task2

April 18, 2020

```
x = r * x * (1 - x)
phi(x) = r * x * (1 - x)
phi'(x) = r - 2rx
```

для сходимости к корню необходимо, чтобы первое выбранное нами значение лежало в такой окрестности корня, что:
`|phi'(x)| <= 1`

Отсюда следует, что:
`(r - 1) / (2 * r) < x < (r + 1) / (2 * r)`

[2]:
```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.ticker as ticker
import math
import random
```

[3]:
```
def phi(x, r):
    return r * x * (1 - x)

def get_window(r):
    return (r - 1) / (2 * r), (r + 1) / (2 * r)

def is_wave(x, x_0, x_1):
    return x_0 < x < x_1 or x_0 > x > x_1
```

[4]:
```
def simple_iterations(epsilon, r, iter=1000, draw=False, left=0, right=0):
    (left_border, right_border) = get_window(r)
    iterations = 0

    cur_x = random.uniform(left_border, right_border)
    next_x = phi(cur_x, r)

    wave_cur = cur_x
    wave_next = next_x

    x = [next_x]
    wave = False
```

```
    while (abs(cur_x - next_x) > epsilon):
        cur_x = next_x
        next_x = phi(cur_x, r)
        x.append(next_x)
        iterations += 1
        if (iterations > iter):
            return (None, iter, wave)

    for i in range(1, len(x)):
        wave = is_wave(next_x, x[i - 1], x[i])
        if wave:
            break

    if draw:
        print('График зависимости членов итерационной последовательности от␣
↪номера')
        plt.figure(figsize=(15,5))
        plt.plot([float(i) / 10 for i in range(1, len(x) + 1)], x, 'r.', ms=3,␣
↪ls='-', lw=0.4)
        plt.show()
        print(f'Траектория сходимости в плоскости x - phi(x) на отрезке ({left},␣
↪{right})')
        plt.figure(figsize=(15,5))
        plt.plot([phi(i, r) for i in x], x, 'r.', ms=3, ls='-', lw=0.4)
        plt.show()

    return (next_x, iterations, wave)
```

[5]:
```
def check_conv(left, right, iter, eps, check_eps, draw=False):
    roots = [[0, 0], [0, 0], [0, []]]
    for i in range(0, iter):
        cur_r = random.uniform(left, right)
        (answer, iters, wave) = simple_iterations(eps, cur_r, draw=draw,␣
↪left=left, right=right)
        x1 = 0.0
        x2 = 1 - (1 / cur_r)
        if (answer is not None):
            if (abs(answer - x1) < abs(answer - x2) and abs(answer - x1) <␣
↪check_eps):
                roots[0][wave] += 1
            elif (abs(answer - x2) < check_eps):
                roots[1][wave] += 1
            else:
                roots[2][0] += 1
                roots[2][1].append(cur_r)
        else:
            roots[2][0] += 1
```

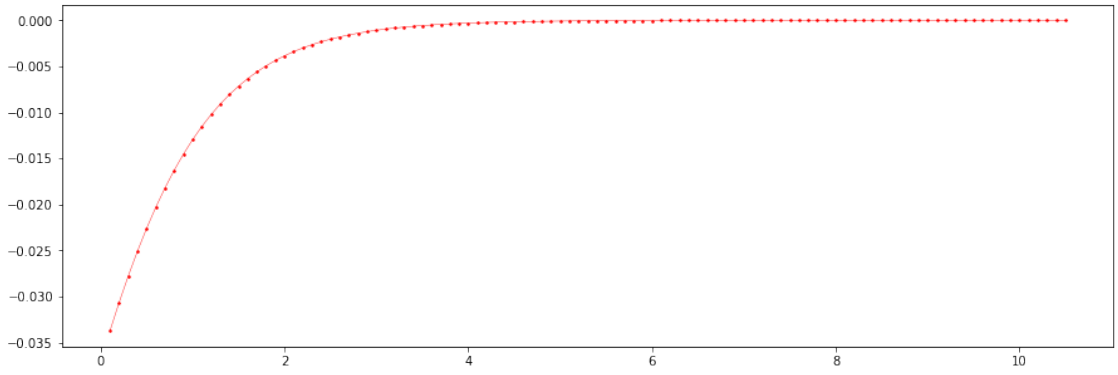
```
            roots[2][1].append(cur_r)

    if not draw:
        print(f"{left} to {right} with eps = {eps} and iter = {iter}:\n")
        print("root 0.0, not wave =         " + str(roots[0][0]))
        print("root 0.0, wave =             " + str(roots[0][1]))
        print("root 1 - (1 / r), not wave = " + str(roots[1][0]))
        print("root 1 - (1 / r), wave =     " + str(roots[1][1]))
        print("not conv =                   " + str(roots[2][0]))
        print("not conv points =            " + str(roots[2][1]))
```

[6]:
```
check_conv(0.01, 0.99, 10**4, 1e-8, 1e-6)
```

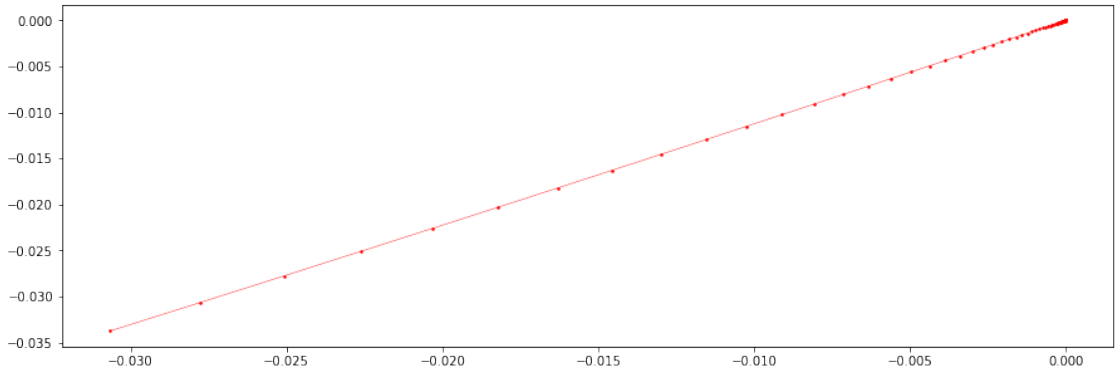
```
0.01 to 0.99 with eps = 1e-08 and iter = 10000:

root 0.0, not wave =         10000
root 0.0, wave =             0
root 1 - (1 / r), not wave = 0
root 1 - (1 / r), wave =     0
not conv =                   0
not conv points =            []
```

[7]:
```
check_conv(0.01, 0.99, 2, 1e-8, 1e-6, draw=True)
```

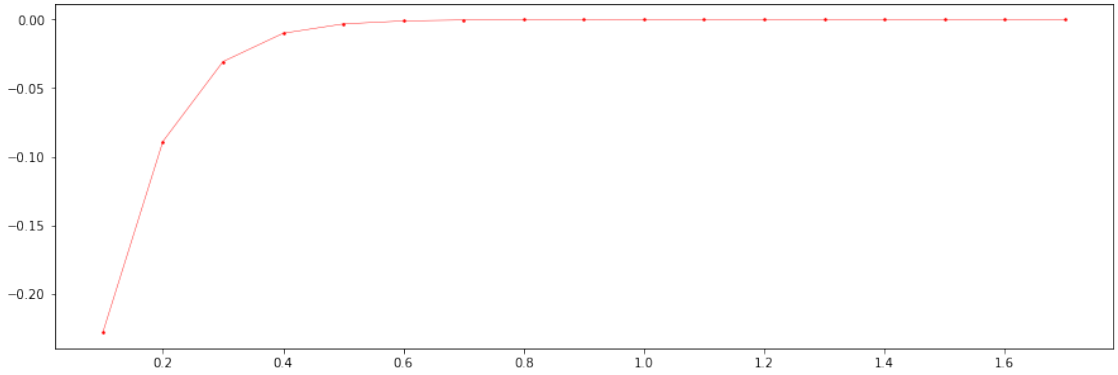
```
График зависимости членов итерационной последовательности от номера
```



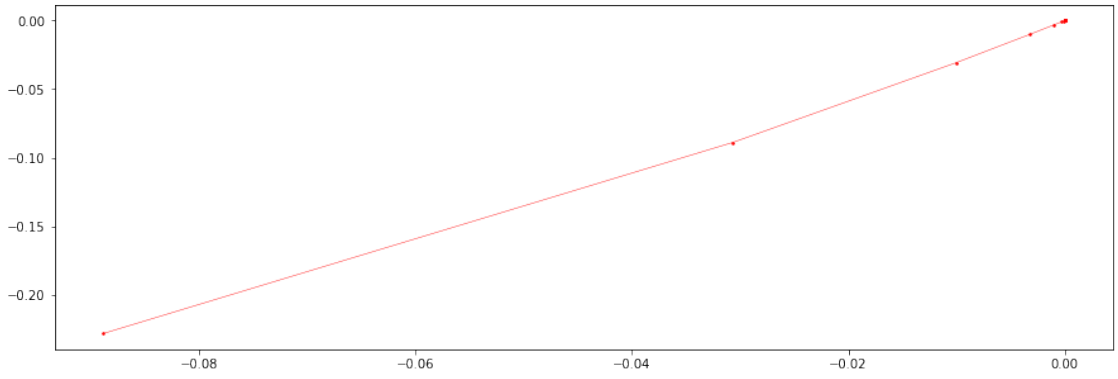

```
Траектория сходимости в плоскости x - phi(x) на отрезке (0.01, 0.99)
```



`График зависимости членов итерационной последовательности от номера`



`Траектория сходимости в плоскости x - phi(x) на отрезке (0.01, 0.99)`



Как видно из приведенных нами вычислений, при выборе `r` из промежутка $(0, 1)$ данная нам функция `phi` сходится монотонно к корню x0 $= 0$

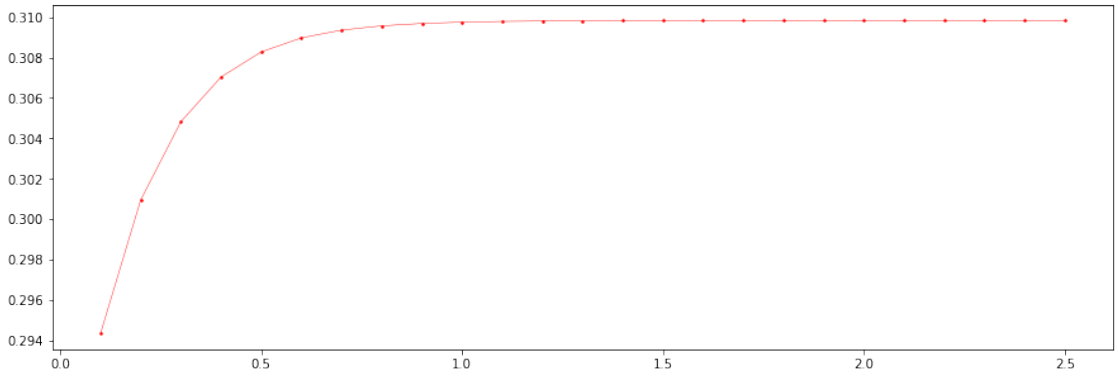
[8]:
```
check_conv(1.01, 2.99, 10**4, 1e-8, 1e-6)
print('\n' * 3)
check_conv(1.01, 1.99, 10**4, 1e-8, 1e-6)
print('\n' * 3)
check_conv(2.01, 2.99, 10**4, 1e-8, 1e-6)
```

```
1.01 to 2.99 with eps = 1e-08 and iter = 10000:

root 0.0, not wave =         0
root 0.0, wave =             0
root 1 - (1 / r), not wave = 4952
root 1 - (1 / r), wave =     5024
not conv =                   24
not conv points =            [2.9892349550382793, 2.9845431350773706,
2.986365700667192, 2.985286764745876, 2.989432606371902, 2.987416387403414,
2.9877140806574776, 2.989492832782317, 2.986090572588725, 2.9865694559909226,
2.9856805322178643, 2.9885698117480555, 2.9872050991342274, 2.9853009295306943,
2.9885920705689135, 2.9852830597167834, 2.9899355767279827, 2.9867268642330727,
2.98629996898916, 2.9862933582892666, 2.9890135412729224, 2.9854290538687955,
2.987130080155495, 2.98477702869347]




1.01 to 1.99 with eps = 1e-08 and iter = 10000:

root 0.0, not wave =         0
root 0.0, wave =             0
root 1 - (1 / r), not wave = 10000
root 1 - (1 / r), wave =     0
not conv =                   0
not conv points =            []




2.01 to 2.99 with eps = 1e-08 and iter = 10000:

root 0.0, not wave =         0
root 0.0, wave =             0
root 1 - (1 / r), not wave = 0
root 1 - (1 / r), wave =     9940
not conv =                   60
not conv points =            [2.9862714249309716, 2.989222617444504,
2.9856741404675207, 2.9889537774427635, 2.986040990395459, 2.9854267802815206,
2.9888789974099357, 2.9863523897144666, 2.985269168437876, 2.9898253800813572,
2.988245187465537, 2.9884576373842178, 2.985015949866633, 2.989826752108873,
```

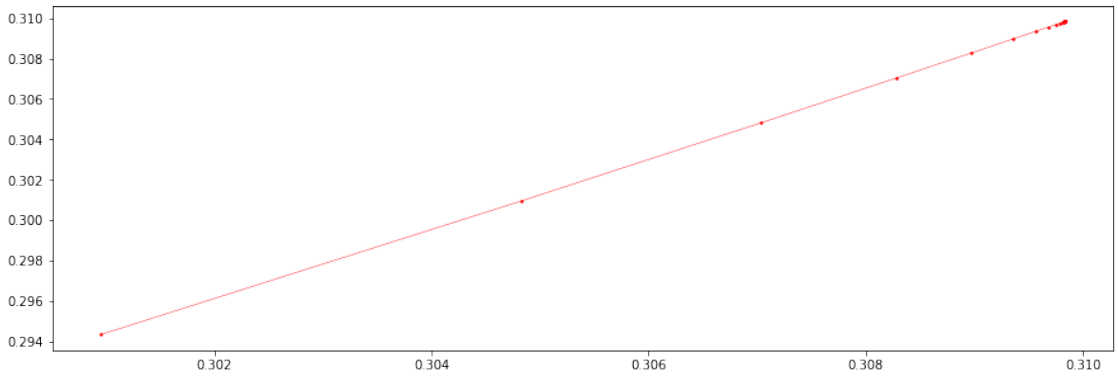
```
2.984763646690693, 2.9856906784269968, 2.9884289800966877, 2.9890485077534574,
2.9888389452249857, 2.9894803832509074, 2.9873829124364857, 2.9885820227448283,
2.987133512697204, 2.9895486979692496, 2.9871358088499687, 2.989256236992371,
2.9873703019751794, 2.986544481192915, 2.987818394433347, 2.9848090668799037,
2.9863769736705597, 2.987431146996556, 2.984946683874562, 2.988958840751083,
2.987333899338654, 2.9857916851518898, 2.984660199187437, 2.987273893166856,
2.98848191798856, 2.984794017449434, 2.9880088297468776, 2.9856325910563224,
2.9870102006634665, 2.9885419116294782, 2.9869363452561344, 2.9845174606050384,
2.9878425959165797, 2.986768078574401, 2.984588352969049, 2.9884038183181465,
2.989706153415825, 2.9854506248850115, 2.98930933630881, 2.985032814037498,
2.9853620044646902, 2.985831763102135, 2.9862976658190736, 2.989656010915273,
2.985133625994884, 2.989571614660689]
```

[9]:
```
check_conv(1.01, 1.99, 2, 1e-8, 1e-6, draw=True)
```

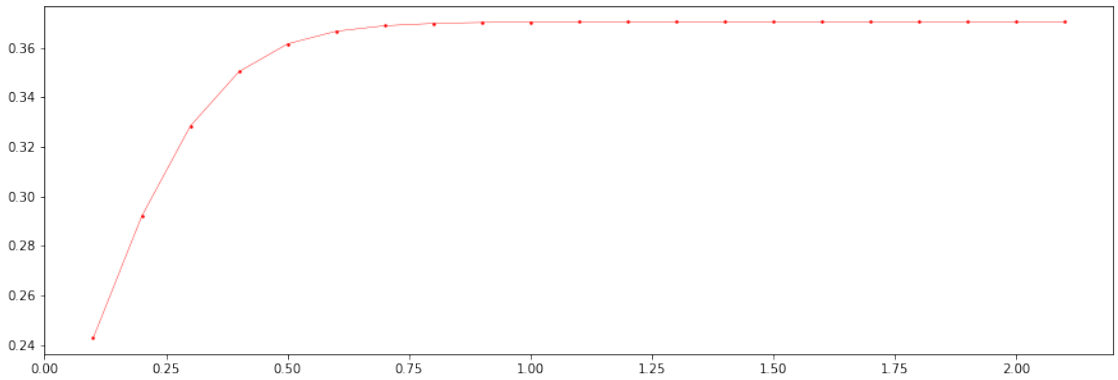
```
График зависимости членов итерационной последовательности от номера
```



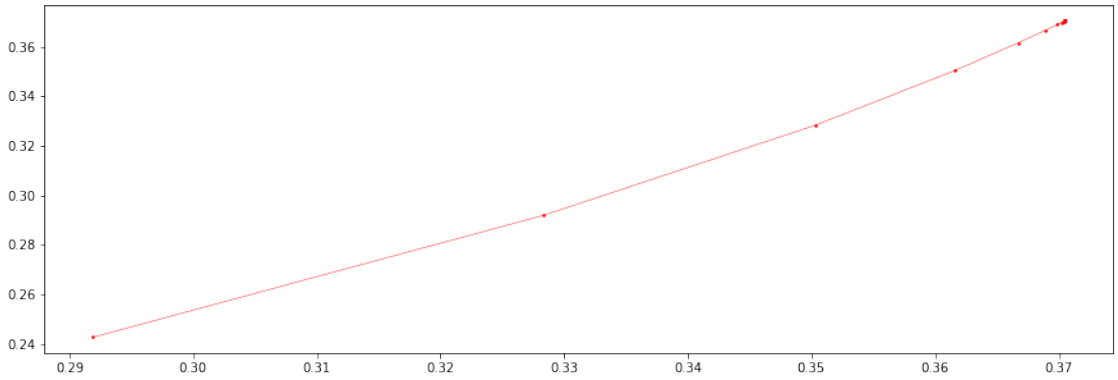

```
Траектория сходимости в плоскости x - phi(x) на отрезке (1.01, 1.99)
```




```
График зависимости членов итерационной последовательности от номера
```

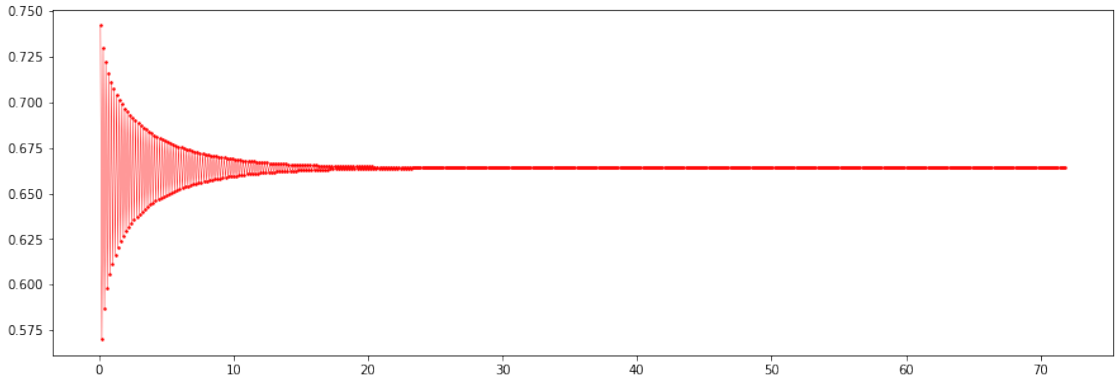


Траектория сходимости в плоскости x - phi(x) на отрезке (1.01, 1.99)



[10]:
```
check_conv(2.01, 2.99, 2, 1e-8, 1e-6, draw=True)
```

График зависимости членов итерационной последовательности от номера



Траектория сходимости в плоскости x - phi(x) на отрезке (2.01, 2.99)

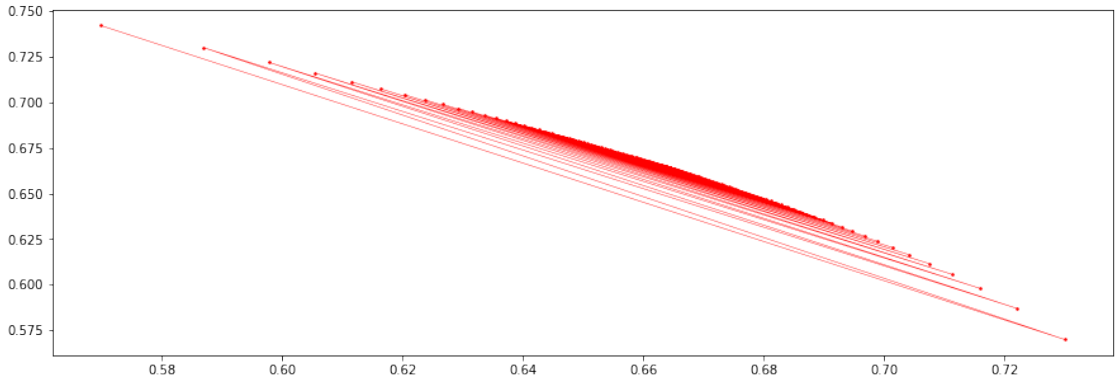
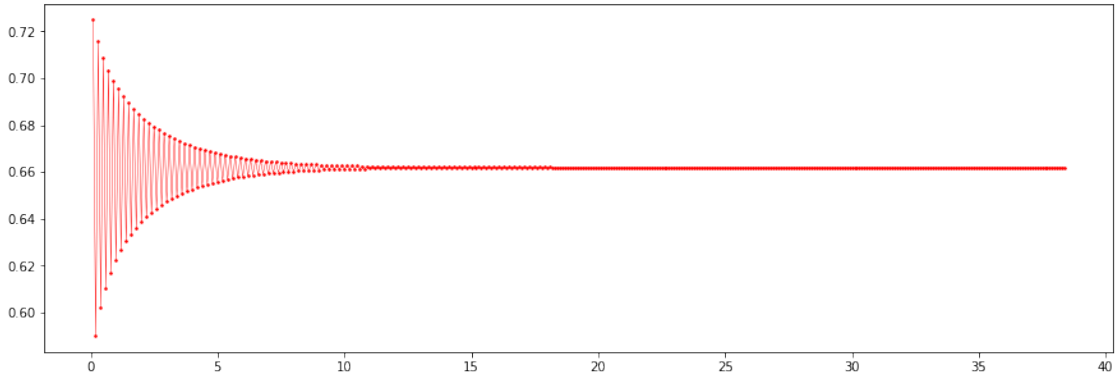
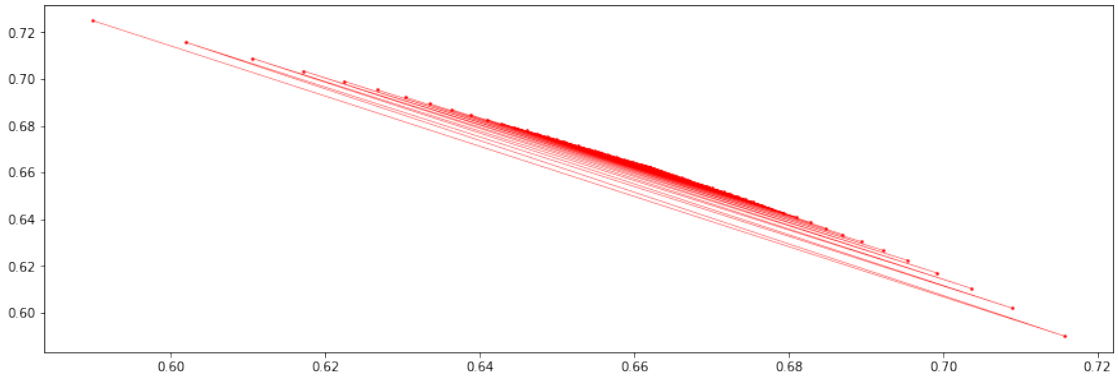


График зависимости членов итерационной последовательности от номера



Траектория сходимости в плоскости x - phi(x) на отрезке (2.01, 2.99)



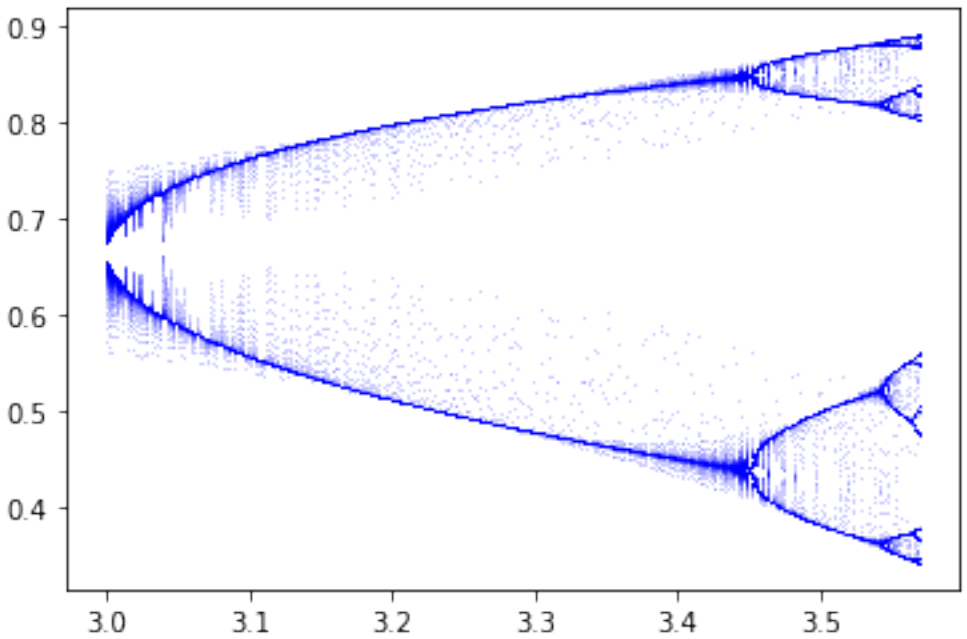
Как видно из приведенных нами вычислений, при выборе **r** из промежутка $(1, 3)$ данная нам

функция `phi` сходится к корню x1 = 1 - (1 / r)

При этом если выбирать `r` из промежутка (1, 2), то функция `phi` сходится к корню x1 монотонно, а при выборе `r` из промежутка (2, 3) — колебательно

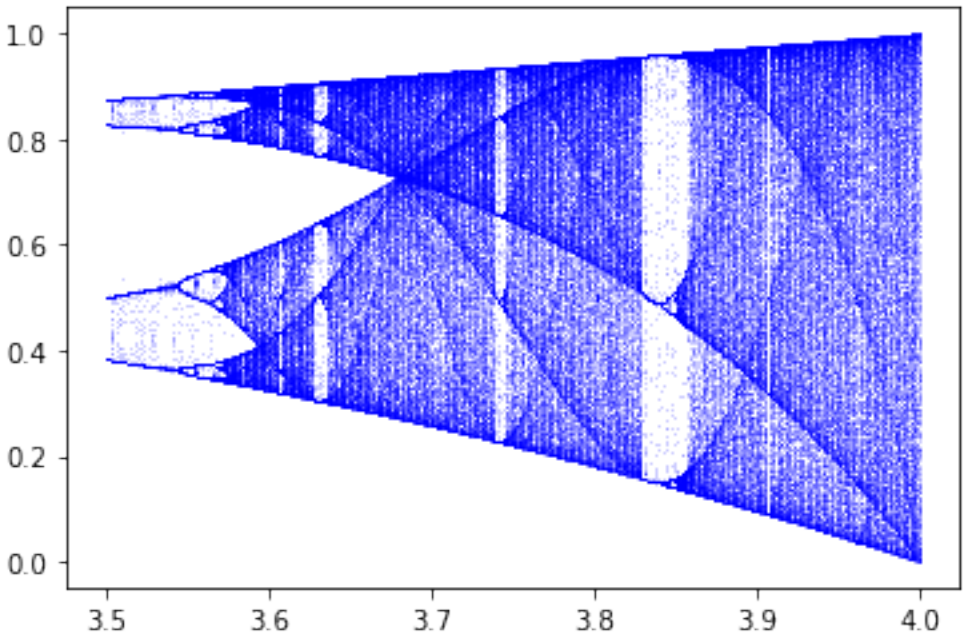
[88]:
```
def show_graphic(r, rang, eps):
    xVals = []
    yVals = []
    yVals1 = []

    for j in range(rang):
        (lBorder, rBorder) = get_window(r)
        x = random.uniform(lBorder, rBorder)
        for i in range(500):
            x = phi(x, r)
            xVals.append(x)
            yVals.append(r)
        r += eps

    plt.plot(yVals, xVals, 'b+', ms=0.1)
    plt.show()
```

[89]:
```
def show_bifurcation():
    show_graphic(3.0, 570, 0.001)

show_bifurcation()
```



В теории график должен распасться в диапазонах 3-3.35..., 3.35...-3.52..., 3.52...-3.56... и т.д. Мы можем заметить, что на нашем графике это выполняется. Мы получили каскад бифуркаций удвоения периода.

[13]:
```
def show_chaos():
    show_graphic(3.50, 500, 0.001)

show_chaos()
```



По теории, между числами R(inf) и 4, поведение последовательности должно представлять из себя детерменированный хаос. График ведет себя в соответствии с теорией. На нем присутствуют зоны таких R, при которых наблюдаются сгущения и разрежения итерационной последовательности. По теории в окрестности R = 4 должен наблюдаться белый шум. Наш график подходит под теорию и в этом случае.

[118]:
```
def show_iteration1(r, rang, eps):
    xVals = []
    yVals1 = []

    (lBorder, rBorder) = get_window(r)
    x = random.uniform(lBorder, rBorder)
    for i in range(rang):
        x = phi(x, r)
        xVals.append(x)
        yVals1.append(i)
```

```
    plt.figure(figsize=(15,5))
    plt.plot(yVals1, xVals, 'g.', ms=6, ls='-', lw = 0.4)
    plt.show()
```

[119]:
```
print("График зависимости членов итерационной последовательности от номера␣
 ↪итерации в области каскада бифуркаций"
      " удвоения периода r = 3.1.")
show_iteration1(3.1, 100, 0.01)
print("График зависимости членов итерационной последовательности от номера␣
 ↪итерации в области каскада бифуркаций"
      " удвоения периода r = 3.53.")
show_iteration1(3.53, 100, 0.01)
print("График зависимости членов итерационной последовательности от номера␣
 ↪итерации в области хаоса r = 3.7.")
show_iteration1(3.7, 100, 0.01)
```

График зависимости членов итерационной последовательности от номера итерации в области каскада бифуркаций удвоения периода r = 3.1.

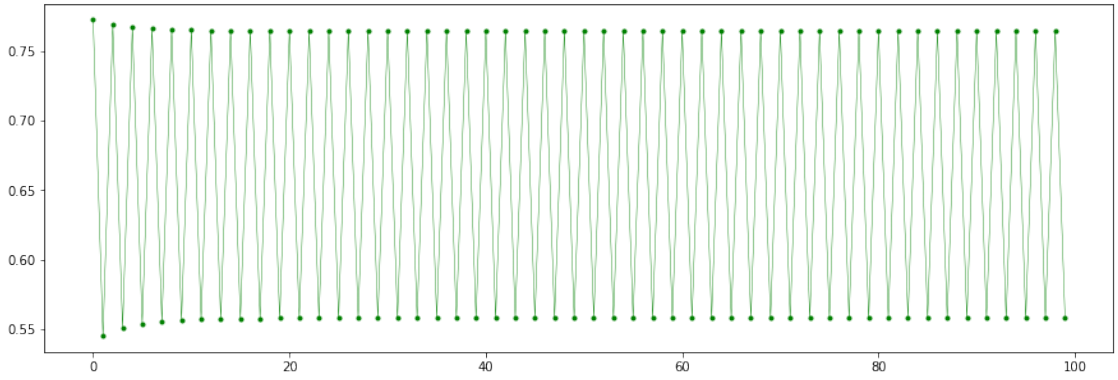


График зависимости членов итерационной последовательности от номера итерации в области каскада бифуркаций удвоения периода r = 3.53.

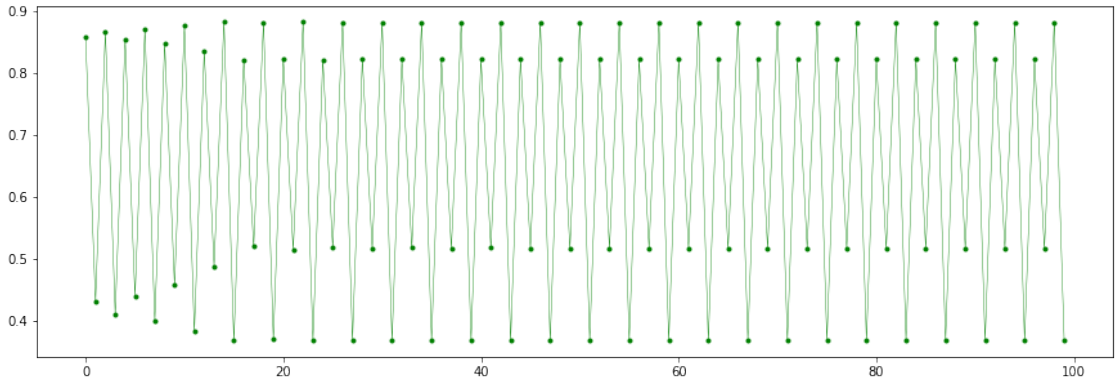
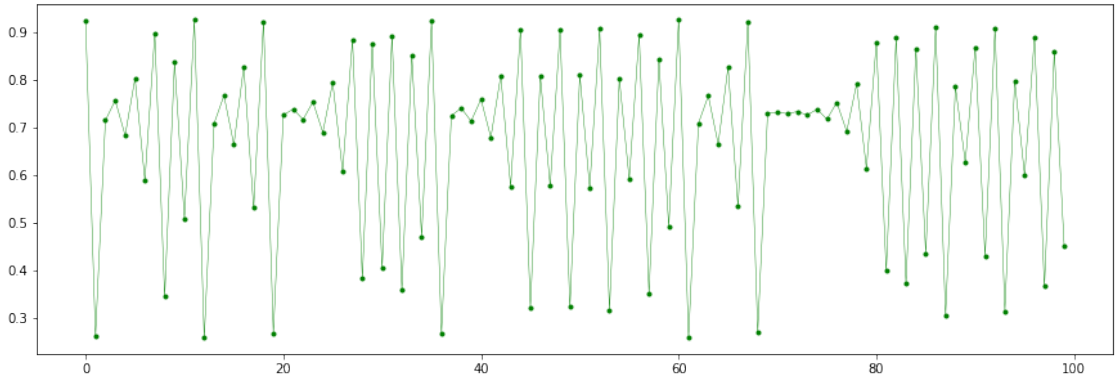


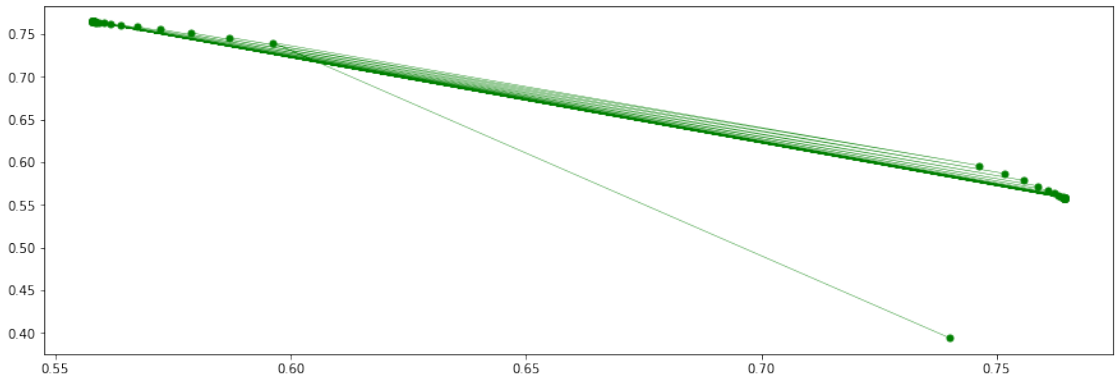
График зависимости членов итерационной последовательности от номера итерации в области хаоса r = 3.7.



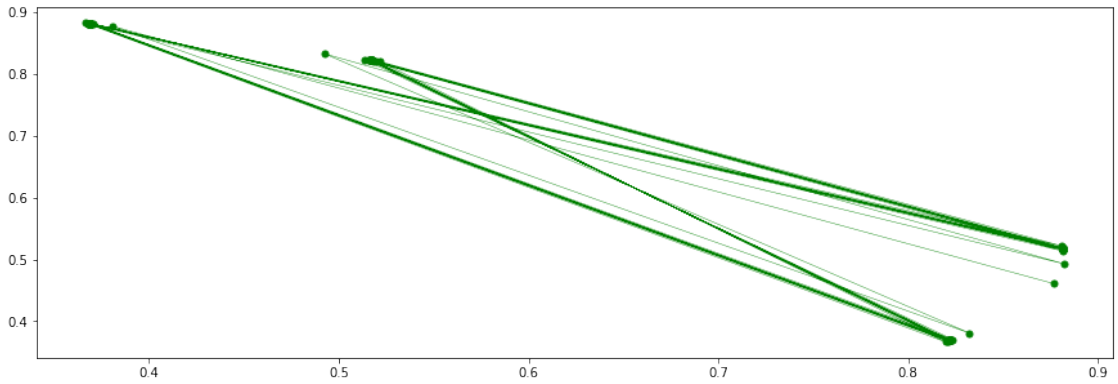
[122]:
```
def show_iteration2(r, rang, eps):
    xVals = []
    yVals1 = []

    (lBorder, rBorder) = get_window(r)
    x = random.uniform(lBorder, rBorder)
    for i in range(rang):
        xVals.append(x)
        x = phi(x, r)
        yVals1.append(x)

    plt.figure(figsize=(15,5))
    plt.plot(yVals1, xVals, 'g.', ms=10, ls='-', lw = 0.4)
    plt.show()
```

[123]:
```
print("Траектория сходимости в плоскости x phi(x). r = 3.1.")
show_iteration2(3.1, 200, 0.01)
print("Траектория сходимости в плоскости x phi(x). r = 3.53.")
show_iteration2(3.53,200, 0.01)
print("Траектория сходимости в плоскости x phi(x). r = 3.7.")
show_iteration2(3.7, 100, 0.01)
```

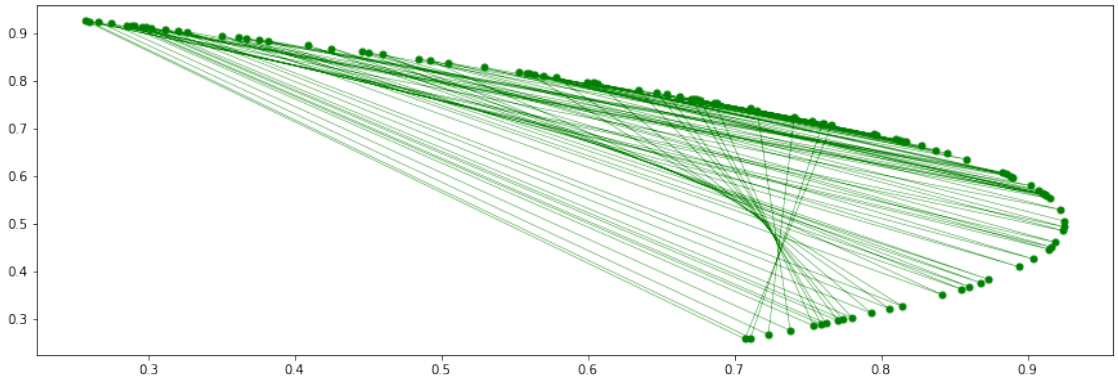
Траектория сходимости в плоскости x phi(x). r = 3.1.



Траектория сходимости в плоскости x phi(x). r = 3.53.



Траектория сходимости в плоскости x phi(x). r = 3.7.



[ ]:

[ ]: