# Отчет по лабораторной работе 8

**НФИбд-02-18**

Оразклычев Довлет

# Содержание

# List of Tables

# List of Figures

# 1 Цель работы

Постройте графики фирм для двух случаев

# 2 Задание

**Вариант 6**

**Случай 1**. Рассмотрим две фирмы, производящие взаимозаменяемые товары одинакового качества и находящиеся в одной рыночной нише. Считаем, что в рамках нашей модели конкурентная борьба ведётся только рыночными методами. То есть, конкуренты могут влиять на противника путем изменения параметров своего производства: себестоимость, время цикла, но не могут прямо вмешиваться в ситуацию на рынке («назначать» цену или влиять на потребителей каким-либо иным

способом.) Будем считать, что постоянные издержки пренебрежимо малы, и в модели учитывать не будем. В этом случае динамика изменения объемов продаж фирмы 1 и фирмы 2 описывается следующей системой уравнений:

$$\frac{dM_1}{d\theta} = M_1 - \frac{b}{c_1} M_1 M_2 - \frac{a_1}{c_1} M_1^2$$
$$\frac{dM_2}{d\theta} = \frac{c_2}{c_1} M_2 - \frac{b}{c_1} M_1 M_2 - \frac{a_2}{c_1} M_2^2,$$

где $a_1 = \frac{p_{cr}}{\tau_1^2 \tilde{p}_1^2 Nq}$, $a_2 = \frac{p_{cr}}{\tau_2^2 \tilde{p}_2^2 Nq}$, $b = \frac{p_{cr}}{\tau_1^2 \tilde{p}_1^2 \tau_2^2 \tilde{p}_2^2 Nq}$, $c_1 = \frac{p_{cr} - \tilde{p}_1}{\tau_1 \tilde{p}_1}$, $c_2 = \frac{p_{cr} - \tilde{p}_2}{\tau_2 \tilde{p}_2}$.

Также введена нормировка $t = c_1 \theta$.

Figure 2.1: Задание лабораторной работы

# 3 Выполнение лабораторной работы

Для начала мы импортируем библиотеки для построения кода и вводим наши переменные:

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

from scipy.integrate import odeint

p_cr = 20
tau1 = 10
p1 = 9
tau2 = 16
p2 = 7
V = 10
q = 1

a1 = p_cr / (tau1 * tau1 * p1 * p1 * V * q)
a2 = p_cr / (tau2 * tau2 * p2 * p2 * V * q)
b = p_cr / (tau1 * tau1 * tau2 * tau2 * p1 * p1 * p2 * p2 * V * q)
c1 = (p_cr - p1) / (tau1 * p1)
c2 = (p_cr - p1) / (tau2 * p2)

t0 = 0
```

```
tmax = 30
dt = 0.01
```

Теперь мы создаем список значений t, которое мы будем использовать чтобы вычислять поточечно значения:

```
t = np.arange(t0, tmax, dt)
t = np.append(t, tmax)
```

Обратите внимаение, что я также добавил элемент tmax в конец списка. Дело в том, что функция np.arange заполняет от нуля до tmax - dt, поэтому надо добавлять еще один элемент отдельно.

Теперь создаем систему уравнений:

```
def f(x, t):
    dx1 = (c1 / c1) * x[0] - ((a1 / c1) + 0.0015) * x[0] * x[0] - (b / c1
    dx2 = (c2 / c1) * x[1] - (a2 / c1) * x[1] * x[1] - (b / c1) * x[0] *
    return dx1, dx2
```

Запускаем команду odeint, которая найдет значения поточечно.

```
yf = odeint(f, v0, t)
```

Теперь создаем график и выводим на экран. график будет красного цвета с обозначением “x”. Размер графика 10 на 10 единиц.

```
plt.figure(figsize=(10, 10))
plt.plot(t, yf)
plt.show()
```

И получаем:

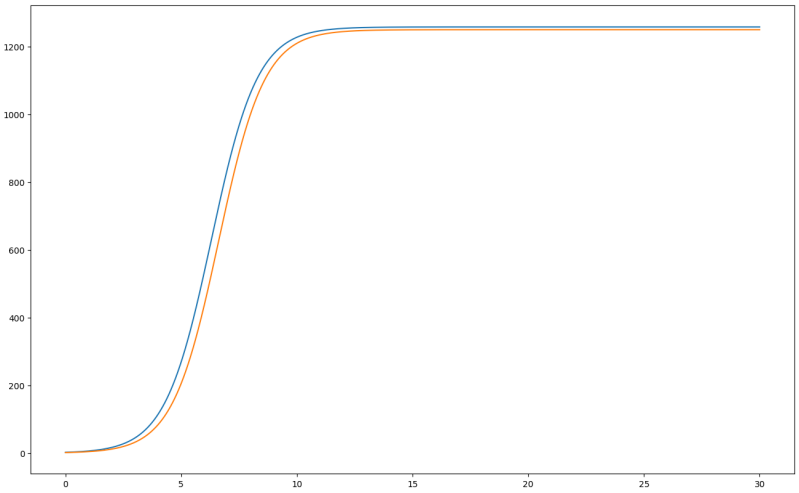


Figure 3.1: Случай 1

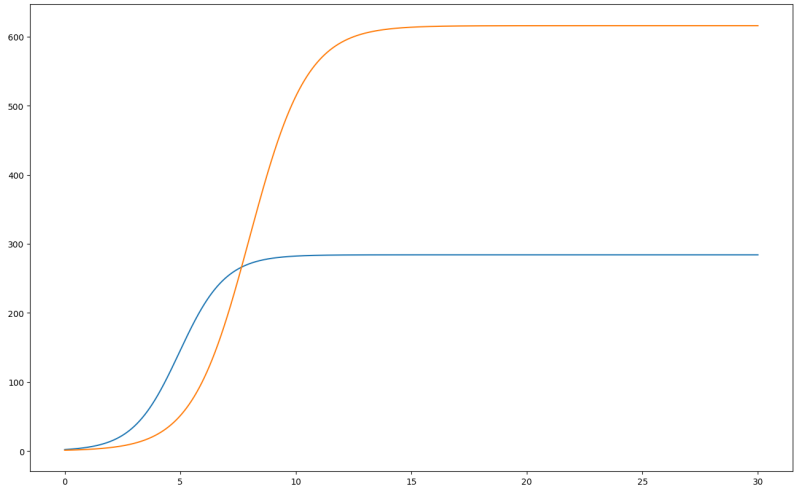


Figure 3.2: Случай 2

Код на Python для графика 1:

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

from scipy.integrate import odeint
```

```
p_cr = 18
tau1 = 14
p1 = 11
tau2 = 17
p2 = 9
V = 21
q = 1

a1 = p_cr / (tau1 * tau1 * p1 * p1 * V * q)
a2 = p_cr / (tau2 * tau2 * p2 * p2 * V * q)
b = p_cr / (tau1 * tau1 * tau2 * tau2 * p1 * p1 * p2 * p2 * V * q)
c1 = (p_cr - p1) / (tau1 * p1)
c2 = (p_cr - p1) / (tau2 * p2)


t0 = 0
tmax = 30
dt = 0.01

t = np.arange(t0, tmax, dt)
t = np.append(t, tmax)


def f(x, t):
    dx1 = (c1 / c1) * x[0] - (a1 / c1) * x[0] * x[0] - (b / c1) * x[0] *
    dx2 = (c2 / c1) * x[1] - (a2 / c1) * x[1] * x[1] - (b / c1) * x[0] *
    return dx1,dx2
```

```
x0 = [2.3,1.6]
yf = odeint(f, x0, t)


plt.figure(figsize=(10, 10))
plt.plot(t, yf)
plt.show()
```

Код на Python для графика 2:

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

from scipy.integrate import odeint

p_cr = 20
tau1 = 10
p1 = 9
tau2 = 16
p2 = 7
V = 10
q = 1

a1 = p_cr / (tau1 * tau1 * p1 * p1 * V * q)
a2 = p_cr / (tau2 * tau2 * p2 * p2 * V * q)
b = p_cr / (tau1 * tau1 * tau2 * tau2 * p1 * p1 * p2 * p2 * V * q)
c1 = (p_cr - p1) / (tau1 * p1)
c2 = (p_cr - p1) / (tau2 * p2)

t0 = 0
tmax = 30
```

```
dt = 0.01

t = np.arange(t0, tmax, dt)
t = np.append(t, tmax)
v0 = [2, 1]


def f(x, t):
    dx1 = (c1 / c1) * x[0] - ((a1 / c1) + 0.0015) * x[0] * x[0] - (b / c1
    dx2 = (c2 / c1) * x[1] - (a2 / c1) * x[1] * x[1] - (b / c1) * x[0] *
    return dx1, dx2


yf = odeint(f, v0, t)

plt.figure(figsize=(10, 10))
plt.plot(t, yf)
plt.show()
```

# 4 Вывод

Построили код на Python для решения и вывода на экран графиков для 2 случаев.