

October 30, 2021

```
[1]: import scipy.integrate as integr
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import argrelextrema
import pylab

colors = ["#00FFFF",          # Azure
          "#0000FF",          # Blue
          "#FF0000",          # Red
          "#00FF00",          # Green
          "#D2691E",          #
          "#AAA662",
          "#9A0EEA",
          "#808000",
          "#FFA500",
          "#9ACD32",
          "#ED0DD9",
          "#FC5A50",
          "#929591",
          "#029386",
          "#C79FEF",
          "#FAC205",
          "#6E750E",
          "#06C2AC",
          "#CD5C5C",          # IndianRed
          "#A52A2A",          # Brown
          "#7B68EE",          # MediumState_blue
          "#4682B4",          # SteelBlue
          "#800000"          # Maroon
        ]
colors = 10*colors

mycolors = [
    '#000000',
    '#696969',
    '#A9A9A9',
    '#C0C0C0',
```

```

        '#D3D3D3',
        '#DCDCDC',
    ]

    def mesh(x, y):
        general = []
        for i in range(len(x)):
            for j in range(len(y)):
                local = []
                local.append(x[i])
                local.append(y[j])
                general.append(local)
        return general

    stat_dpi = 90
    rect_pics = (7, 7)
    big_pics = (10, 10)
    dynamics_pics = (14, 7)

```

```

[2]: def plotFP(y1, y2, centers = None, starts = None, color = "b"):
    fig = plt.figure(facecolor="white", figsize = rect_pics, dpi=stat_dpi)
    plt.plot(y1, y2, c = color)
    if centers is not None:
        for i in centers:
            plt.scatter(i[0], i[1])
    if starts is not None:
        for i in starts:
            plt.scatter(i[0], i[1])
    plt.grid(True)
    plt.show()

```

```

[3]: def logistic(t, r, x_0, t_0):
    result = 1/(1 + (1/x_0 - 1) * np.exp(-r*(t-t_0)))
    return result

```

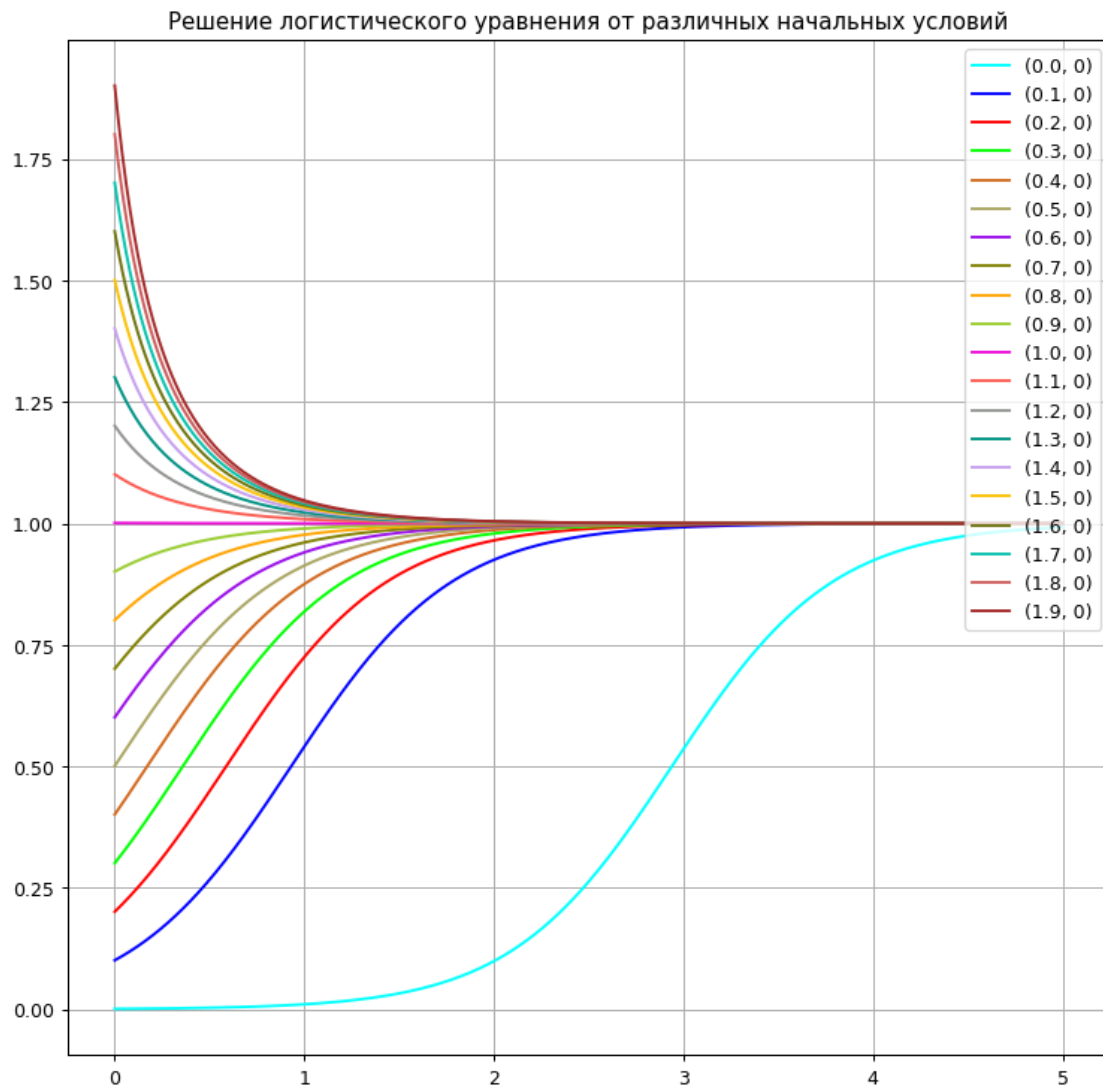
```

[4]: time = np.linspace(0, 5, 500)
    #print(y)
    #print(time)

    fig = plt.figure(facecolor="white", figsize = big_pics, dpi=stat_dpi)
    step = 0.1
    for i in np.arange(0.001, 2, step):
        plt.plot(time, logistic(time, 2.35, i, 0), c = colors[int(i/step)%30],
        ↪label = f'({round(i, 2)}, 0)')
    plt.title(" ")
    plt.legend(loc="upper right")
    plt.grid(True)

```

```
plt.show()
```

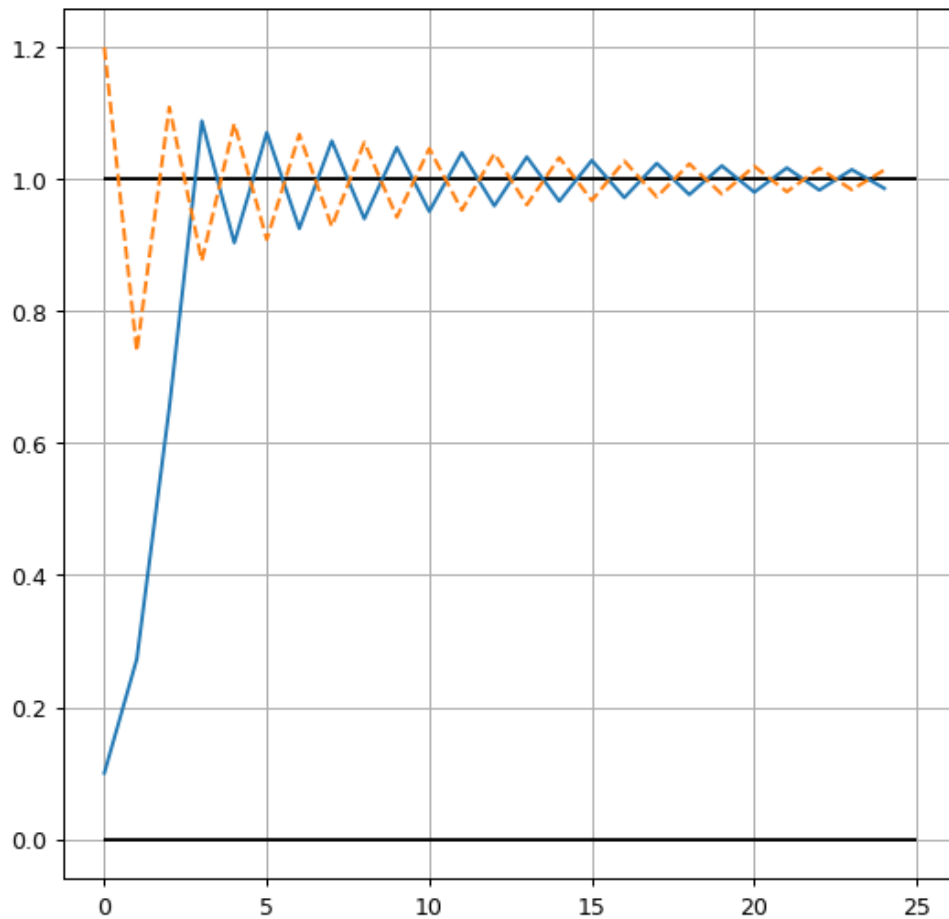


```
[5]: def discr_sol(x_0, r, times=0):  
    X = [x_0]  
    for i in range(times-1):  
        X.append((1 + r)*X[-1] - r*X[-1]**2)  
    return X
```

```
[6]: time = np.arange(0, 25)  
fig = plt.figure(facecolor="white", figsize = rect_pics, dpi=stat_dpi)  
r=1.92  
  
plt.hlines(1, 0, len(time), color='black')
```

```
plt.hlines(0, 0, len(time), color='black')
plt.plot(discr_sol(0.1, r, len(time)))
plt.plot(discr_sol(1.2, r, len(time)), linestyle='dashed')
plt.title(r"$f(x) = (1+r)x - rx^2$   r = "+str(r))
plt.grid(True)
plt.show()
```

Диаграмма последования для отображения $f(x) = (1+r)x - rx^2$ при $r = 1.92$

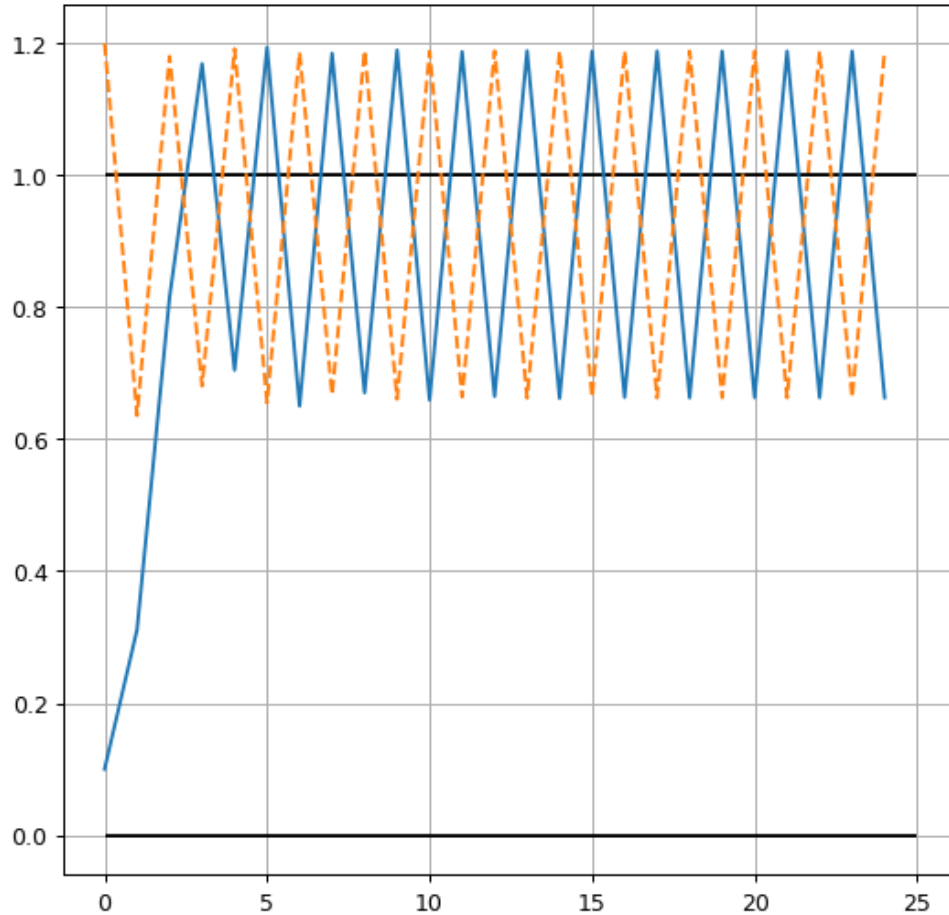


```
[7]: time = np.arange(0, 25)
fig = plt.figure(facecolor="white", figsize = rect_pics, dpi=stat_dpi)
r=2.35

plt.hlines(1, 0, len(time), color='black')
plt.hlines(0, 0, len(time), color='black')
plt.plot(discr_sol(0.1, r, len(time)))
plt.plot(discr_sol(1.2, r, len(time)), linestyle='dashed')
plt.title(r"$f(x) = (1+r)x - rx^2$   r = "+str(r))
```

```
plt.grid(True)
plt.show()
```

Диаграмма последования для отображения $f(x) = (1 + r)x - rx^2$ при $r = 2.35$



```
[8]: time = np.arange(0, 50)
fig = plt.figure(facecolor="white", figsize = big_pics, dpi=stat_dpi)

r = 2.3

plt.hlines(1, 0, len(time), color='black')
plt.hlines(0, 0, len(time), color='black')

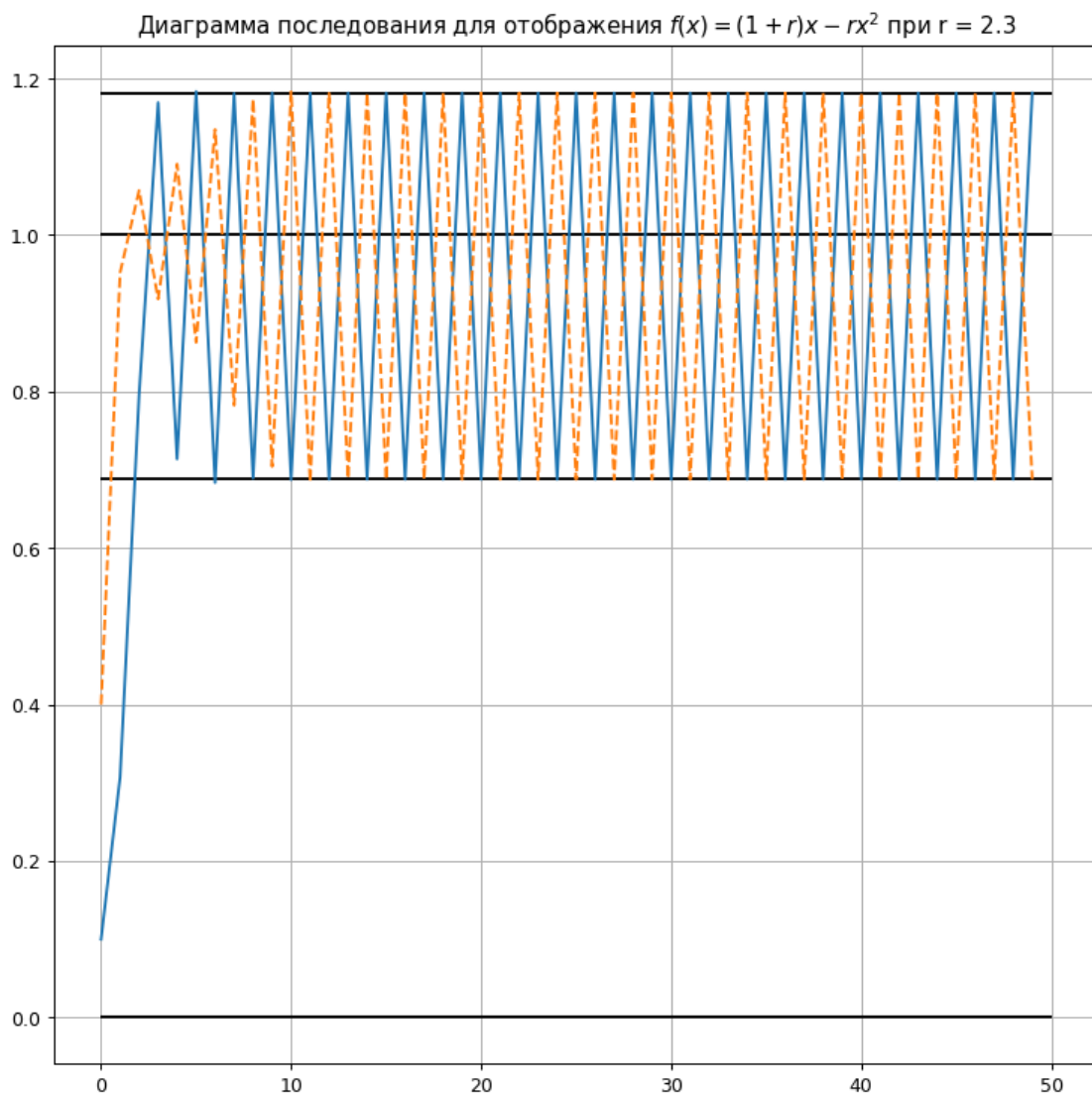
xp3 = (2*r + r**2 - r*(-4 + r**2)**0.5)/(2*r**2)
xp4 = (2*r + r**2 + r*(-4 + r**2)**0.5)/(2*r**2)

plt.hlines(xp3, 0, len(time), color='black')
plt.hlines(xp4, 0, len(time), color='black')
```

```

#plt.plot(discr_sol(0.1, 2.35, len(time)))
plt.plot(discr_sol(0.1, r, len(time)))
plt.plot(discr_sol(0.4, r, len(time)), linestyle='dashed')
plt.title(r"$f(x) = (1+r)x - rx^2$   r = "+str(r))
plt.grid(True)
plt.show()

```



```

[9]: time = np.arange(0, 50)
fig = plt.figure(facecolor="white", figsize = rect_pics, dpi=stat_dpi)

r = -2.3

```

```

plt.hlines(1, 0, len(time), color='black')
plt.hlines(0, 0, len(time), color='black')

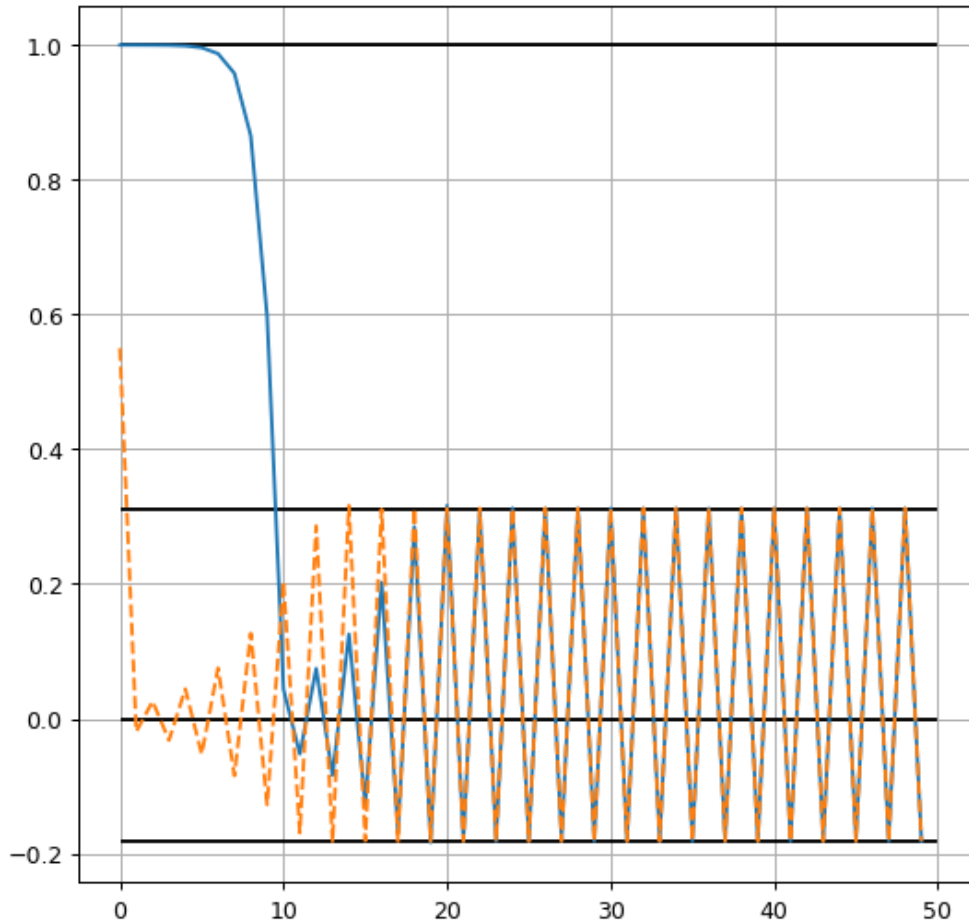
xp3 = (2*r + r**2 - r*(-4 + r**2)**0.5)/(2*r**2)
xp4 = (2*r + r**2 + r*(-4 + r**2)**0.5)/(2*r**2)

plt.hlines(xp3, 0, len(time), color='black')
plt.hlines(xp4, 0, len(time), color='black')

plt.plot(discr_sol(0.99999, r, len(time)))
plt.plot(discr_sol(0.55, r, len(time)), linestyle='dashed')
plt.title(r"$f(x) = (1+r)x - rx^2$   r = "+str(r))
plt.grid(True)
plt.show()

```

Диаграмма последования для отображения $f(x) = (1 + r)x - rx^2$ при $r = -2.3$



```

[10]: time = np.arange(0, 100)
fig = plt.figure(facecolor="white", figsize = rect_pics, dpi=stat_dpi)

r = -2

plt.hlines(1, 0, len(time), color='black')
plt.hlines(0, 0, len(time), color='black')

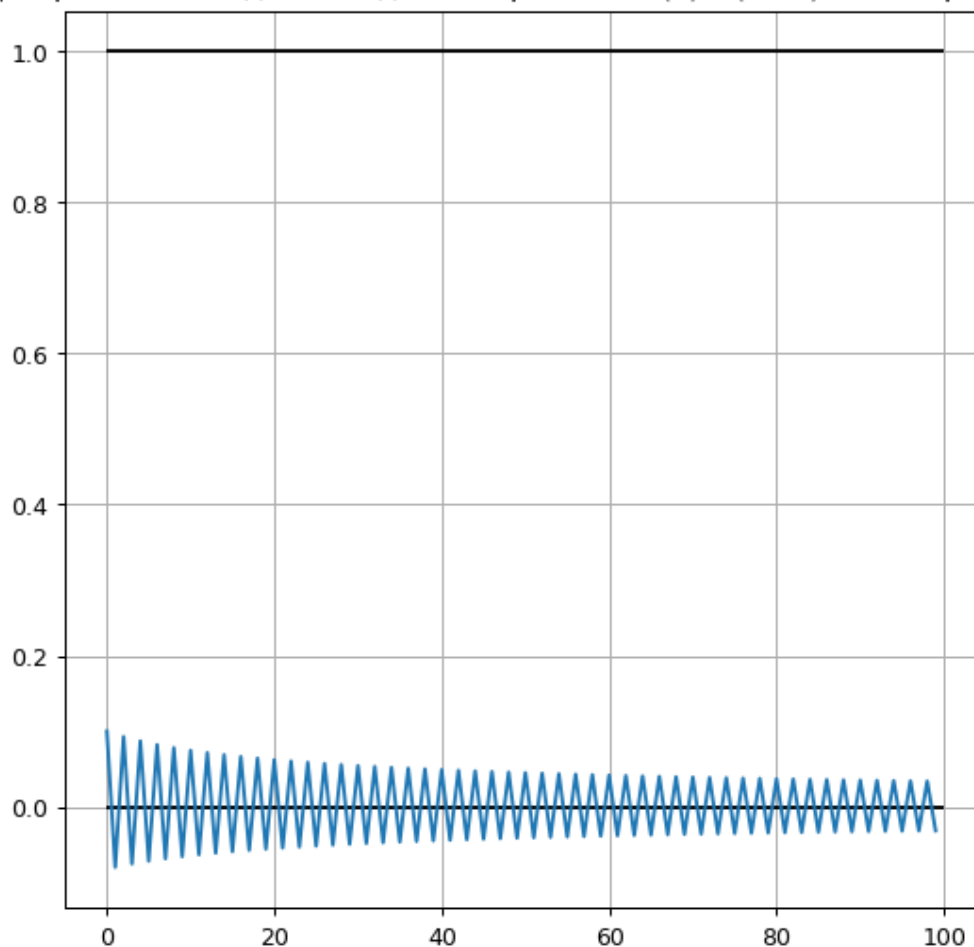
xp3 = (2*r + r**2 - r*(-4 + r**2)**0.5)/(2*r**2)
xp4 = (2*r + r**2 + r*(-4 + r**2)**0.5)/(2*r**2)

#plt.hlines(xp3, 0, len(time), color='black')
#plt.hlines(xp4, 0, len(time), color='black')

#plt.plot(discr_sol(0.1, 2.35, len(time)))
plt.plot(discr_sol(0.1, r, len(time)))
plt.title(r"                                 $f(x) = (1+r)x - rx^2$       r = "+str(r))
plt.grid(True)
plt.show()

```


Диаграмма последования для отображения $f(x) = (1 + r)x - rx^2$ при $r = -2$



```
[11]: def KenLam(times, x_0, r, with_lines = True, with_arrows = True, with_3cr =
↪False, with_4cr=False, constant_size=False, arrow_c=["green"]):
    f = lambda x : (1 + r)*x - r*x**2
    fig = plt.figure(facecolor="white", figsize = big_pics, dpi=stat_dpi)
    dtime = np.arange(x_0, times)
    X = [x_0]
    Y = [x_0]
    plt.scatter(X, Y)

    for i in dtime:
        y = f(X[-1])
        X.append(X[-1])
        Y.append(y)
        X.append(y)
        Y.append(Y[-1])
```

```

scale = (max(X)-min(X))
for i in range(1, len(X)):
    if with_arrows:
        plt.arrow(X[i-1], Y[i-1], X[i]-X[i-1],
                    Y[i]-Y[i-1], head_width=0.01*scale, head_length=0.02*scale,
                    color=arrow_c[i%(len(arrow_c))], width=0.0005*scale,
→alpha=0.3, length_includes_head=True)

time = np.linspace(min(X), max(X))
if constant_size:
    time = np.linspace(0, (1+r)/r)
plt.plot(time, f(time))
if with_arrows:
    plt.plot(time, time)
if with_3cr:
    plt.plot(time, f(f(time)))
if with_4cr:
    plt.plot(time, f(f(f(time))), color='yellow')

if with_lines:
    plt.plot([X[i] for i in range(len(X)) if i%2==0][: -1], [X[i] for i in
→range(len(X)) if i%2==0][1:], color="red")
    #print([X[i] for i in range(len(X)) if i%2==0])
    plt.title(r"          -          $f(x) = (1+r)x - rx^2$"+f"      {x_0=},
→{r=}")
    plt.grid(True)
    plt.show()

```

```

[12]: KenLam(500, 0.001, -2.3, with_arrows=True, with_3cr=False, with_4cr=True,
→arrow_c=['black'], with_lines=False)
KenLam(500, 0.1, 1.9, with_arrows=True, with_3cr=False, with_4cr=True,
→arrow_c=['black'], with_lines=False)

r = 2
x=(2*r + r**2 - r*(-4 + r**2)**0.5)/(2*r**2)
KenLam(100, x, r, with_arrows=True, with_3cr=True, constant_size=True,
→arrow_c=['black'])
x = (2*r + r**2 + r*(-4 + r**2)**0.5)/(2*r**2)

```

```

KenLam(100, x, r, with_arrows=True, with_3cr=True, constant_size=True,
↪arrow_c=['black'])

r = 2.05
x=(2*r + r**2 - r*(-4 + r**2)**0.5)/(2*r**2)
KenLam(100, x, r, with_arrows=True, with_3cr=True, constant_size=True,
↪arrow_c=['black'])
x = (2*r + r**2 + r*(-4 + r**2)**0.5)/(2*r**2)
KenLam(100, x, r, with_arrows=True, with_3cr=True, constant_size=True,
↪arrow_c=['black'])

KenLam(100, 0.1, 2.1, with_arrows=True, with_3cr=True, constant_size=True,
↪arrow_c=['black'], with_lines=False)
KenLam(100, 0.1, 2.3, with_arrows=True, with_3cr=True, constant_size=True,
↪arrow_c=['black'], with_lines=False)
KenLam(100, 0.1, 2.45, with_arrows=True, with_3cr=True, constant_size=True,
↪arrow_c=['black'], with_lines=False)
KenLam(500, 0.1, 2.5, with_arrows=True, with_3cr=False, with_4cr=True,
↪arrow_c=['black'], with_lines=False)
KenLam(500, 0.1, 2.55, with_arrows=True, with_3cr=False, with_4cr=True,
↪arrow_c=['black'], with_lines=False)
KenLam(500, 0.1, 2.57, with_arrows=True, with_3cr=False, with_4cr=True,
↪arrow_c=['black'], with_lines=False)

```

Диаграмма Кёнигса-Ламерея для отображения $f(x) = (1+r)x - rx^2$ при $x_0=0.001$, $r=-2.3$

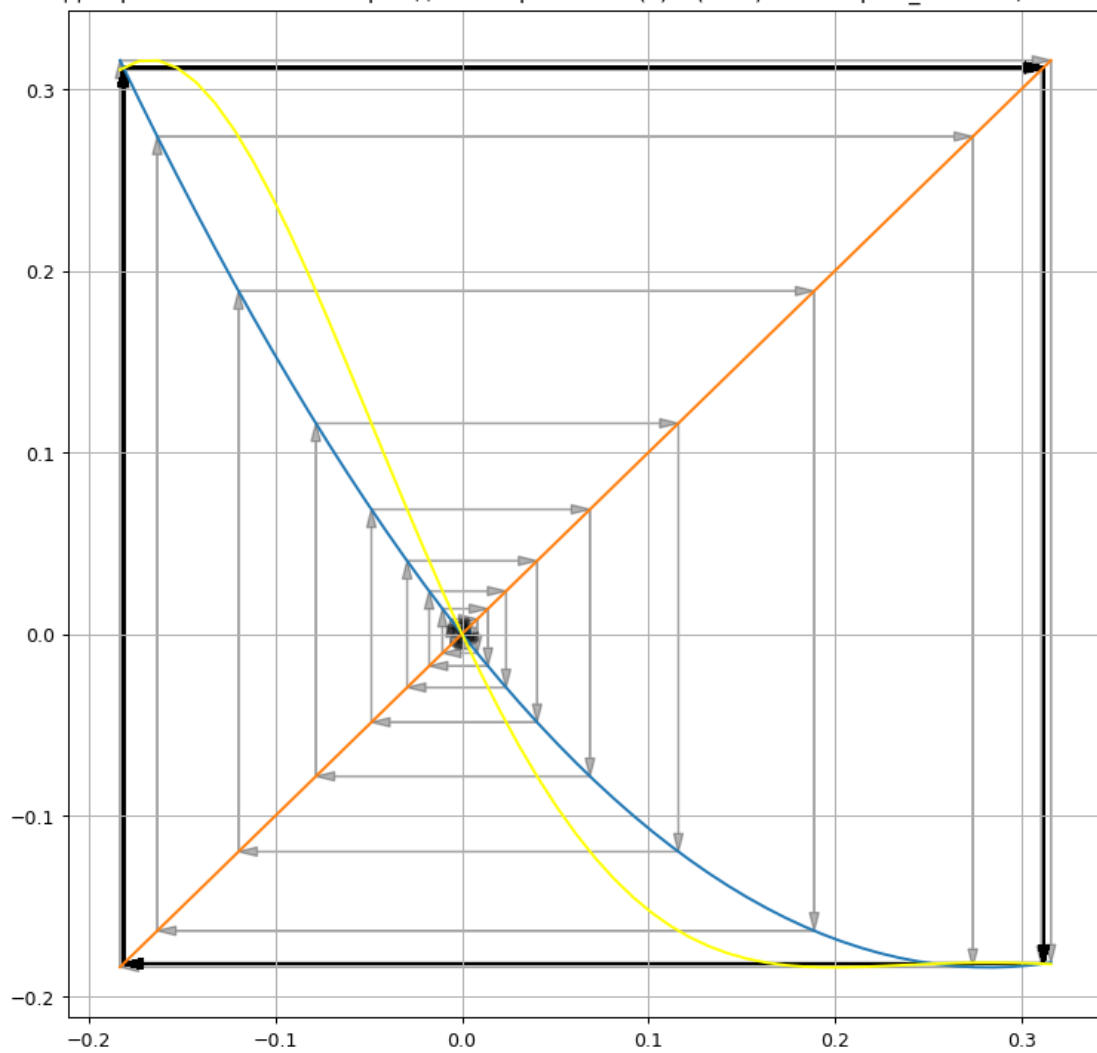
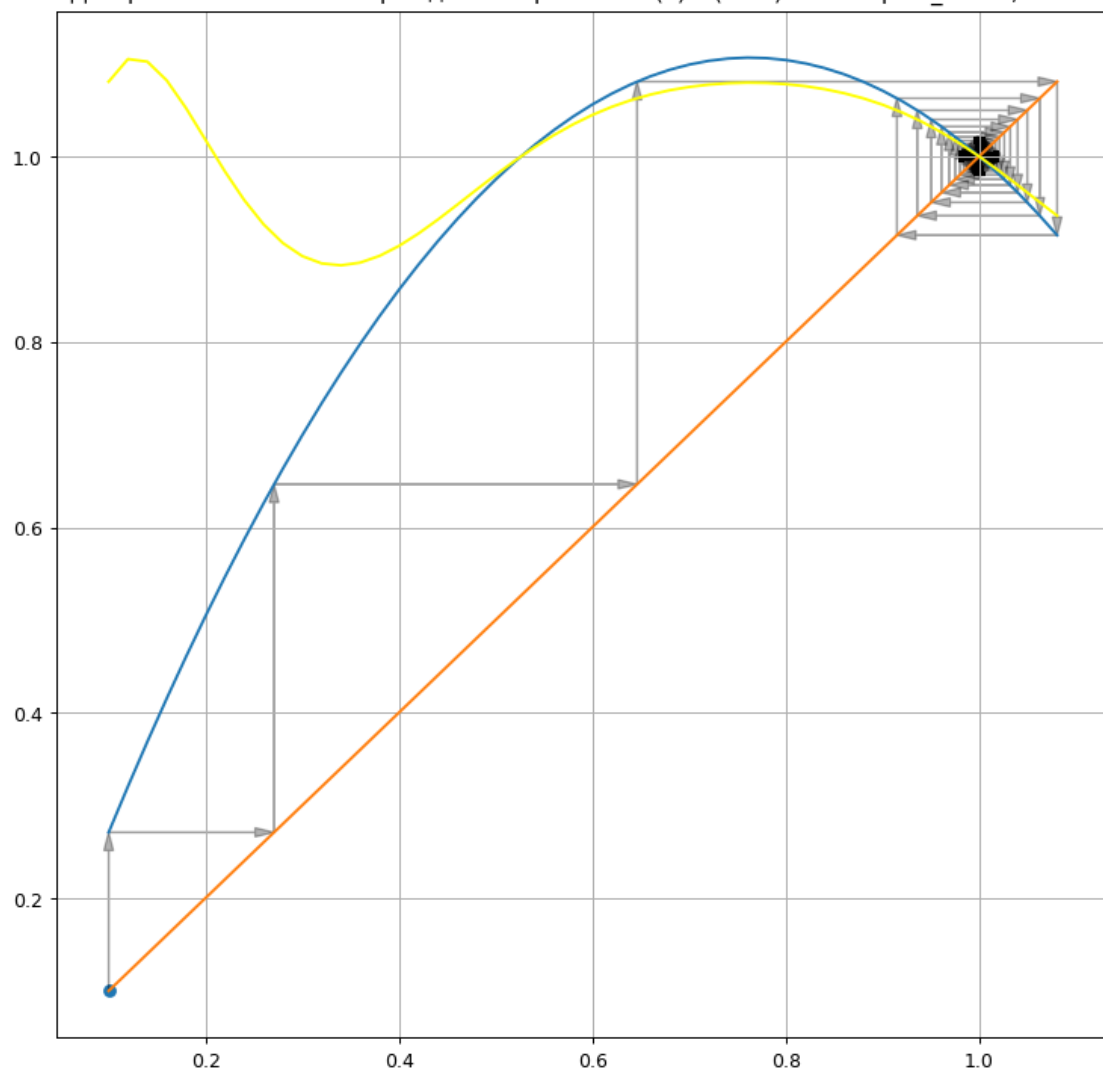
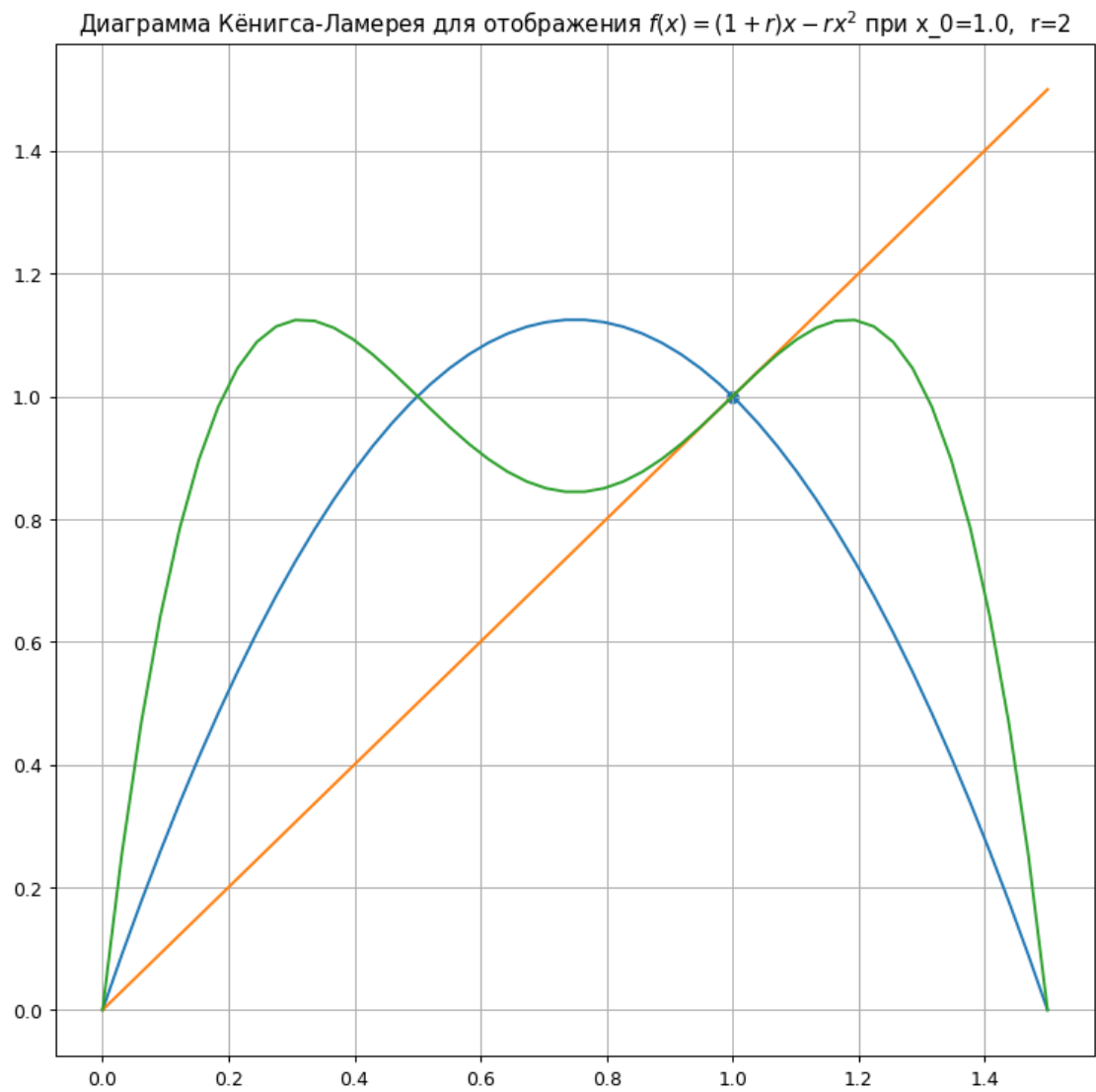


Диаграмма Кёнигса-Ламерея для отображения $f(x) = (1+r)x - rx^2$ при $x_0=0.1$, $r=1.9$





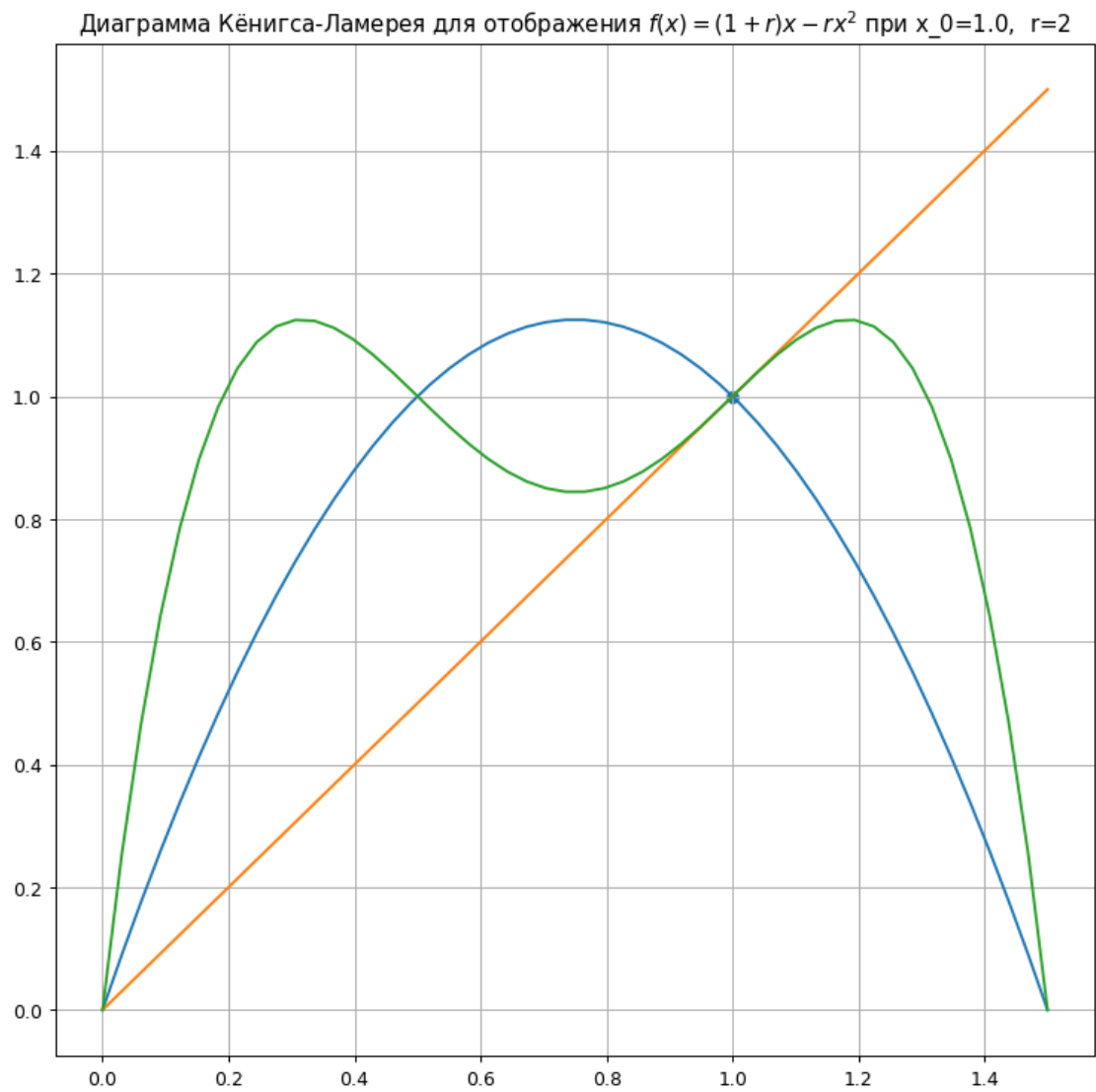


Диаграмма Кёнигса-Ламерея для отображения $f(x) = (1+r)x - rx^2$ при $x_0=0.8780487804878049$, $r=2.05$

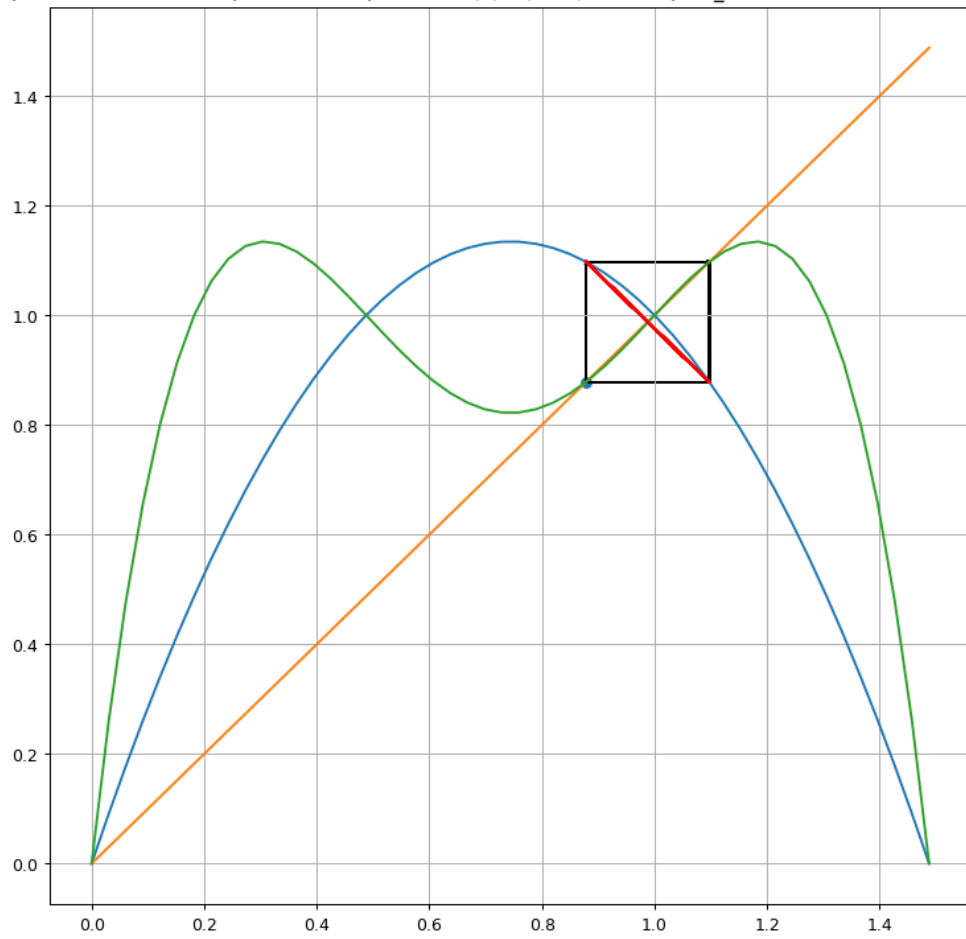


Диаграмма Кёнигса-Ламерея для отображения $f(x) = (1+r)x - rx^2$ при $x_0=1.097560975609756$, $r=2.05$

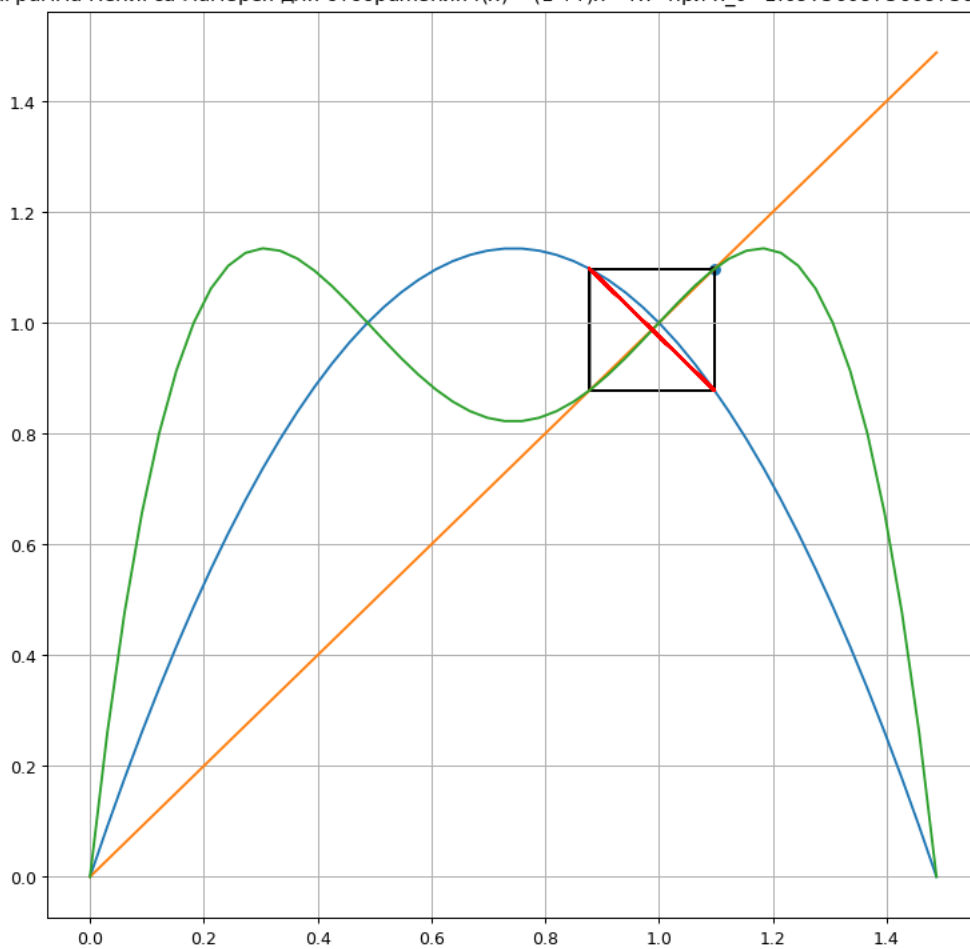


Диаграмма Кёнигса-Ламерея для отображения $f(x) = (1+r)x - rx^2$ при $x_0=0.1$, $r=2.1$

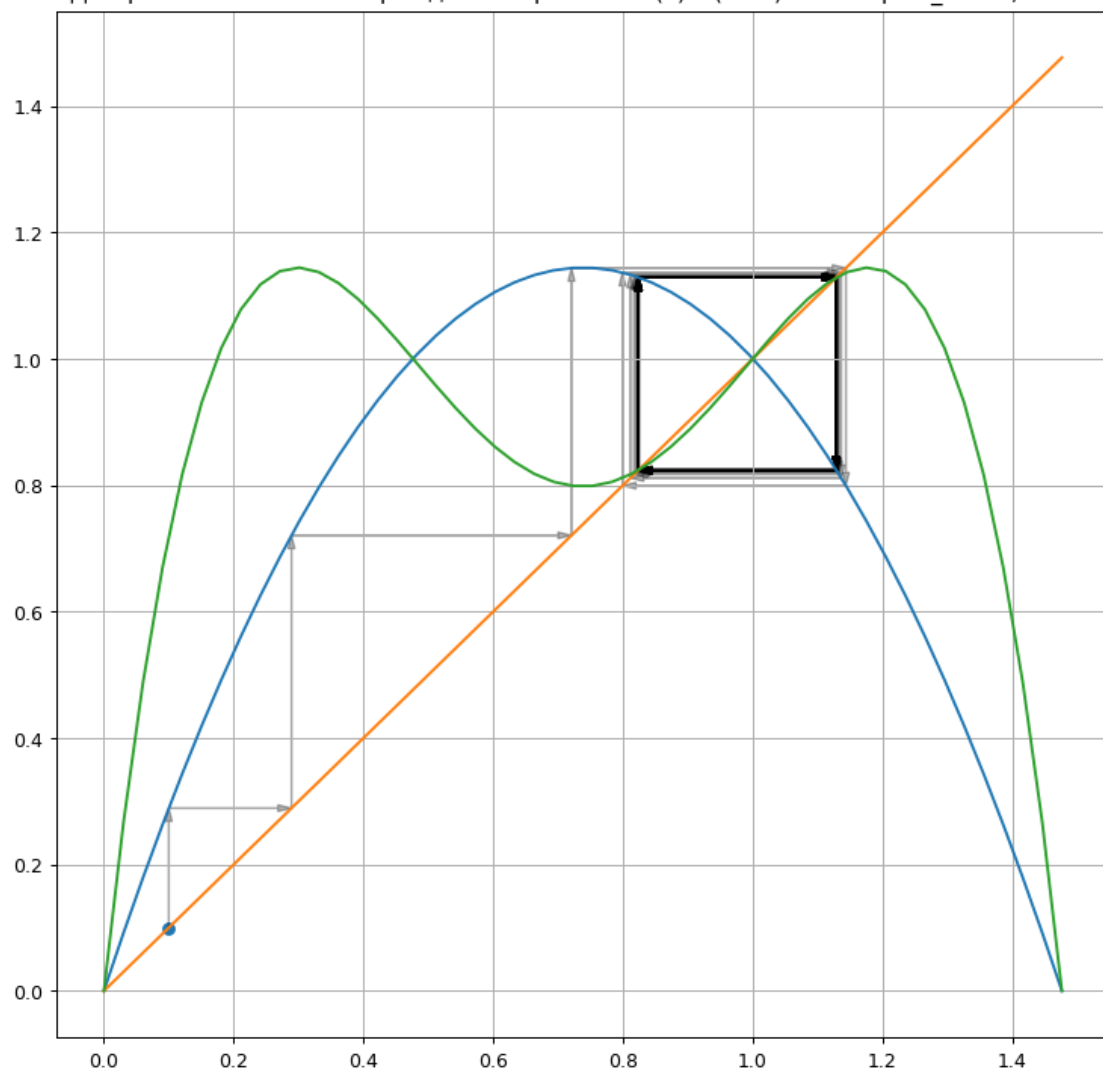


Диаграмма Кёнигса-Ламерея для отображения $f(x) = (1+r)x - rx^2$ при $x_0=0.1$, $r=2.3$

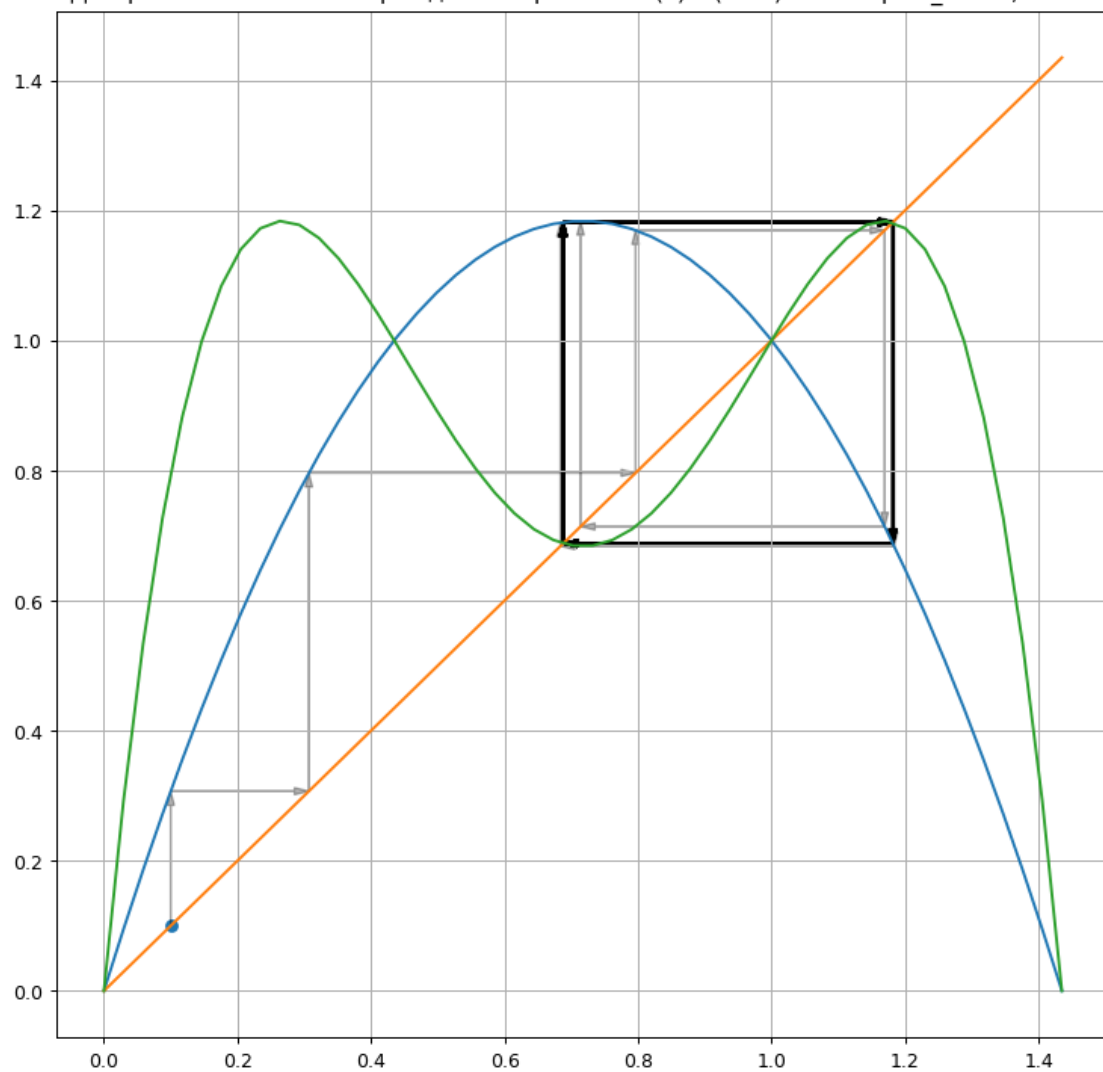


Диаграмма Кёнигса-Ламерея для отображения $f(x) = (1+r)x - rx^2$ при $x_0=0.1$, $r=2.45$

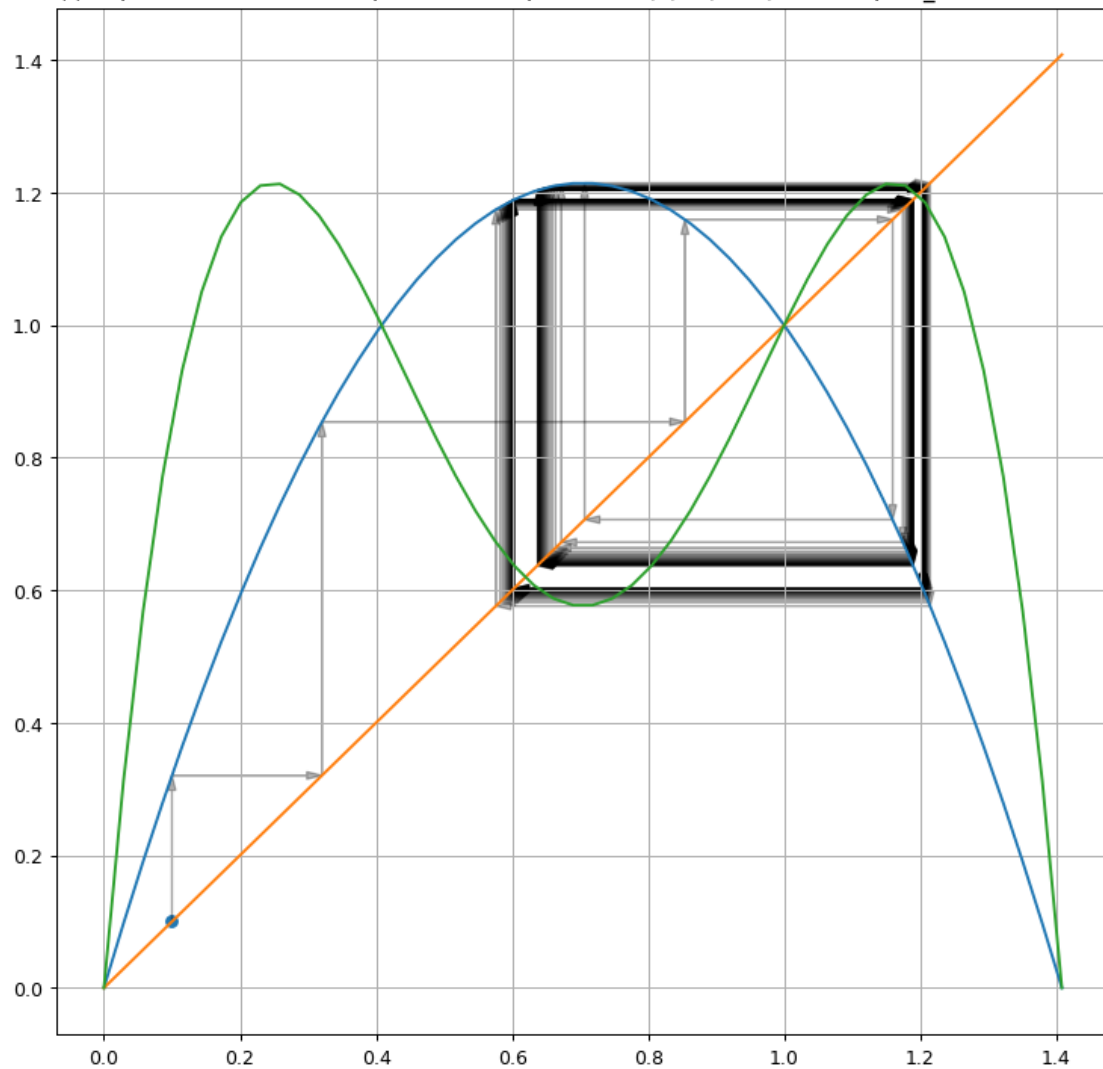


Диаграмма Кёнигса-Ламерея для отображения $f(x) = (1+r)x - rx^2$ при $x_0=0.1$, $r=2.5$

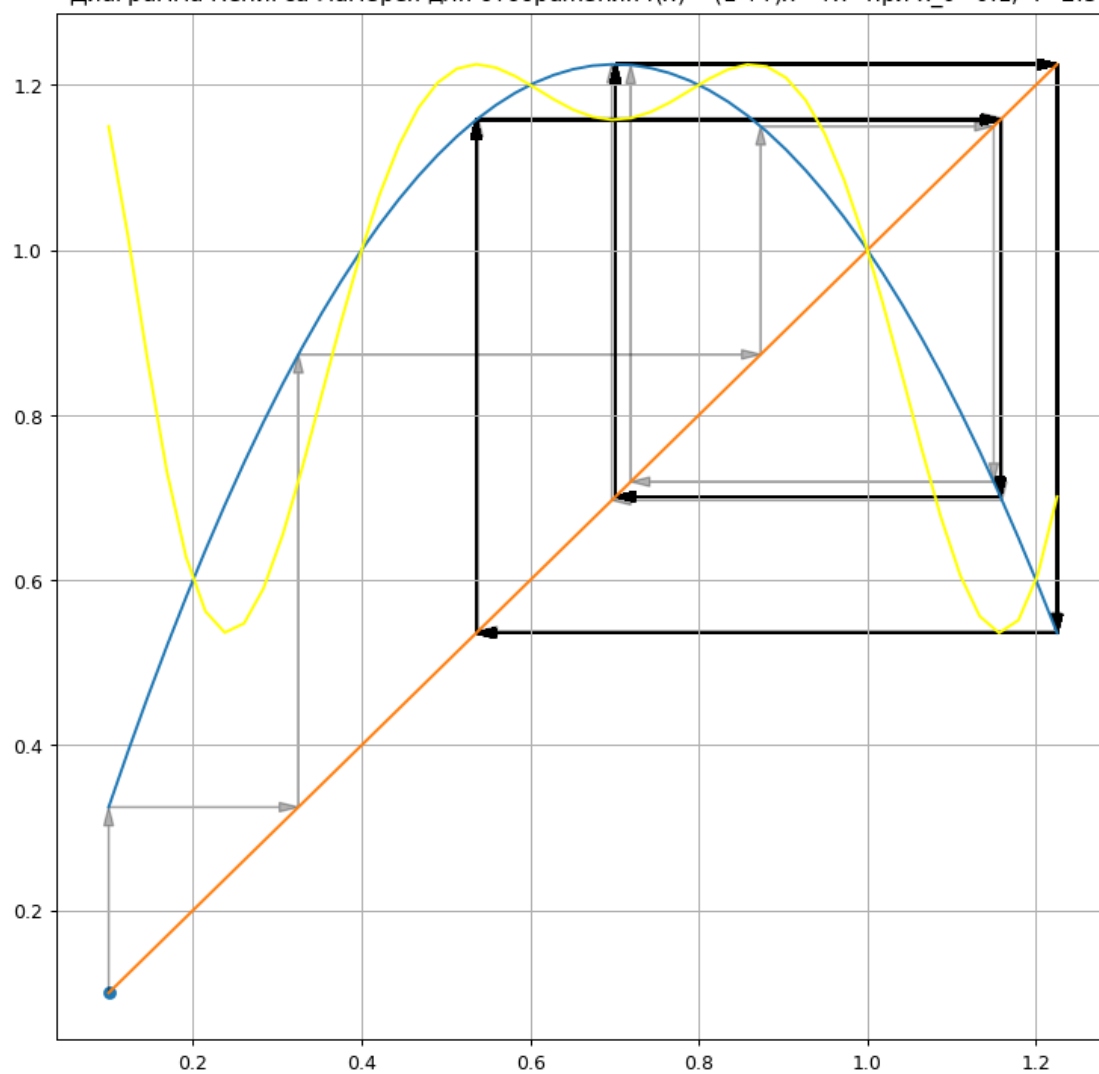
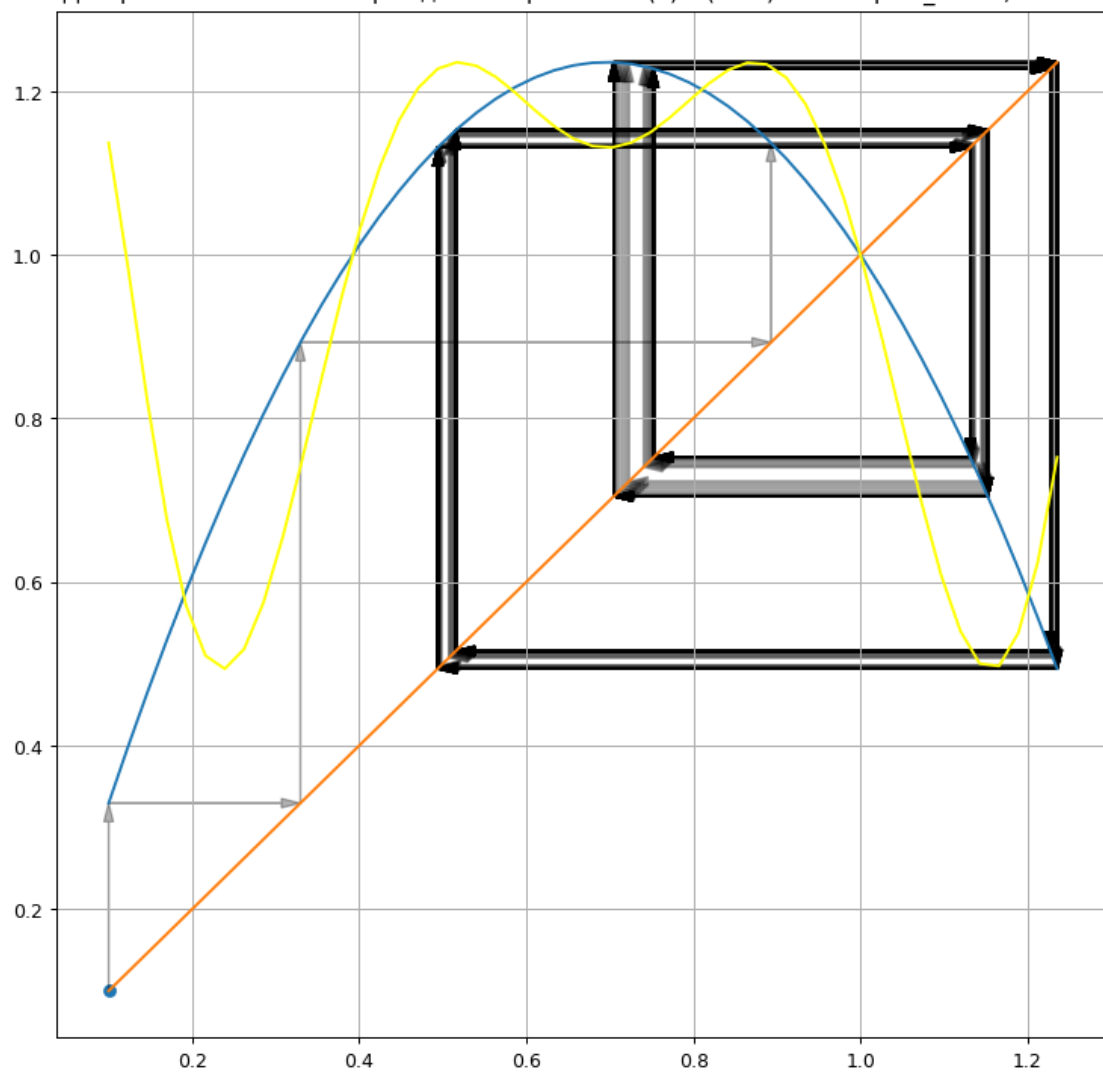
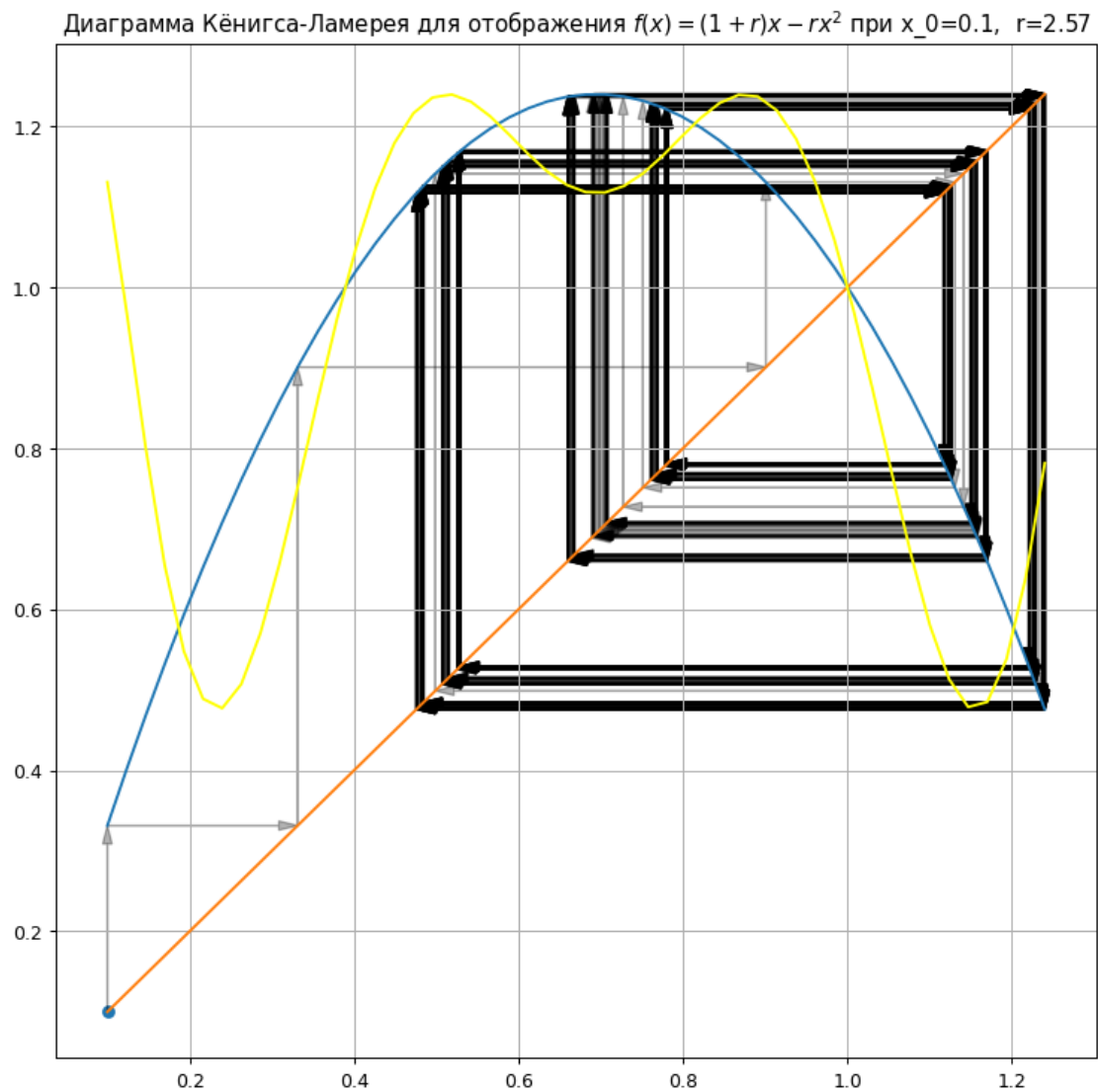


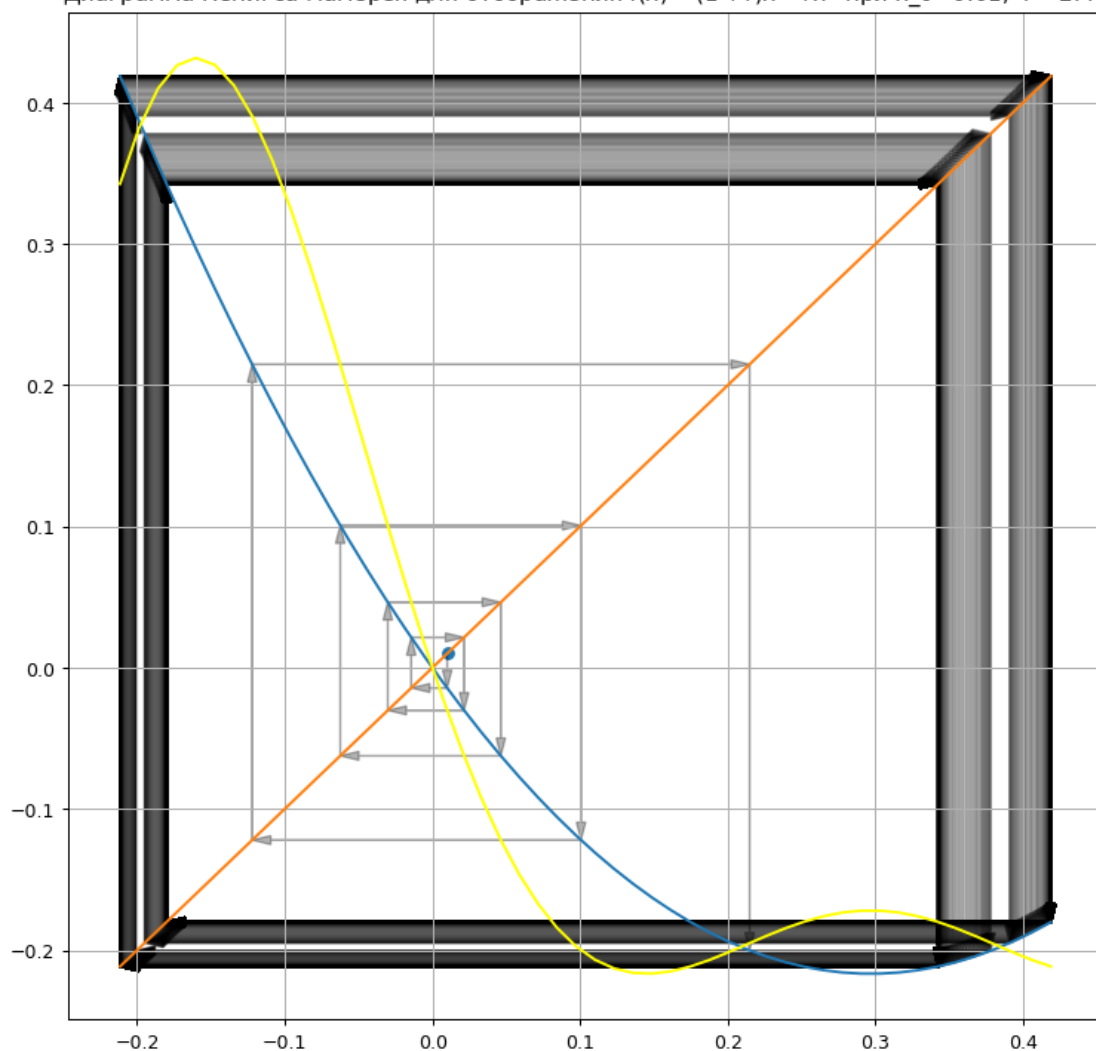
Диаграмма Кёнигса-Ламерея для отображения $f(x) = (1+r)x - rx^2$ при $x_0=0.1$, $r=2.55$





```
[13]: KenLam(1000, 0.01, -2.46, with_arrows=True, with_3cr=False, with_4cr=True,
↪arrow_c=['black'], with_lines=False)
```

Диаграмма Кёнигса-Ламерея для отображения $f(x) = (1+r)x - rx^2$ при $x_0=0.01$, $r=-2.46$



```
[14]: time = np.arange(0, 50)
fig = plt.figure(facecolor="white", figsize = big_pics, dpi=stat_dpi)

r = 2.3
ff = lambda x : (1 + r) * ((1 + r)*x - r*x**2) - r * ((1+r)*x-r*x**2)**2

plt.hlines(1, 0, len(time), color='black')
plt.hlines(0, 0, len(time), color='black')

xp3 = (2*r + r**2 - r*(-4 + r**2)**0.5)/(2*r**2)
xp4 = (2*r + r**2 + r*(-4 + r**2)**0.5)/(2*r**2)

plt.hlines(xp3, 0, len(time), color='black')
```



```

plt.hlines(xp4, 0, len(time), color='black')

x_0 = 1.4

plt.plot(discr_sol(x_0, r, len(time)), linestyle='dashed')
plt.plot(np.linspace(0, 50, 1000), logistic(np.linspace(0, 50, 1000), r, x_0, 0))
plt.title(r"                               r = "+str(r)+f"   {x_0=}")
plt.grid(True)
plt.show()

```

