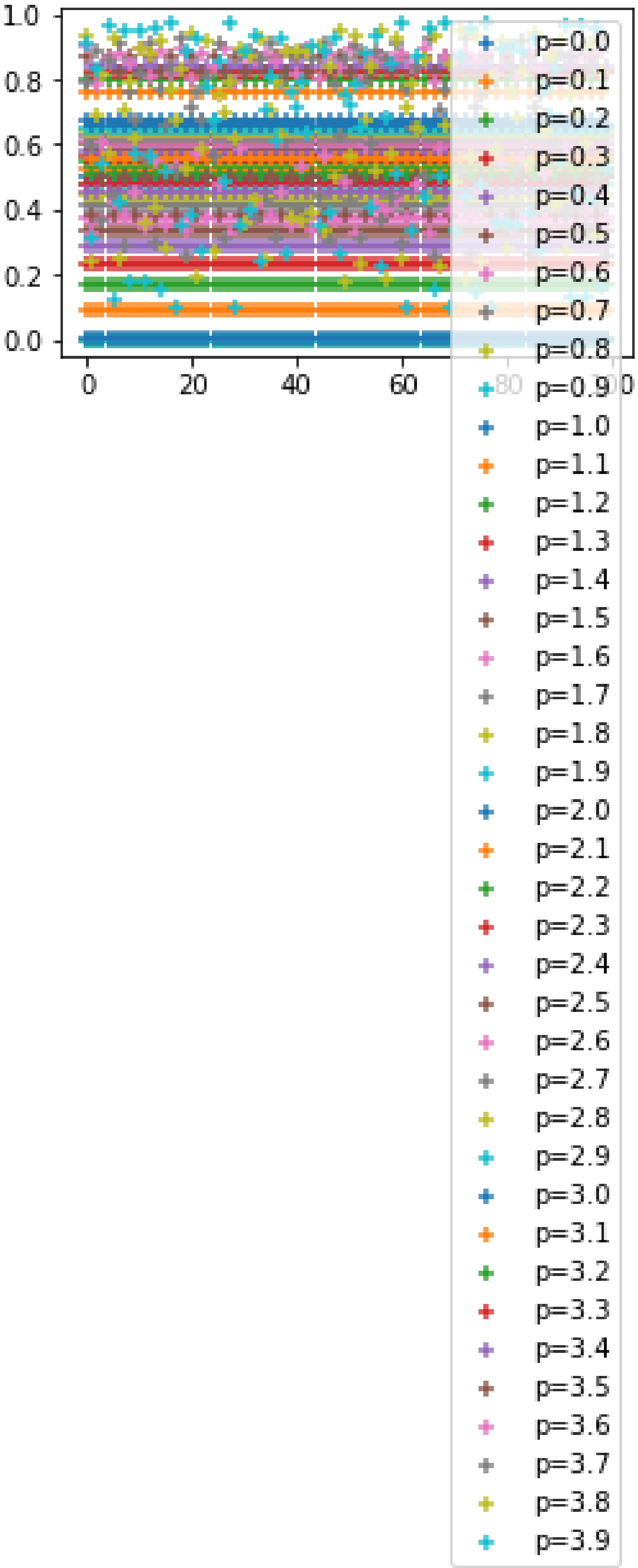
```
import numpy as np
import matplotlib.pyplot as plt
```

Using the relationship $x_{n+1} = p \cdot (x_n)(1 - x_n)$

Iterated through values of p from 0 to 4 in increments of 0.1. Performed 1000 iterations and plotted the next 100 iterations as +'s in the plot below.

[30]:

```
for p in np.arange(0, 4, 0.1):
  x = np.random.rand()
  for i in range(1000):
    x = p*x*(1-x)

  xs = []
  for i in range(100):
    x = p*x*(1-x)
    xs.append(x)

  plt.scatter(np.arange(len(xs)), xs,marker='+', label='p={:.1f}'.format(p))
plt.legend()
plt.show()
```

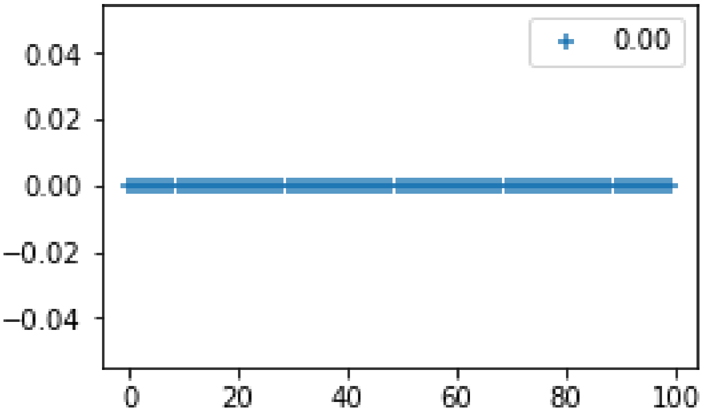

1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
p=0.0
p=0.1
p=0.2
p=0.3
p=0.4
p=0.5
p=0.6
p=0.7
p=0.8
p=0.9
p=1.0
p=1.1
p=1.2
p=1.3
p=1.4
p=1.5
p=1.6
p=1.7
p=1.8
p=1.9
p=2.0
p=2.1
p=2.2
p=2.3
p=2.4
p=2.5
p=2.6
p=2.7
p=2.8
p=2.9
p=3.0
p=3.1
p=3.2
p=3.3
p=3.4
p=3.5
p=3.6
p=3.7
p=3.8
p=3.9

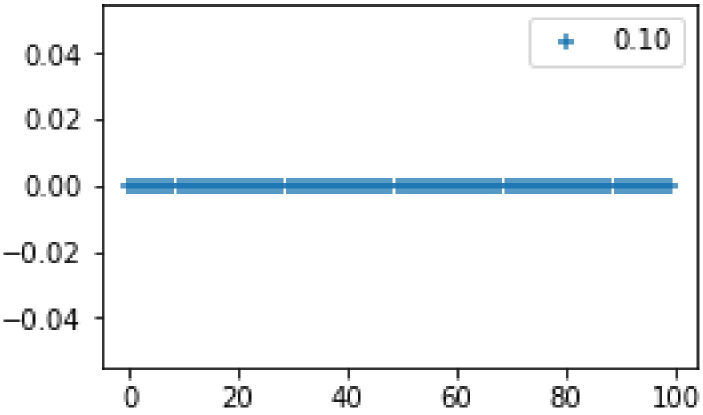
This above graph is a little difficult to see.

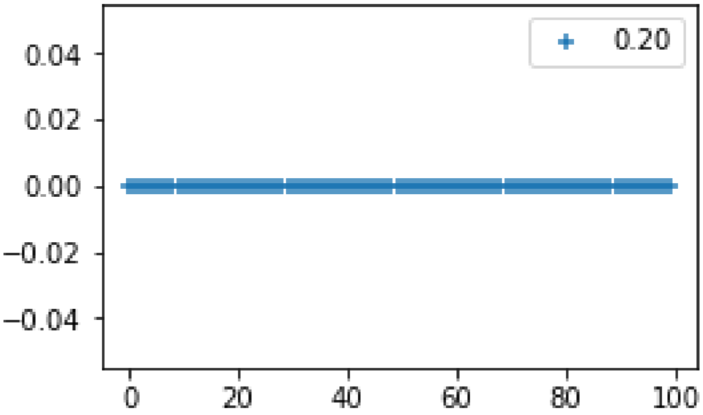
Plot a few cases first to try and get better understanding of the problem

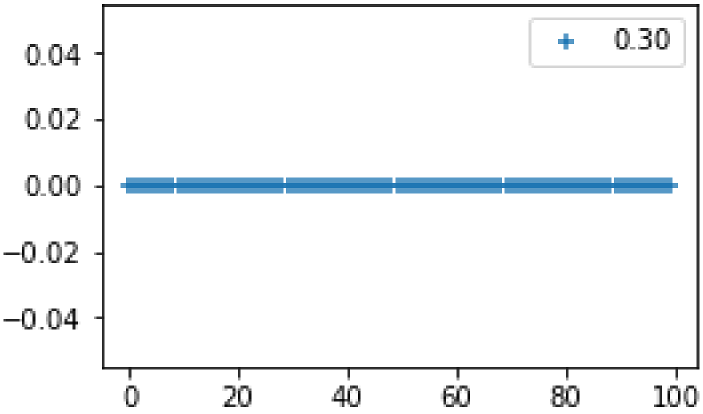
[31]:
```
def generate_plot(p):

  x = np.random.rand()
  for i in range(1000):
    x = p*x*(1-x)

  xs = []
  for i in range(100):
    x = p*x*(1-x)
    xs.append(x)
  plt.scatter(np.arange(len(xs)), xs,marker='+', label='{:.2f}'.format(p))
  # plt.title('p={:.1f}'.format(p))
```

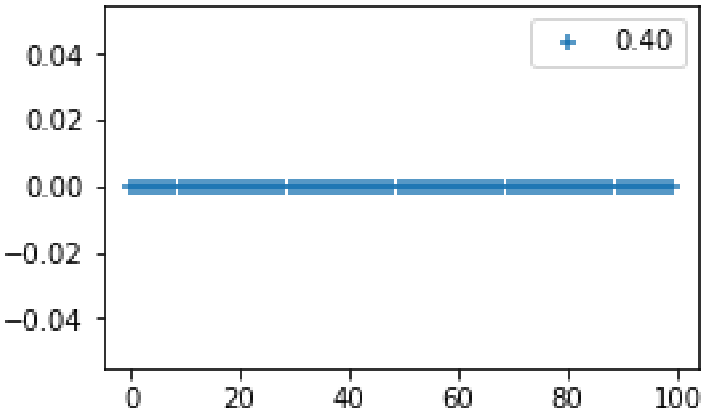
[32]:
```
for i in np.arange(0, 4, 0.1):
  generate_plot(i)
  plt.legend()
  plt.show()
```

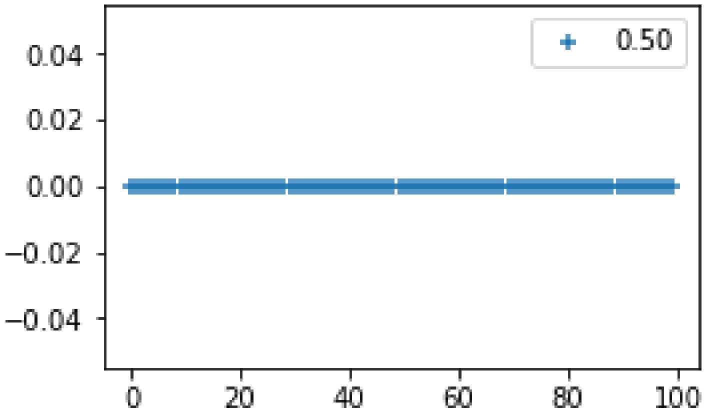


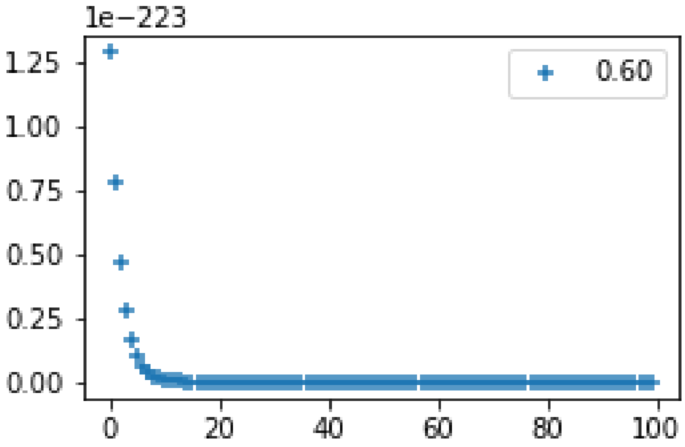

0.10
0.04
0.02
0.00
−0.02
−0.04
0
20
40
60
80
100

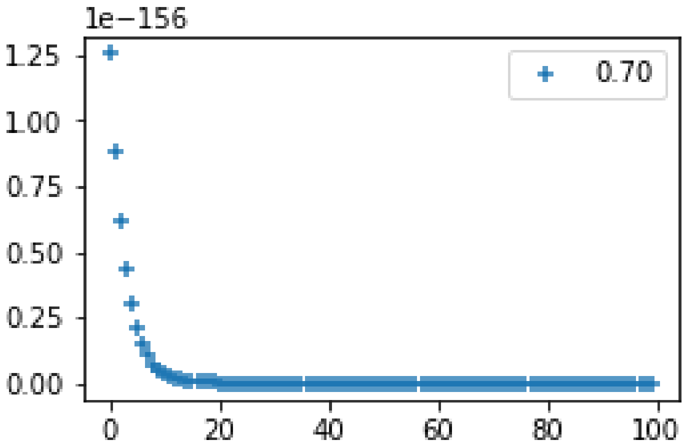

0.20
0.04
0.02
0.00
−0.02
−0.04
0
20
40
60
80
100

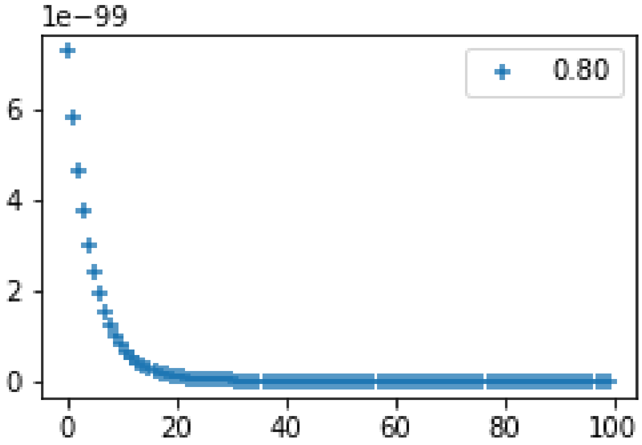

0.30
0.04
0.02
0.00
−0.02
−0.04
0
20
40
60
80
100

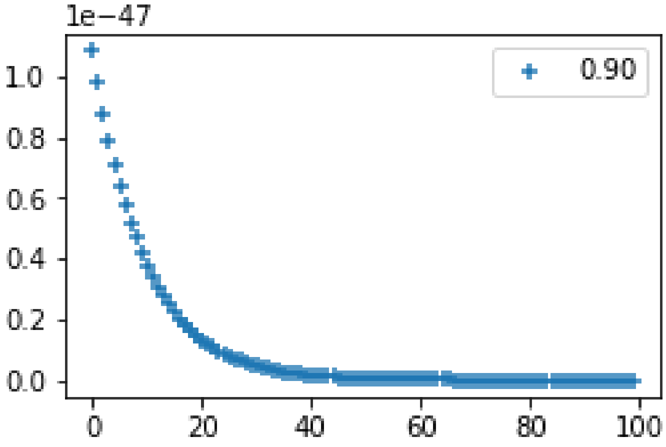

0.40
0.04
0.02
0.00
−0.02
−0.04
0
20
40
60
80
100

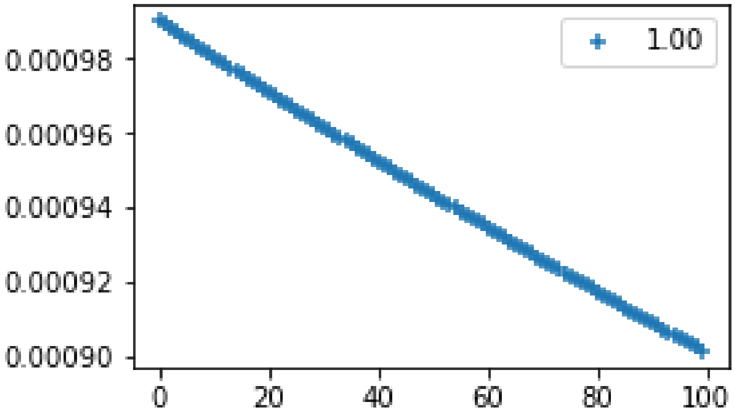

0.50
0.04
0.02
0.00
−0.02
−0.04
0
20
40
60
80
100

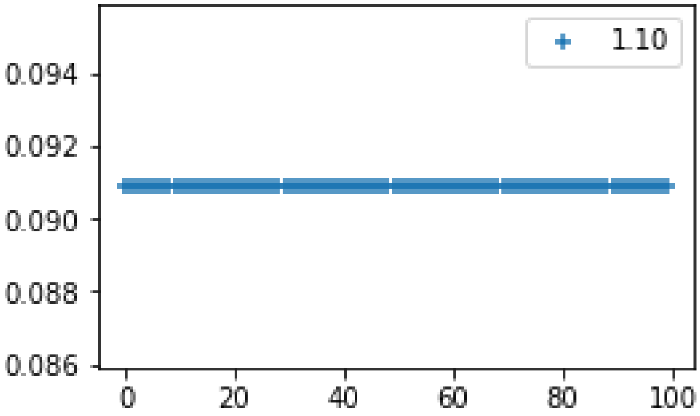

1e−223
0.60
1.25
1.00
0.75
0.50
0.25
0.00
0
20
40
60
80
100

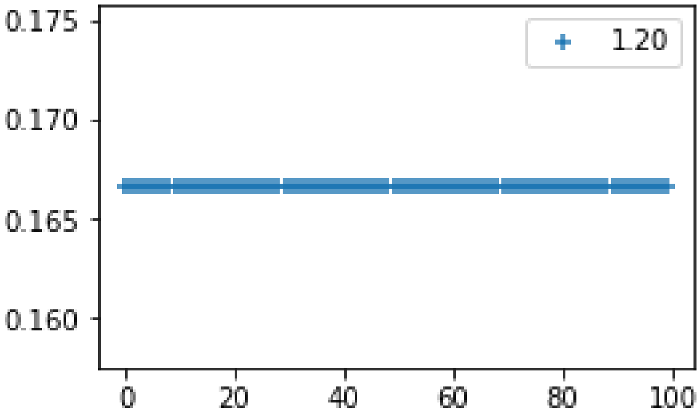

1e−156
1.25
1.00
0.75
0.50
0.25
0.00
0
20
40
60
80
100
+ 0.70

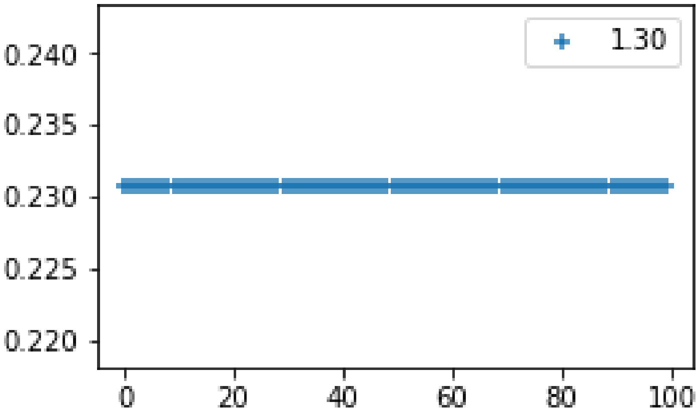

1e−99
6
4
2
0
0
20
40
60
80
100
+ 0.80

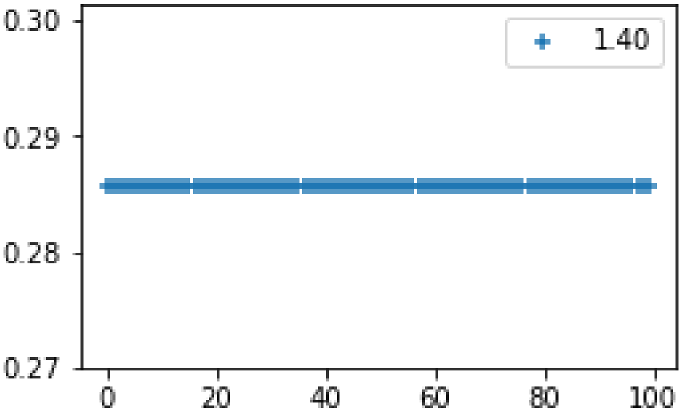

1e−47
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
+ 0.90

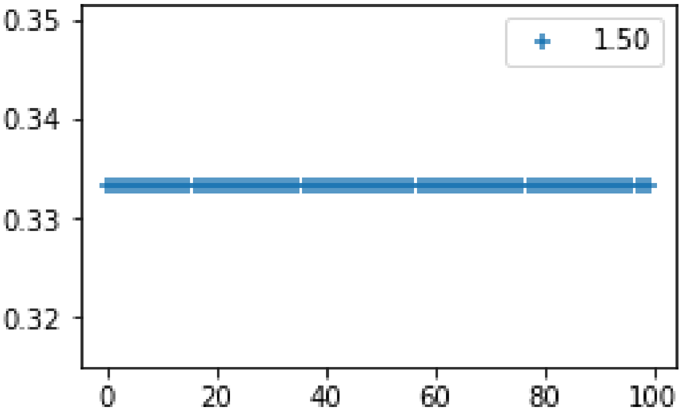

1.00
0.00098
0.00096
0.00094
0.00092
0.00090
0
20
40
60
80
100

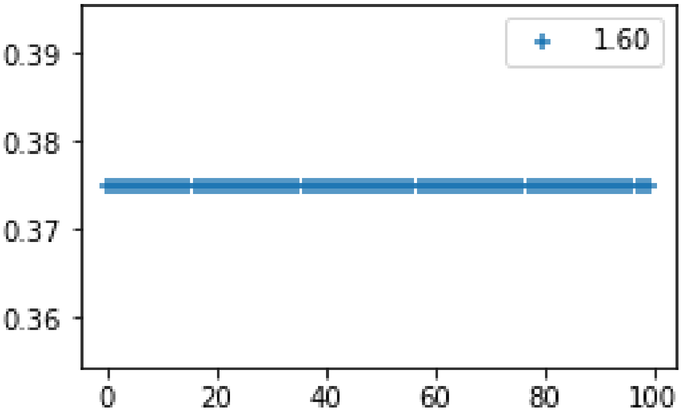

1.10
0.094
0.092
0.090
0.088
0.086
0
20
40
60
80
100

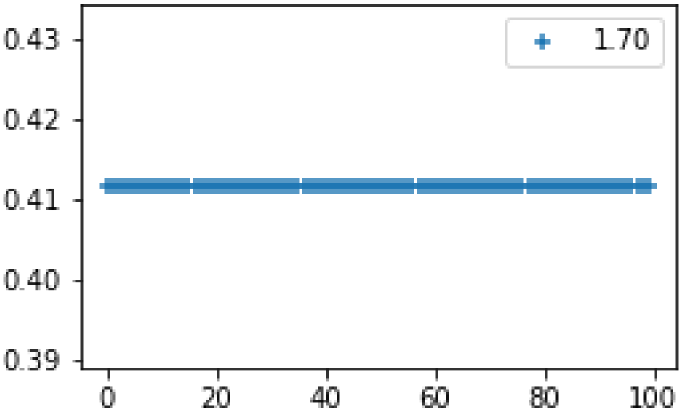

1.20
0.175
0.170
0.165
0.160
0
20
40
60
80
100

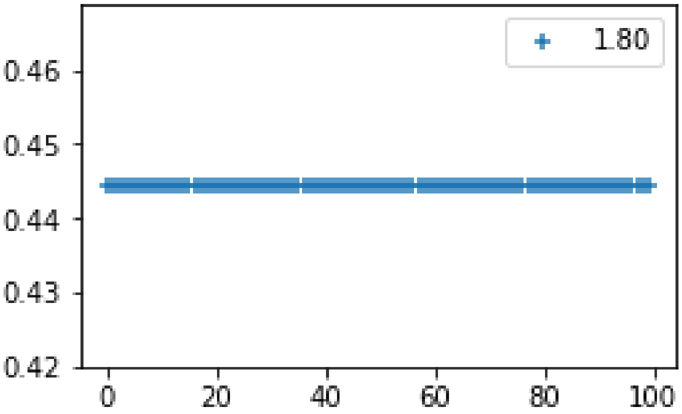

1.30
0.240
0.235
0.230
0.225
0.220
0
20
40
60
80
100

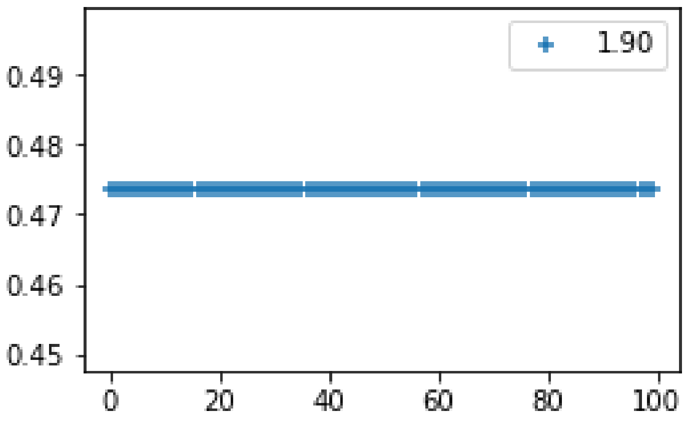

1.40
0.30
0.29
0.28
0.27
0
20
40
60
80
100

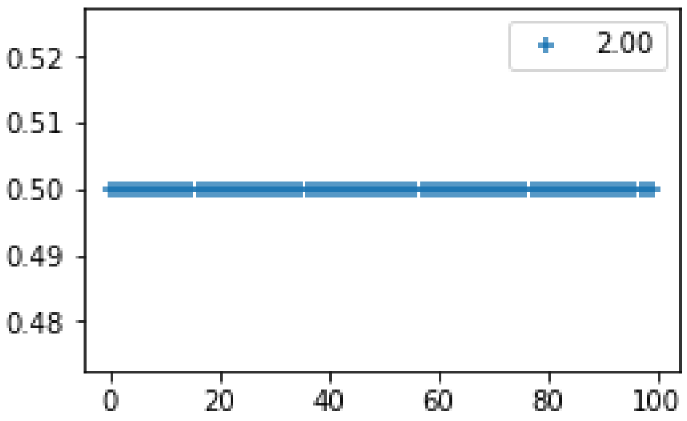

1.50
0.35
0.34
0.33
0.32
0
20
40
60
80
100

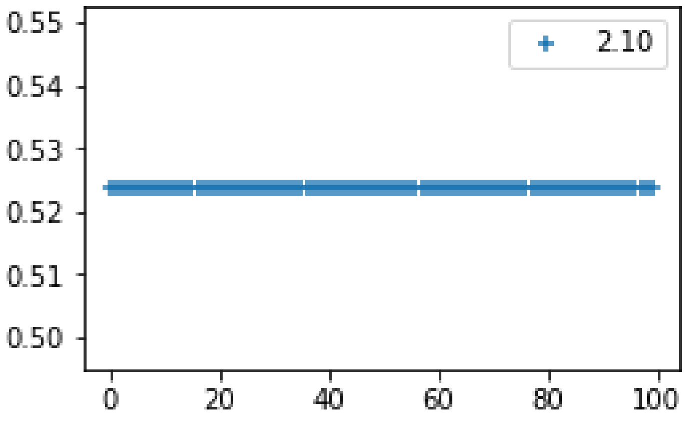

1.60
0.39
0.38
0.37
0.36
0
20
40
60
80
100

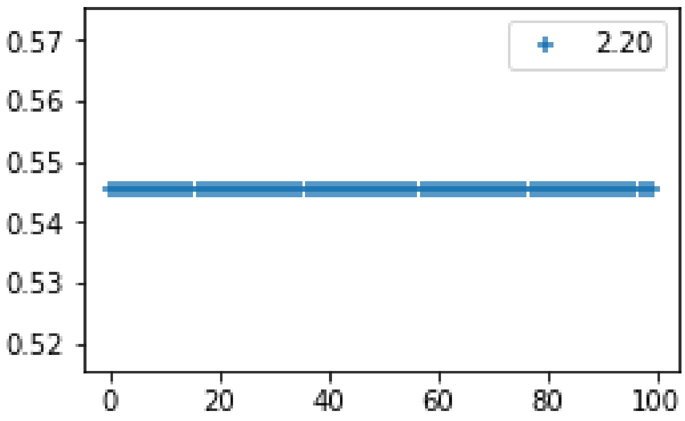

1.70
0.43
0.42
0.41
0.40
0.39
0
20
40
60
80
100

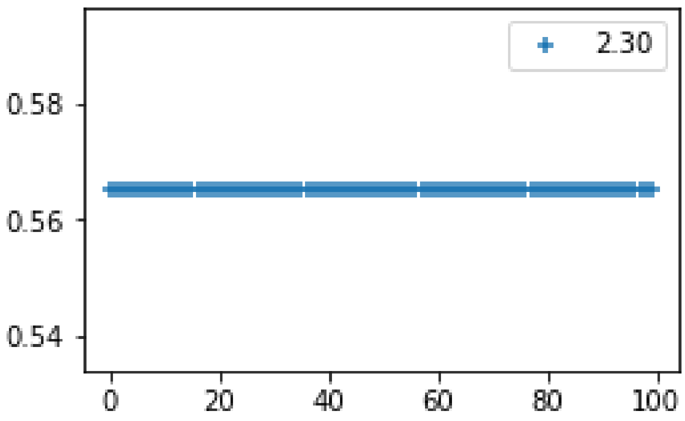

1.80
0.46
0.45
0.44
0.43
0.42
0
20
40
60
80
100

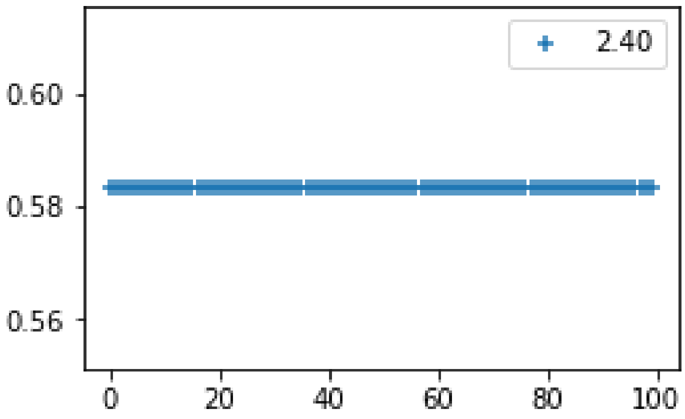

1.90
0.49
0.48
0.47
0.46
0.45
0
20
40
60
80
100

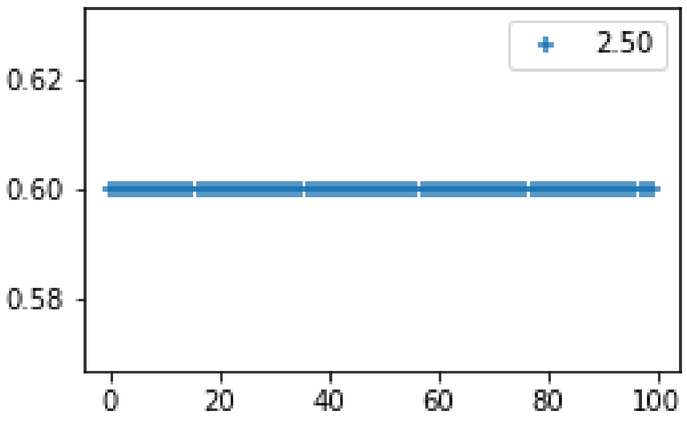

2.00
0.52
0.51
0.50
0.49
0.48
0
20
40
60
80
100

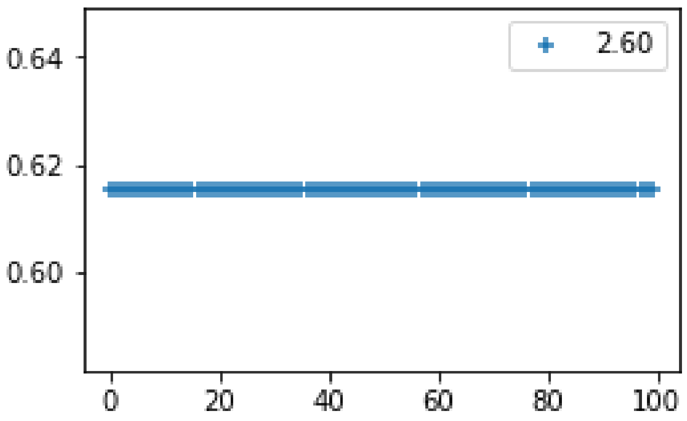

2.10
0.55
0.54
0.53
0.52
0.51
0.50
0
20
40
60
80
100

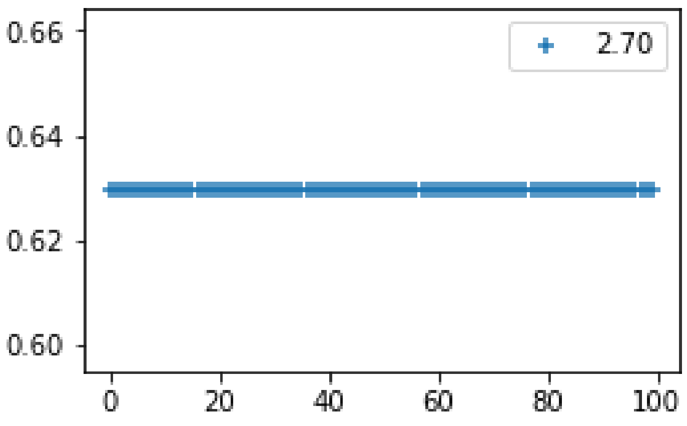

2.20
0.57
0.56
0.55
0.54
0.53
0.52
0
20
40
60
80
100

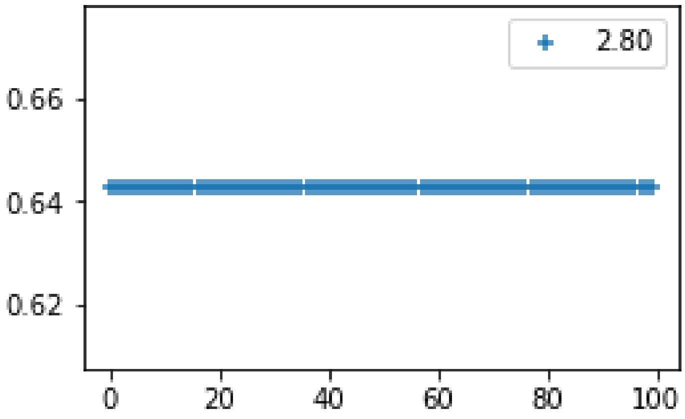

2.30
0.58
0.56
0.54
0
20
40
60
80
100

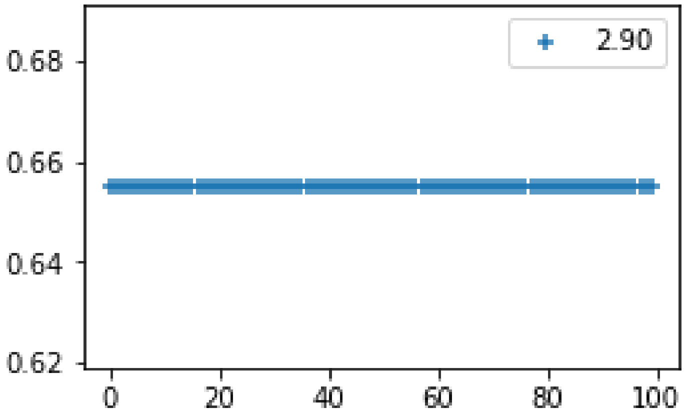

2.40
0.60
0.58
0.56
0
20
40
60
80
100

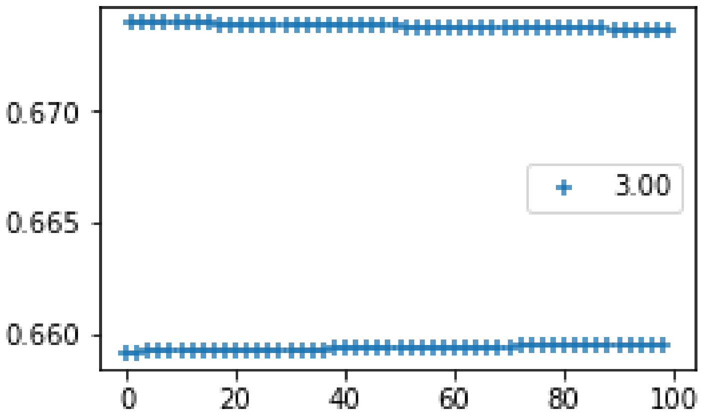

+ 2.50
0.62
0.60
0.58
0
20
40
60
80
100

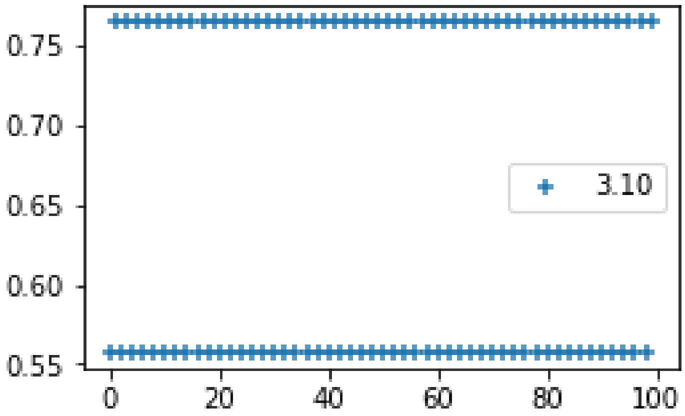

+ 2.60
0.64
0.62
0.60
0
20
40
60
80
100

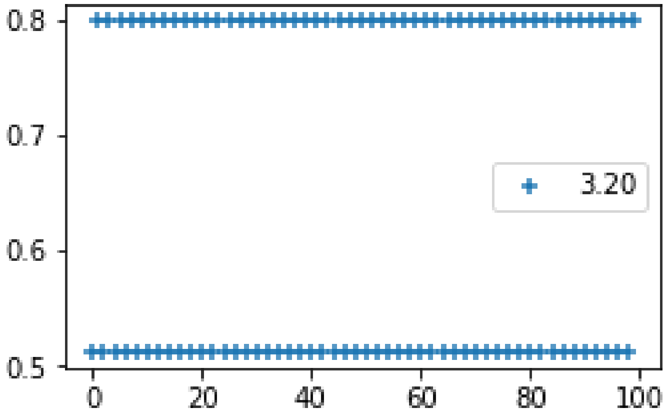

+ 2.70
0.66
0.64
0.62
0.60
0
20
40
60
80
100

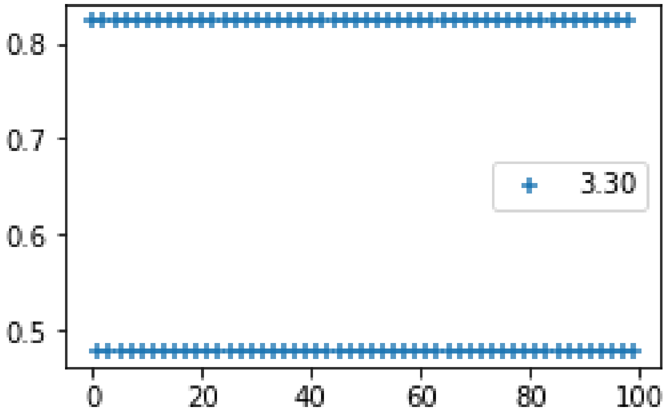

2.80
0.66
0.64
0.62
0
20
40
60
80
100

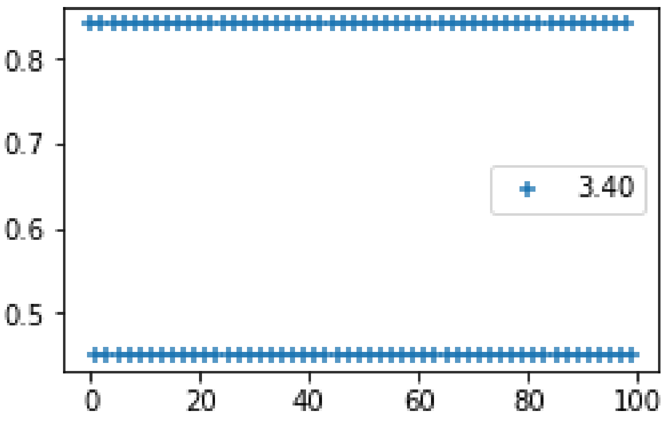

2.90
0.68
0.66
0.64
0.62
0
20
40
60
80
100

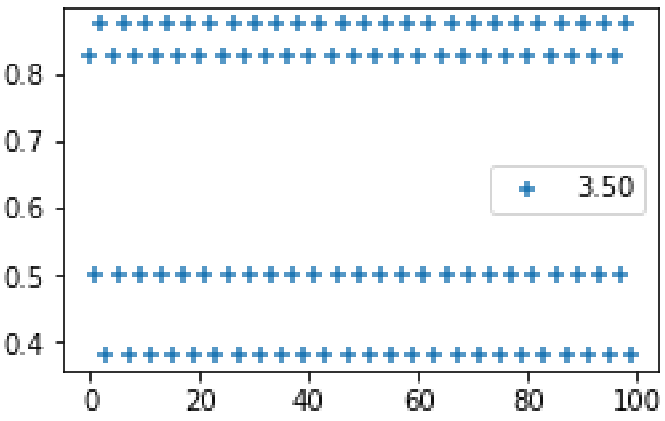

3.00
0.670
0.665
0.660
0
20
40
60
80
100

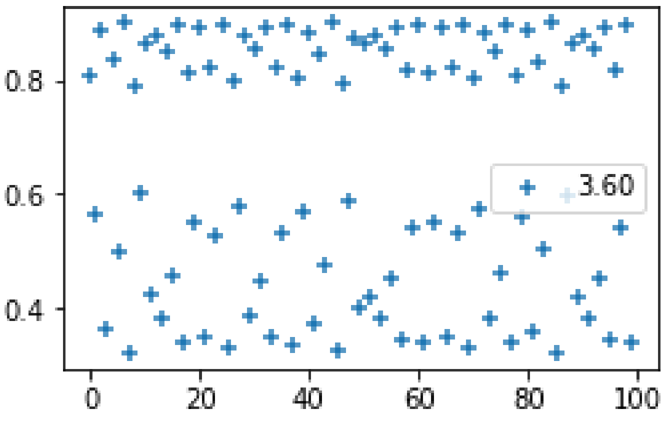

0.75
0.70
0.65
0.60
0.55
0
20
40
60
80
100
3.10

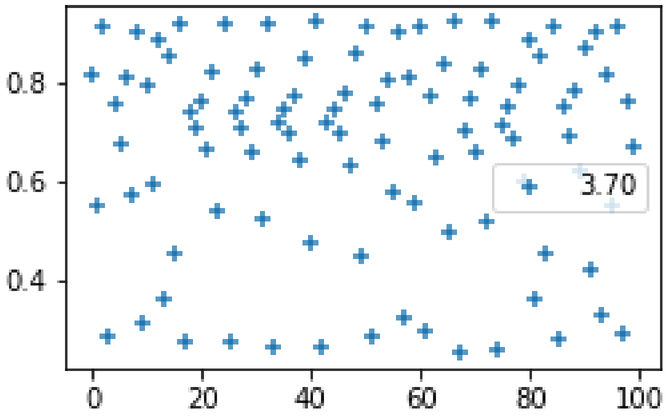

0.8
0.7
0.6
0.5
0
20
40
60
80
100
3.20

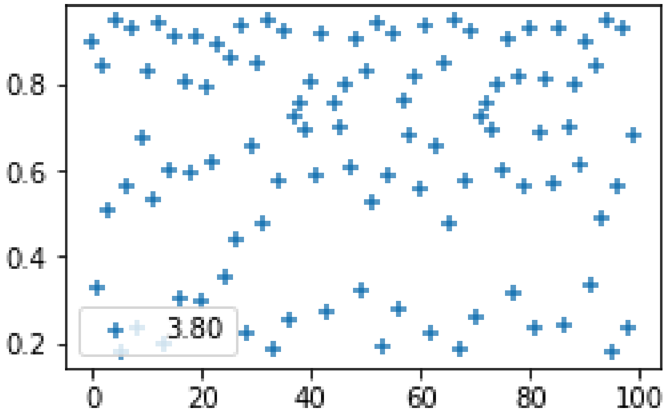

0.8
0.7
0.6
0.5
0
20
40
60
80
100
3.30

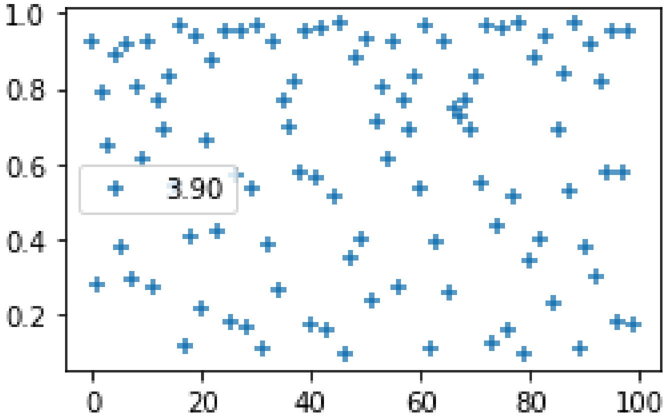

0.8
0.7
0.6
0.5
0
20
40
60
80
100
3.40


0.8
0.7
0.6
0.5
0.4
0
20
40
60
80
100
3.50


0.8
0.6
0.4
0
20
40
60
80
100
3.60

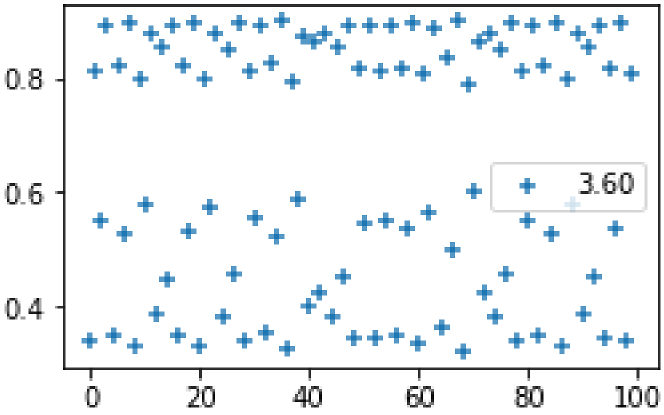


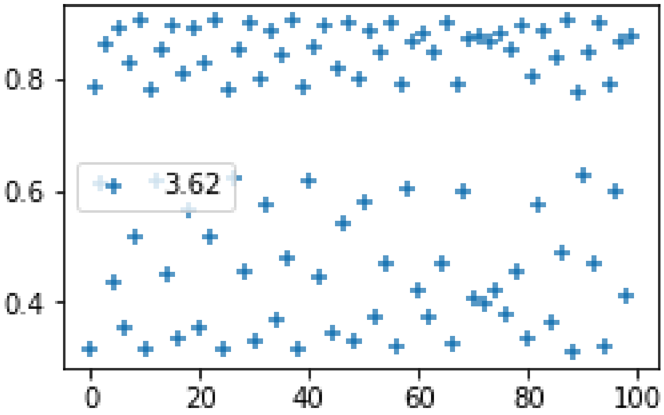


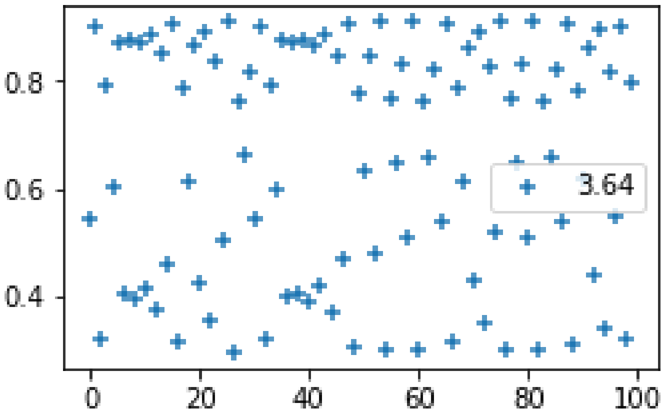


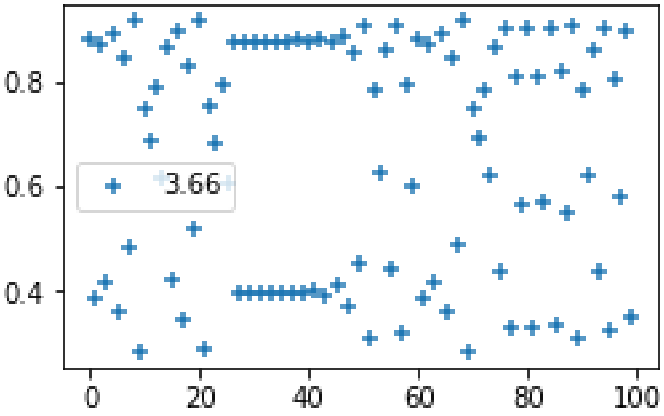
Chaotic behaviour is observed for p higher than 3.6

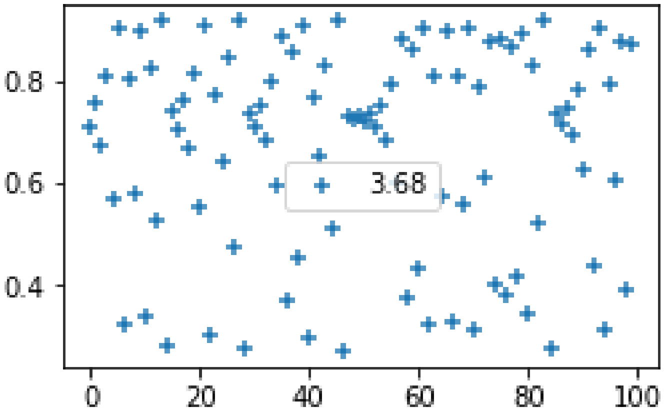
```
[33]: for i in np.arange(3.6, 4, 0.02):
        generate_plot(i)
        plt.legend()
        plt.show()
```

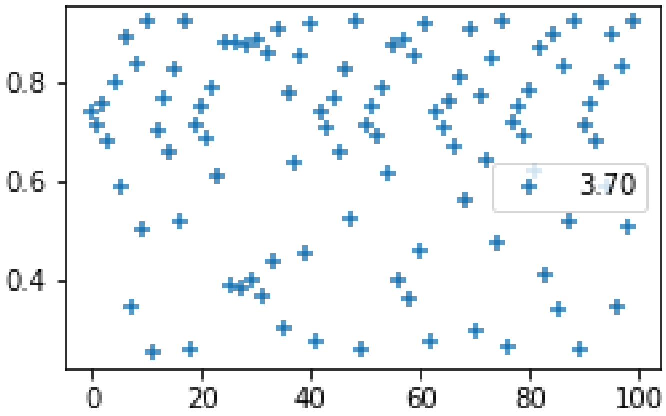




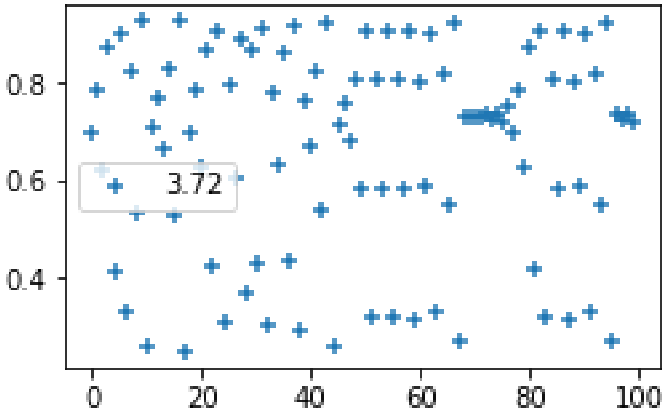

0.8
0.6
0.4
0
20
40
60
80
100
3.64


0.8
0.6
0.4
0
20
40
60
80
100
3.66

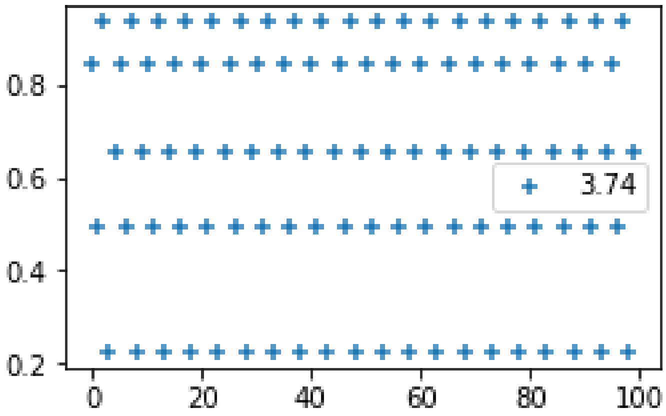

0.8
0.6
0.4
0
20
40
60
80
100
3.68



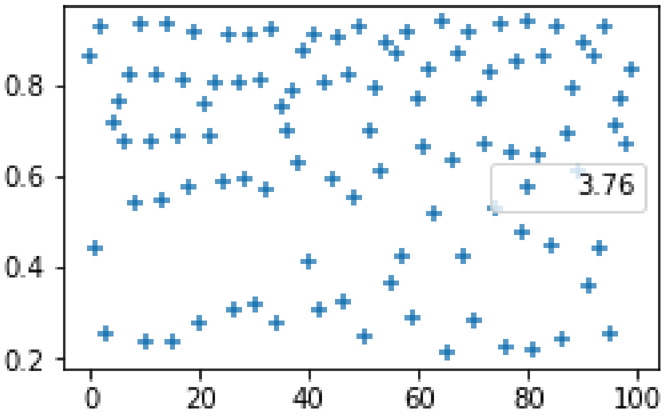
0.8
0.6
0.4
0
20
40
60
80
100
3.70

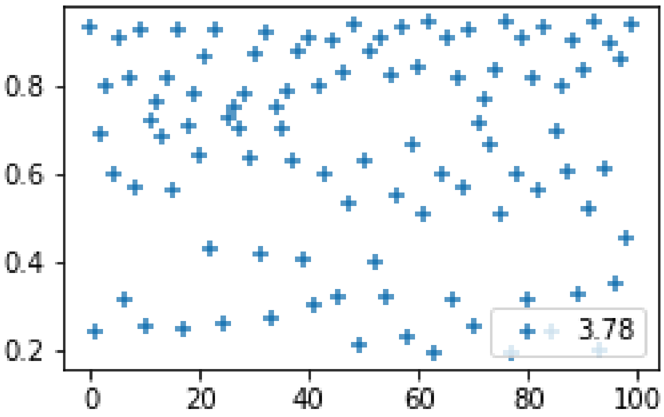


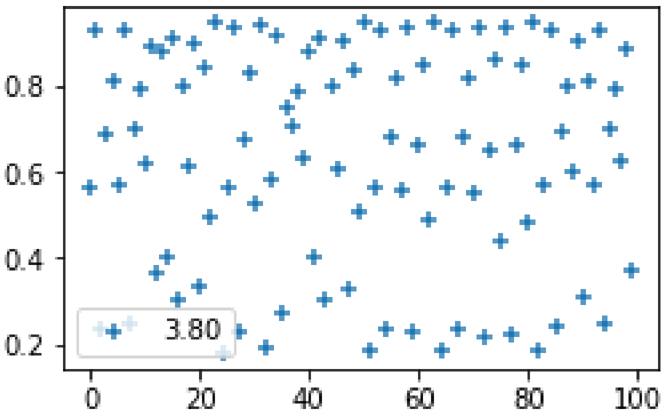
0.8
0.6
0.4
0
20
40
60
80
100
3.72

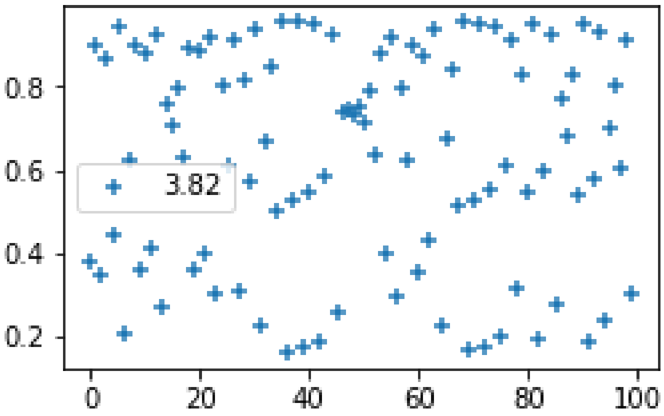


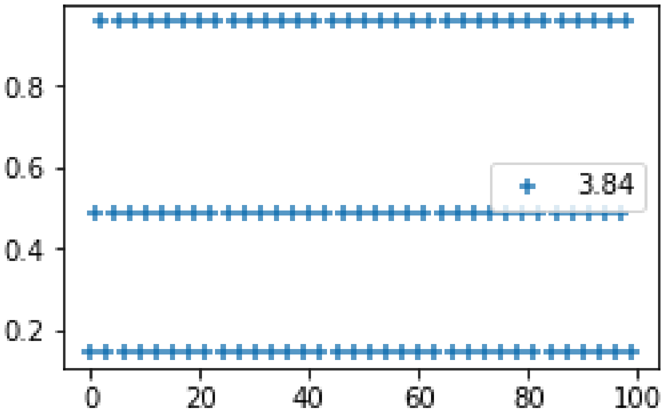
0.8
0.6
0.4
0.2
0
20
40
60
80
100
3.74

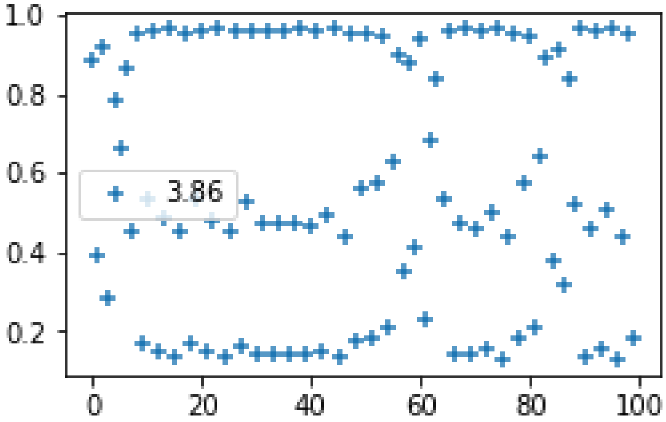

0.8
0.6
0.4
0.2
0
20
40
60
80
100
3.76

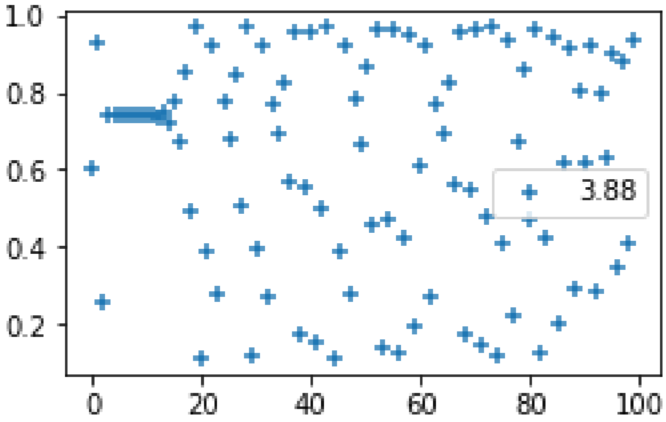

0.8
0.6
0.4
0.2
0
20
40
60
80
100
3.78

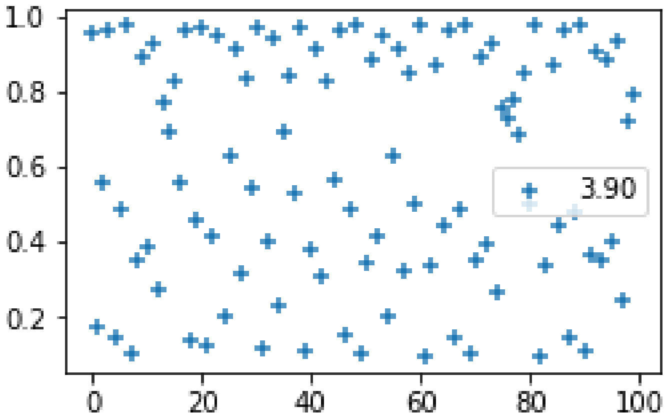

0.8
0.6
0.4
0.2
0
20
40
60
80
100
3.80

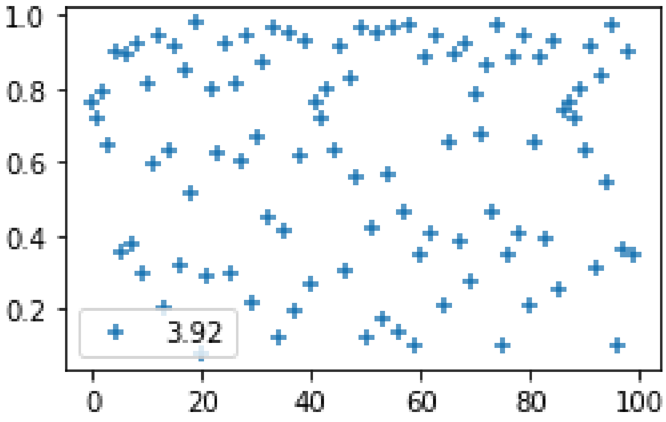

3.82
0.2
0.4
0.6
0.8
0
20
40
60
80
100

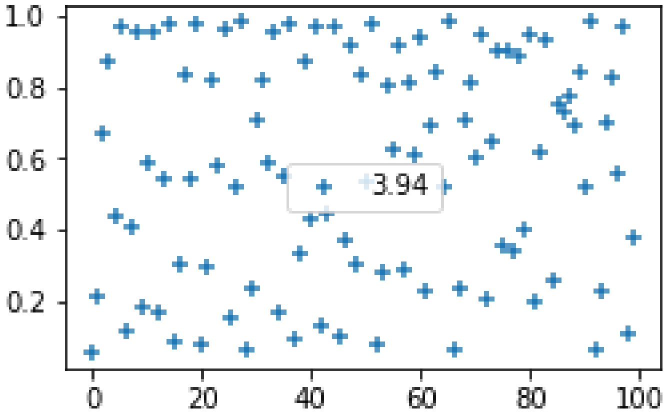

3.84
0.2
0.4
0.6
0.8
0
20
40
60
80
100

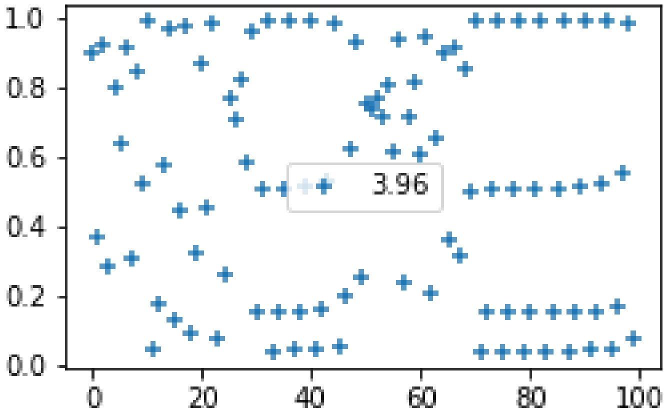

3.86
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

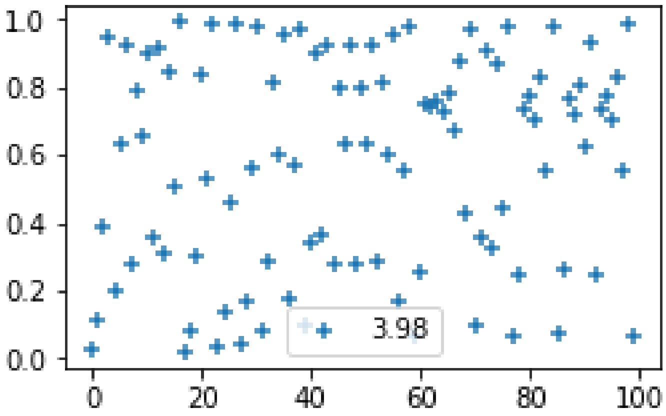

1.0
0.8
0.6
0.4
0.2
0
20
40
60
80
100
+ 3.88


1.0
0.8
0.6
0.4
0.2
0
20
40
60
80
100
+ 3.90


1.0
0.8
0.6
0.4
0.2
0
20
40
60
80
100
+ 3.92



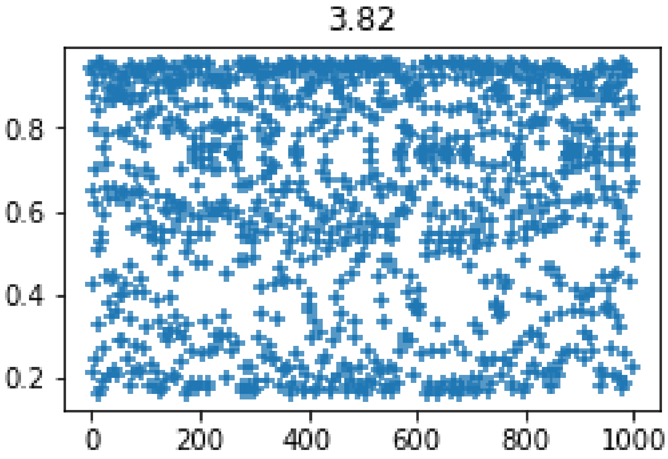




Interesting behaviour seems to occur between various ranges of p. For example, plotting 1000 points, gives concentrated points in some regions but other regions that are less concentrated

[34]:

```
p = 3.82
x = np.random.rand()
for i in range(1000):
  x = p*x*(1-x)

xs = []
for i in range(1000):
  x = p*x*(1-x)
  xs.append(x)
plt.title("{:.2f}".format(p))
plt.scatter(np.arange(len(xs)), xs,marker='+', label='p={:.2f}'.format(p))
plt.show()
```



The equation exibits non linear behaviour at seemlingly random values for p. More analysis will need to be made to determine what these behaviours are and if there are any classifications that can be made for these behavious, and whether any patterns can be observed.