Московский государственный университет имени М.В.Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра математической физики

Отчет

“Исследование устойчивости стационарных состояний нелинейных систем второго порядка. Построение параметрического портрета системы. Автоколебания и множественность стационарных решений.”

Выполнила:
Студентка 601 группы
Зимина Ольга Николаевна

Описание задачи

Рассматривается автокаталитическая химическая реакция, происходящая на поверхности катализатора и основанная на кинетической схеме Ленгмюра-Хиншельвуда:

$$\frac{dx}{dt} = k_1 z - k_{-1} x - k_3{}^0 (1-x)^{\alpha} xy, \qquad (1)$$

$$\frac{dy}{dt} = k_2 z^2 - k_{-2} y^2 - k_3{}^0 (1-x)^{\alpha} xy \qquad (2)$$

Здесь

$$z = 1 - x - y,\ 0 \le x \le 1,\ 0 \le y \le 1,\ 0 \le x + y \le 1$$

Базовый набор параметров в модели №6:

$$\alpha = 18,\ k_1 = 0.012,\ k_{-1} = 0.01,\ k_{-2} = 10^{-9},\ k_3{}^0 = 10,\ k_2 = 0.012$$

Однопараметрический анализ по k_2

Стационарные состояния удовлетворяют системе уравнений:

$$k_1 z - k_{-1} x - k_3{}^0 (1-x)^{\alpha} xy = 0 \qquad (3)$$

$$k_2 z^2 - k_{-2} y^2 - k_3{}^0 (1-x)^{\alpha} xy = 0 \qquad (4)$$

Из уравнения (1) выразим переменную y через x и подставим в уравнение (2). Из получившегося уравнения выразим k_2:

$$k_2 = \frac{k_3{}^0 (1-x)^{\alpha} xy + k_{-2} y^2}{1-x-y},$$

Где

$$y = \frac{k_{-1}x - k_1(1-x)}{-k_1 - k_3{}^0(1-x)^{\alpha}x}$$

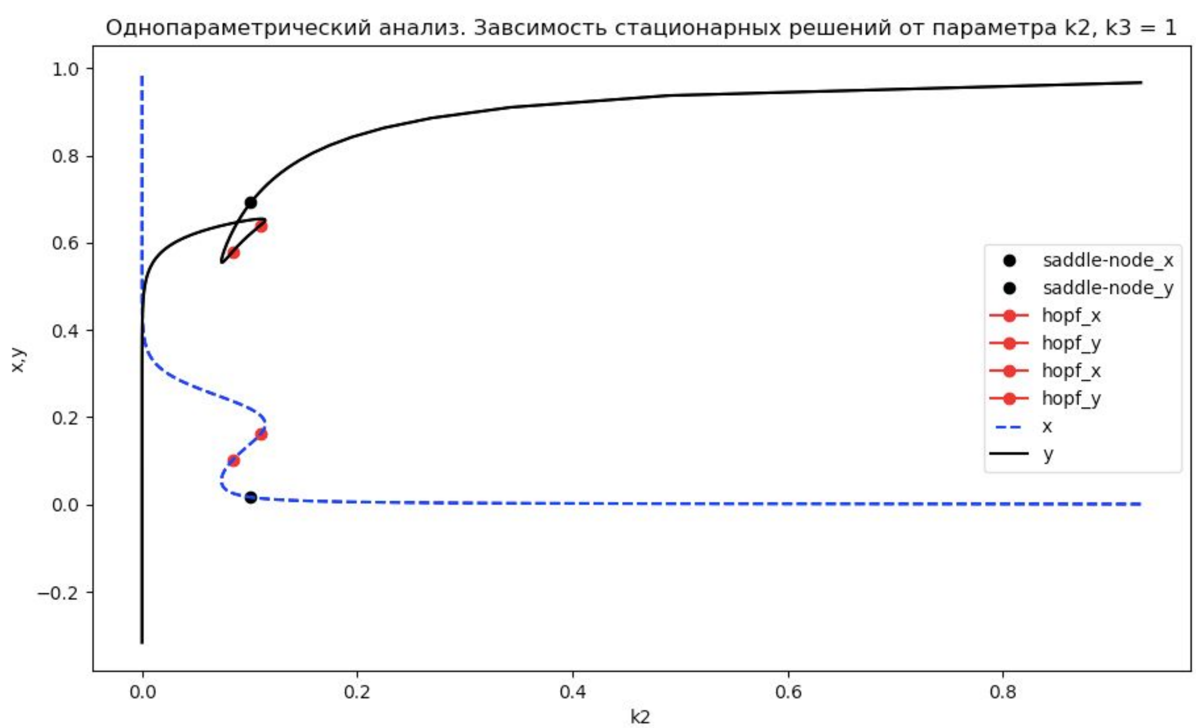
Пробегая с некоторым шагом весь диапазон значений переменной x от 0 до 1, по полученным формулам найдем соответствующие значения переменной y и k_2.

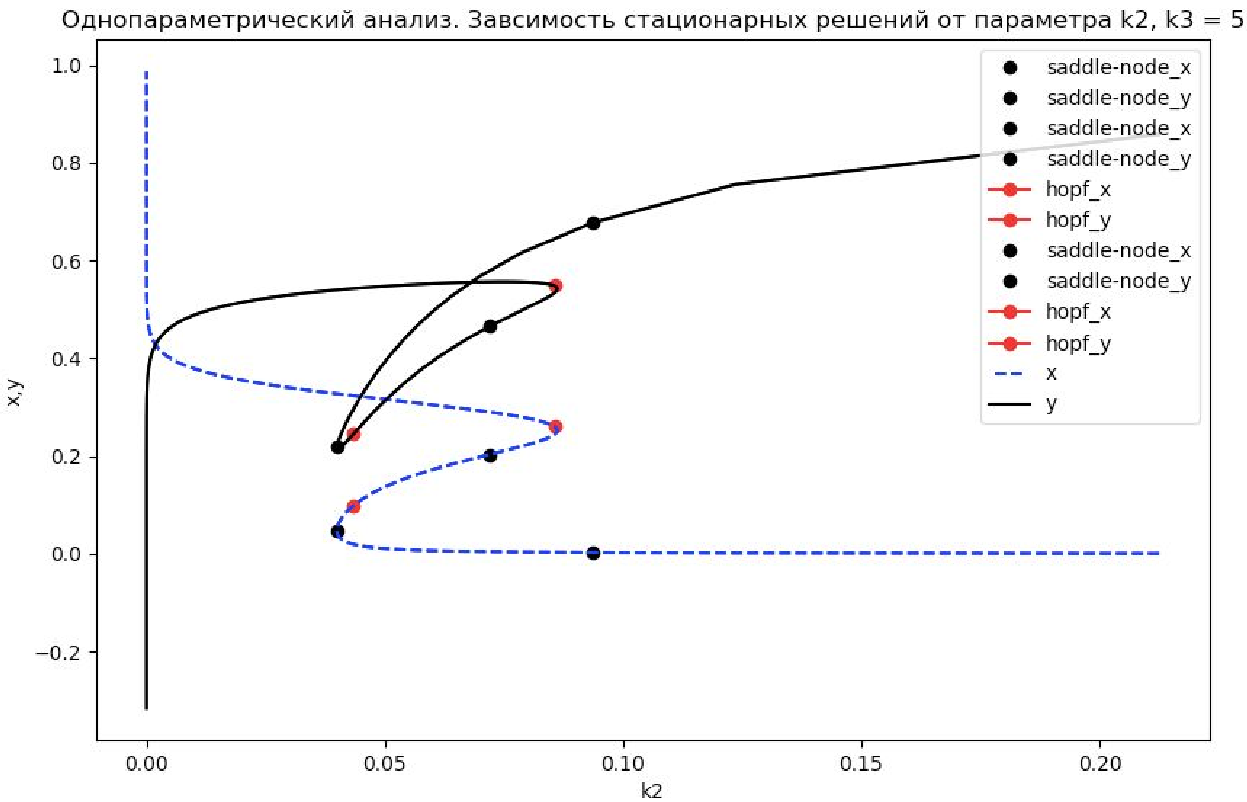
Для исследования устойчивости стационарных решений, необходимо

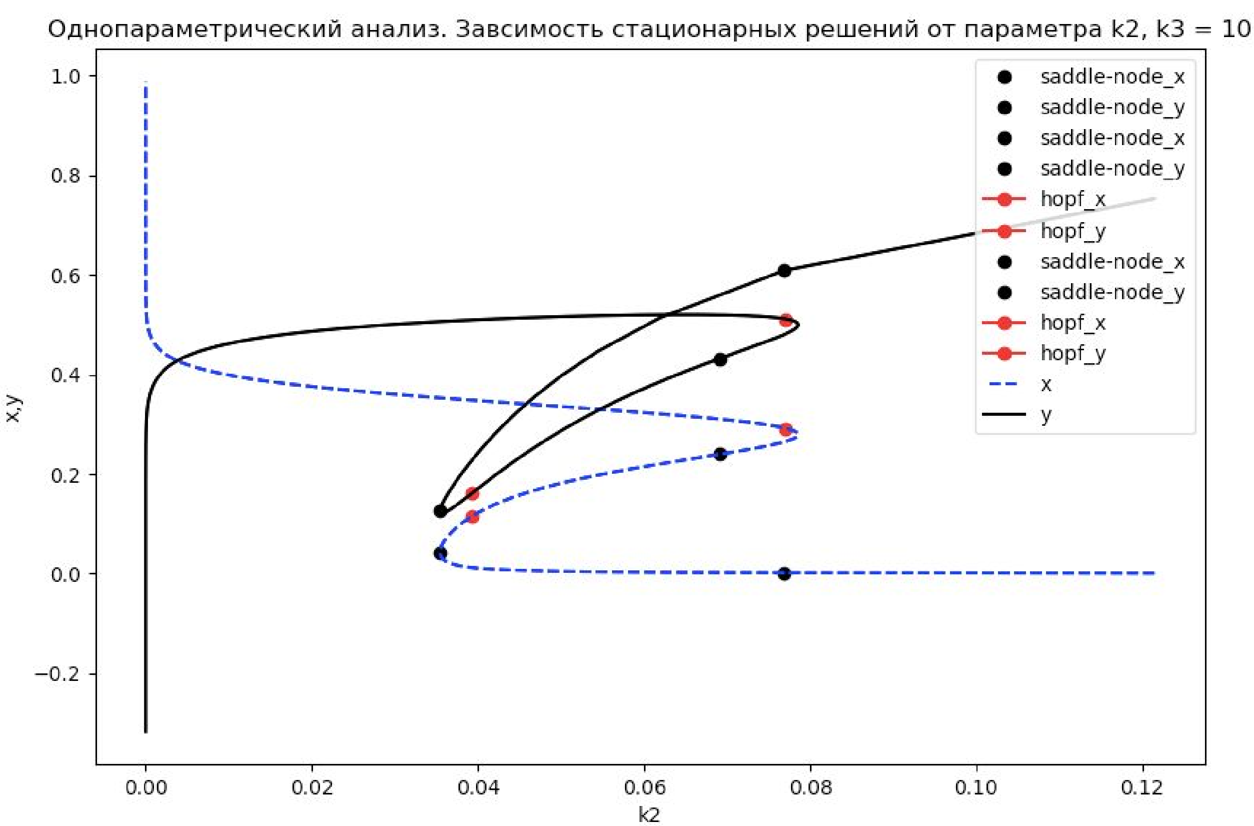
найти элементы матрицы Якоби, вычислить ее след и определитель.

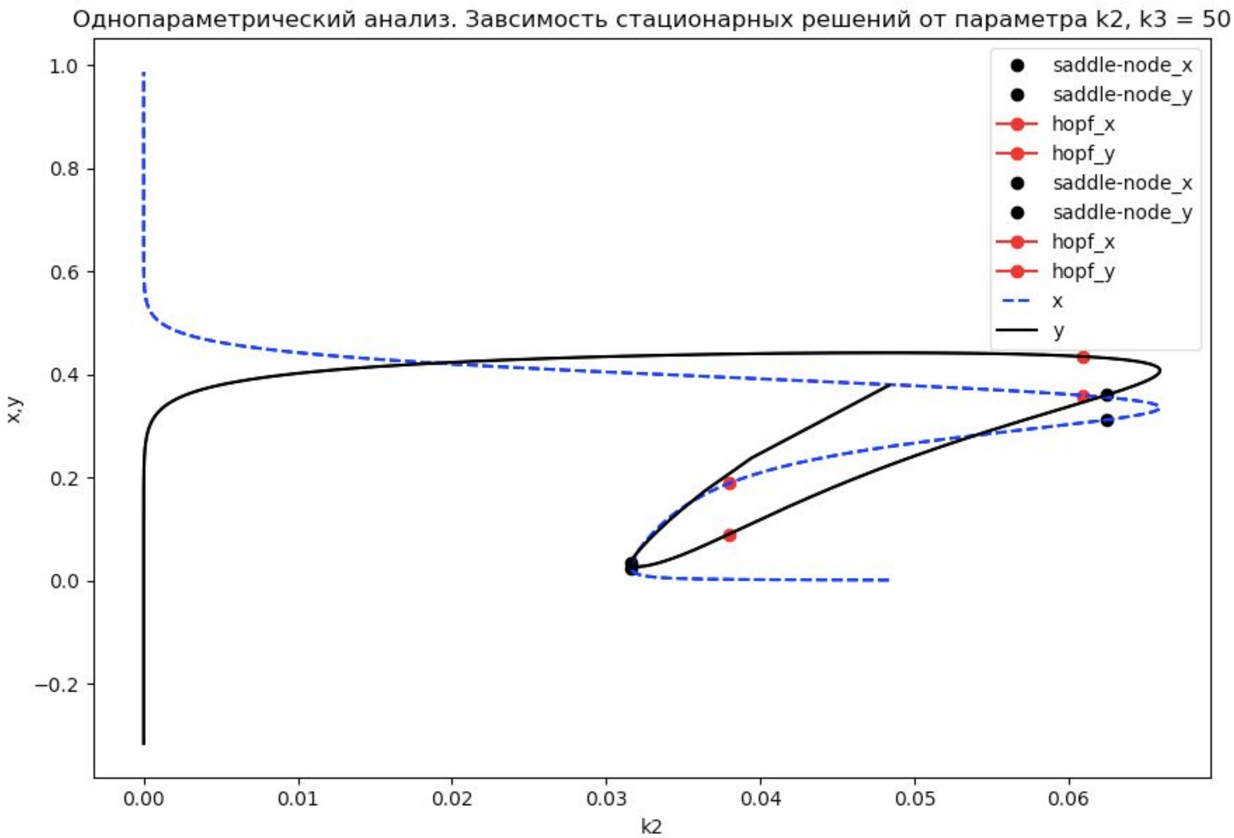
Это делается программно. Отслеживая смены знака якобиана системы (3),(4) на стационарном решении, мы находим точки бифуркации.

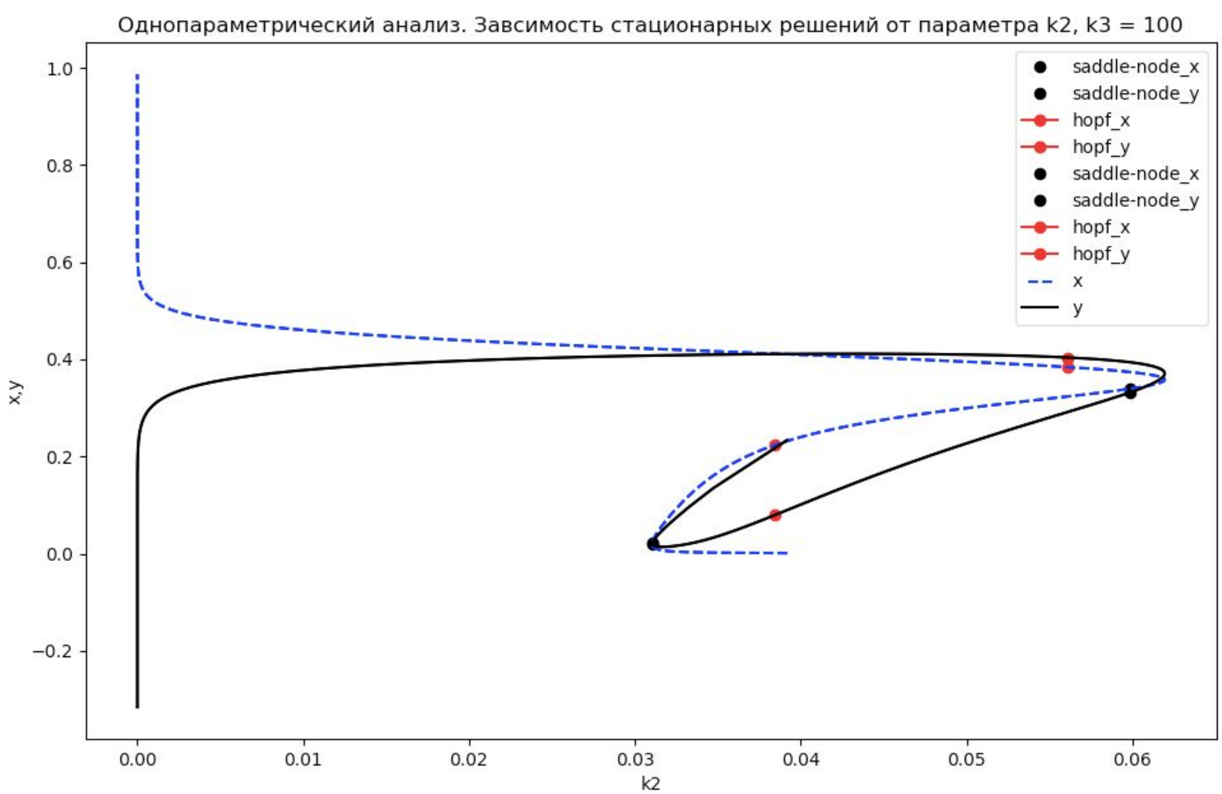
Ниже представлены результаты однопараметрического анализа по параметру k_2 при разных значениях параметра k_1



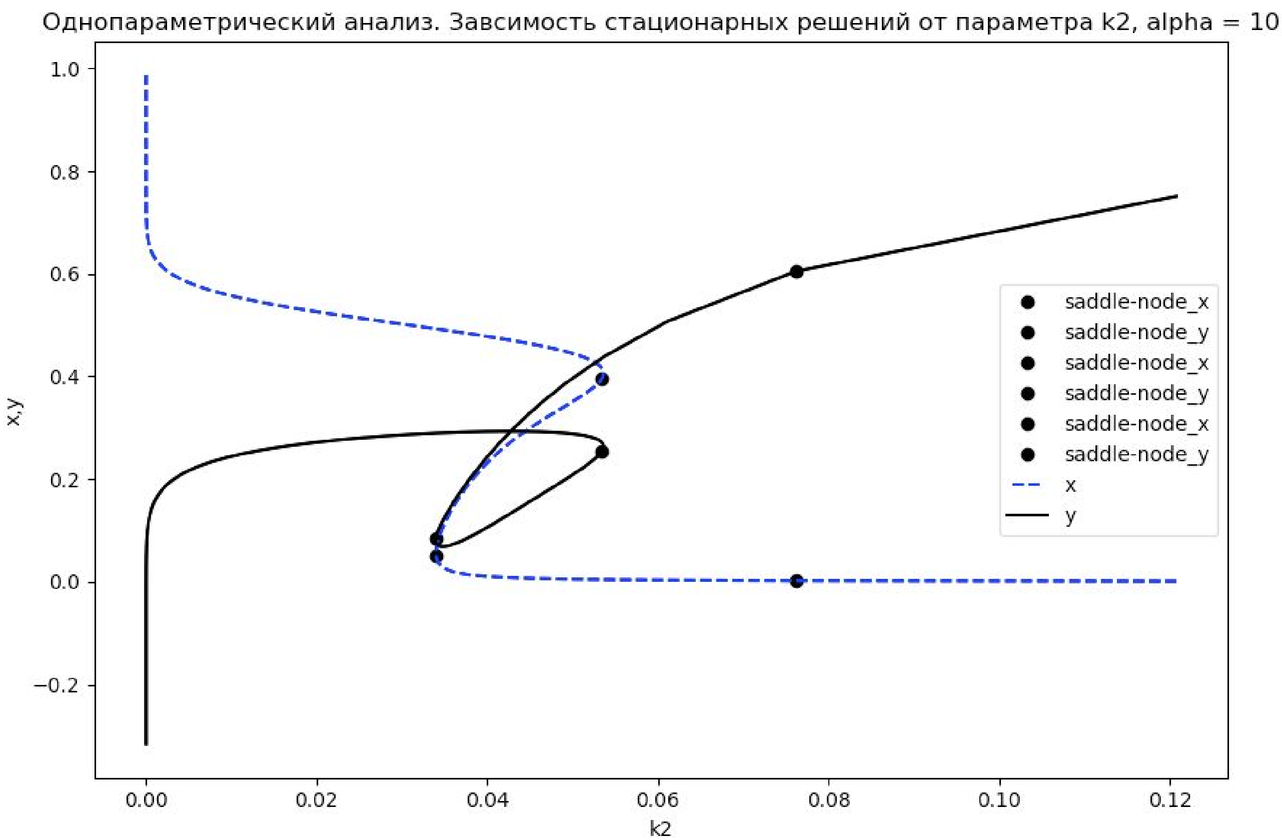

Однопараметрический анализ. Завсимость стационарных решений от параметра k2, k3 = 5
saddle-node_x
saddle-node_y
saddle-node_x
saddle-node_y
hopf_x
hopf_y
saddle-node_x
saddle-node_y
hopf_x
hopf_y
x
y
x,y
k2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.00
0.05
0.10
0.15
0.20


Однопараметрический анализ. Завсимость стационарных решений от параметра k2, k3 = 10
saddle-node_x
saddle-node_y
saddle-node_x
saddle-node_y
hopf_x
hopf_y
saddle-node_x
saddle-node_y
hopf_x
hopf_y
x
y
x,y
k2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.00
0.02
0.04
0.06
0.08
0.10
0.12

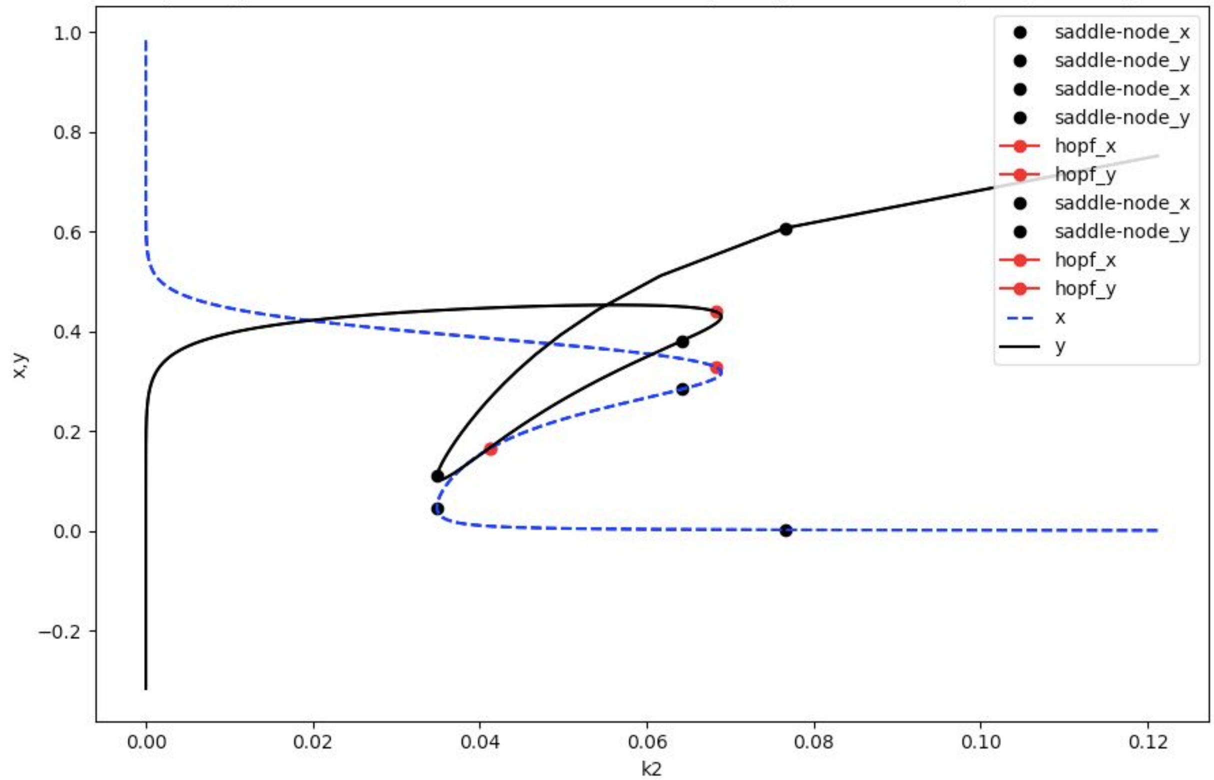

Однопараметрический анализ. Завсимость стационарных решений от параметра k2, k3 = 50
saddle-node_x
saddle-node_y
hopf_x
hopf_y
saddle-node_x
saddle-node_y
hopf_x
hopf_y
x
y
x,y
k2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.00
0.01
0.02
0.03
0.04
0.05
0.06


Однопараметрический анализ. Завсимость стационарных решений от параметра k2, k3 = 100
saddle-node_x
saddle-node_y
hopf_x
hopf_y
saddle-node_x
saddle-node_y
hopf_x
hopf_y
x
y
x,y
k2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.00
0.01
0.02
0.03
0.04
0.05
0.06

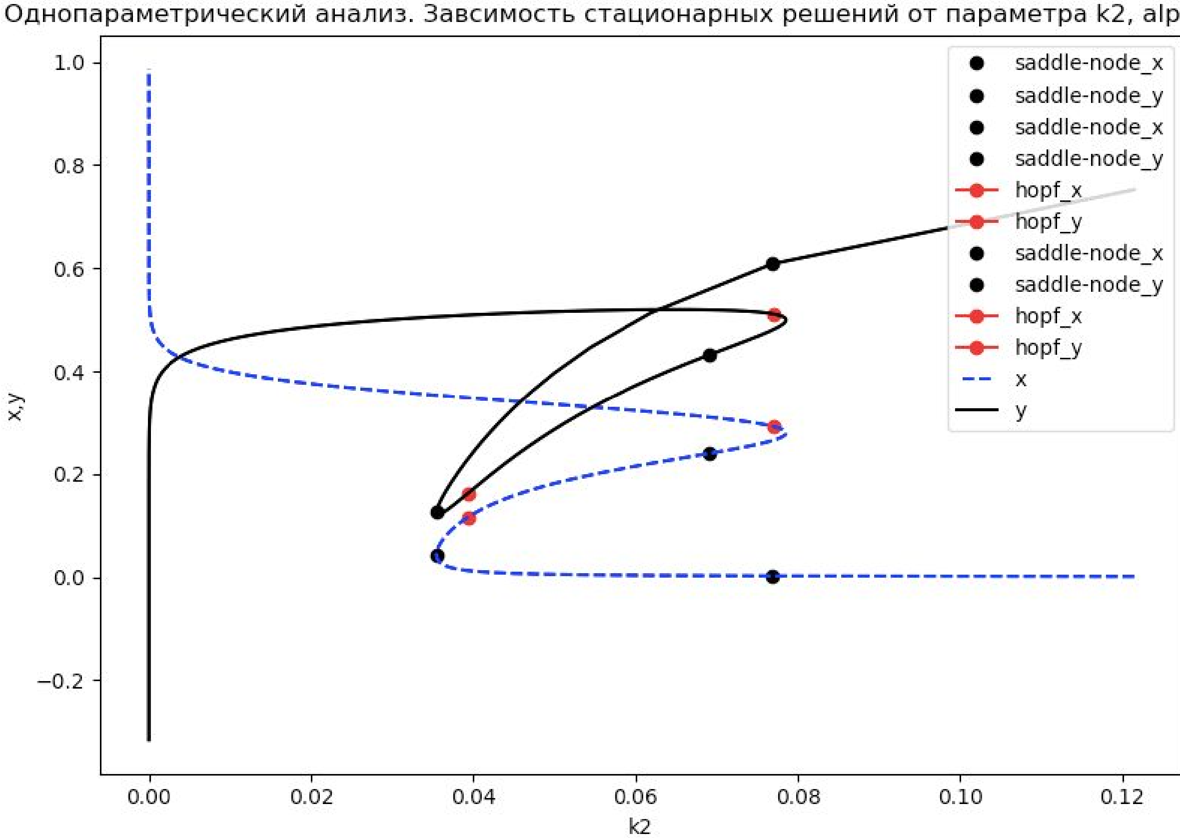
Ниже представлены результаты однопараметрического анализа по параметру k_2 при разных значениях параметра α

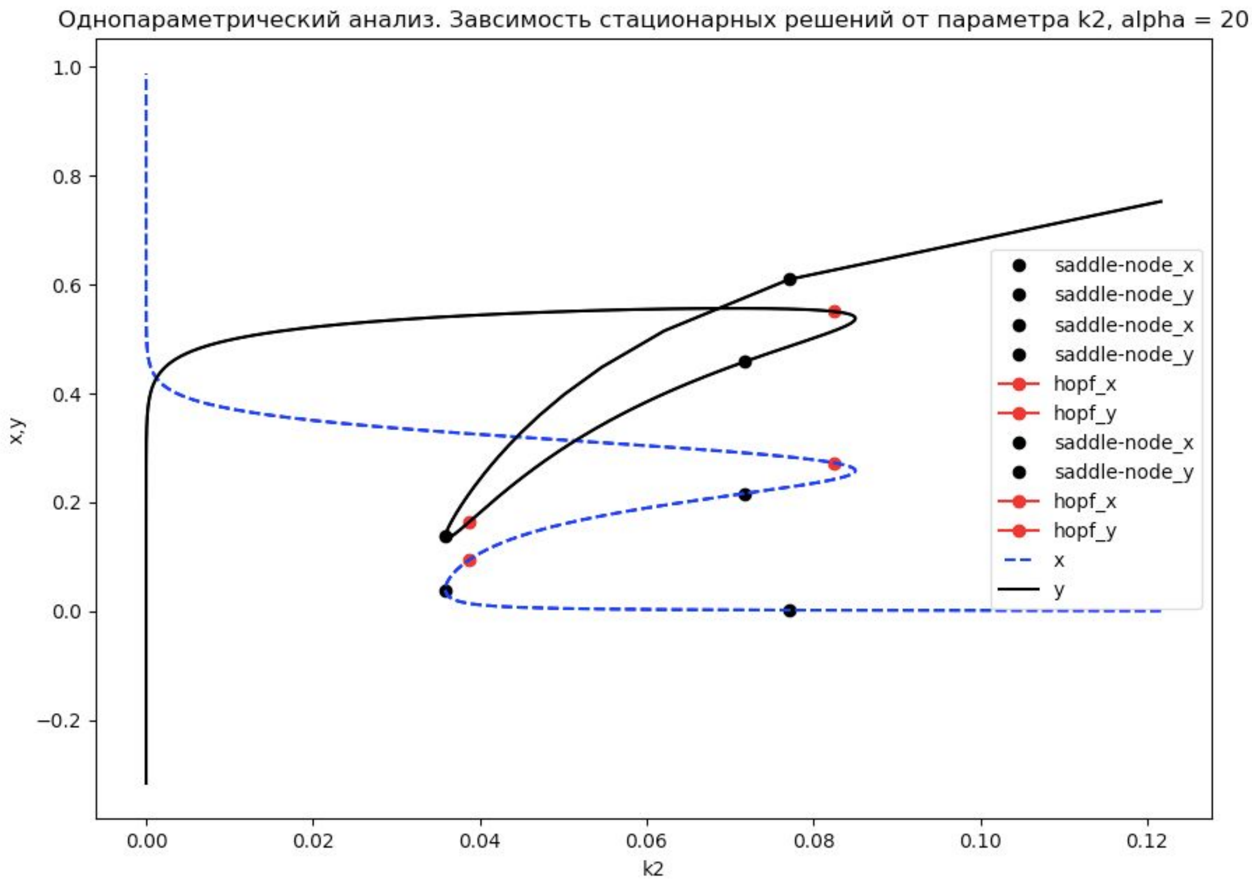


Однопараметрический анализ. Завсимость стационарных решений от параметра k2, alpha = 15

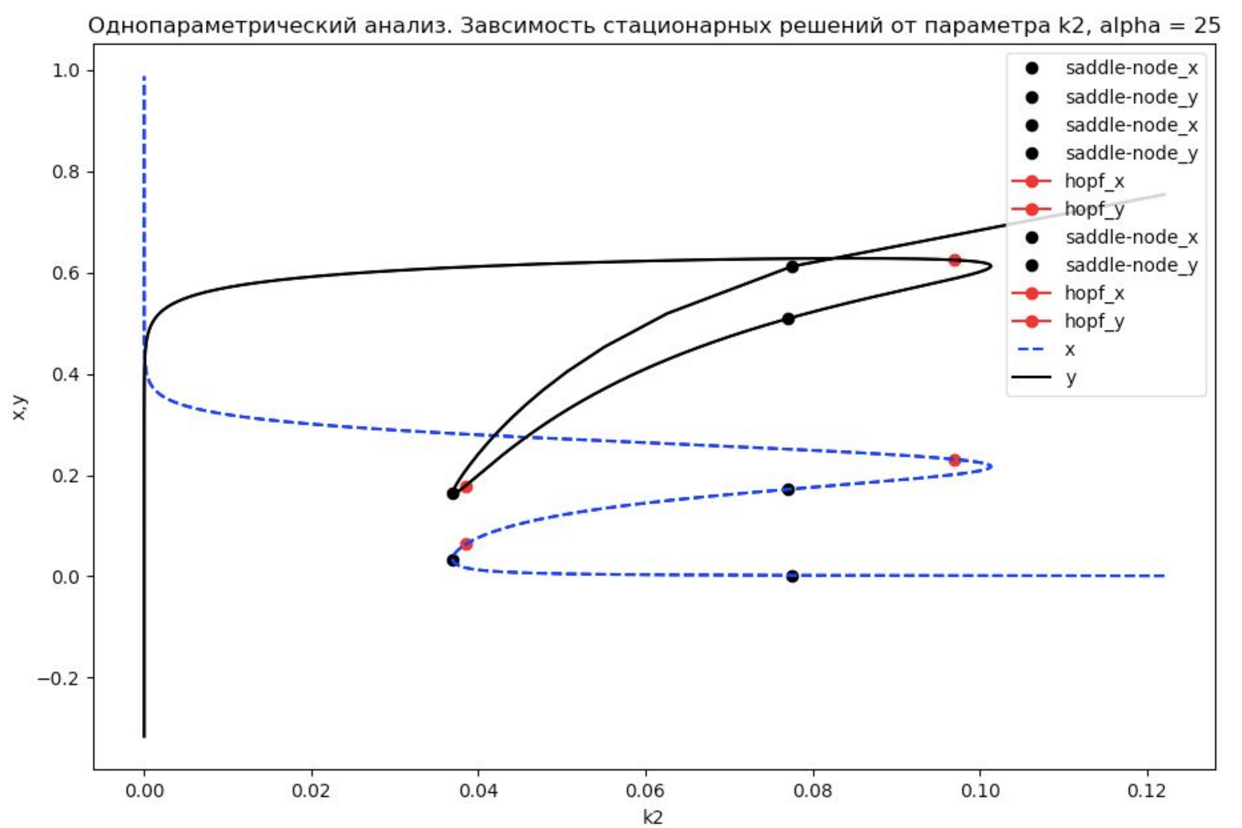


Однопараметрический анализ. Завсимость стационарных решений от параметра k2, alpha = 18





Однопараметрический анализ. Завсимость стационарных решений от параметра k2, alpha = 20
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
x,y
0.00
0.02
0.04
0.06
0.08
0.10
0.12
k2
saddle-node_x
saddle-node_y
saddle-node_x
saddle-node_y
hopf_x
hopf_y
saddle-node_x
saddle-node_y
hopf_x
hopf_y
x
y

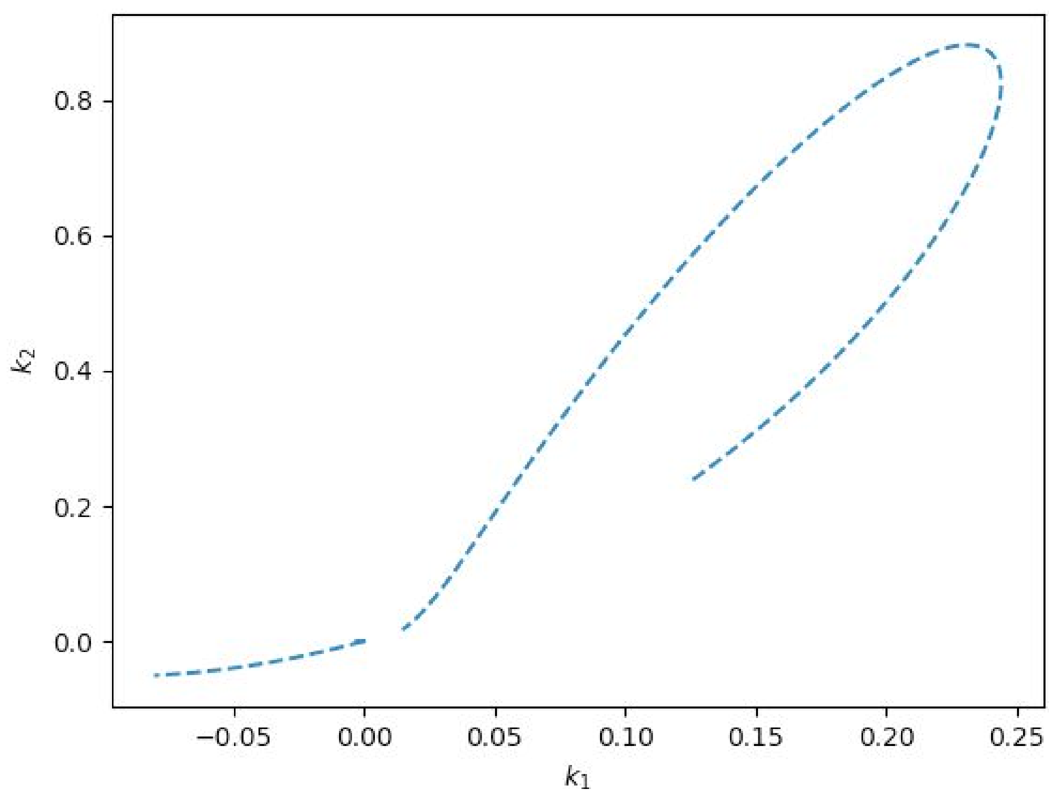


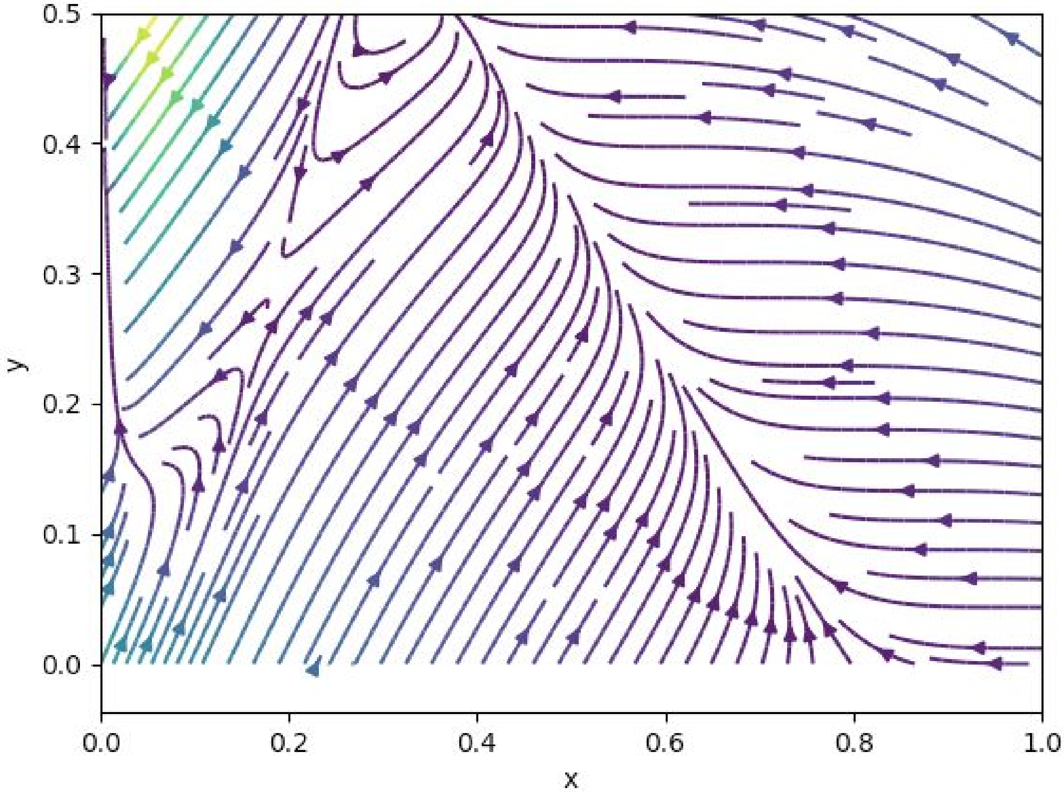
Однопараметрический анализ. Завсимость стационарных решений от параметра k2, alpha = 25
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
x,y
0.00
0.02
0.04
0.06
0.08
0.10
0.12
k2
saddle-node_x
saddle-node_y
saddle-node_x
saddle-node_y
hopf_x
hopf_y
saddle-node_x
saddle-node_y
hopf_x
hopf_y
x
y

Двухпараметрический анализ $(k1, k2)$

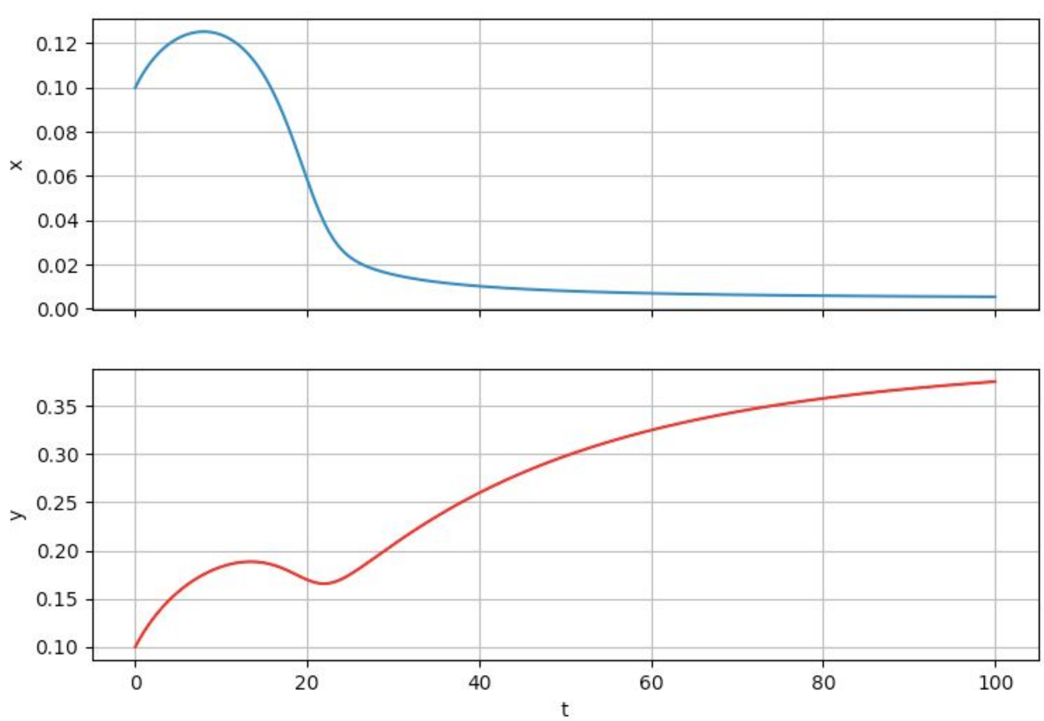
На плоскости параметров $(k1, k2)$ построим параметрический портрет системы: для этого необходимо построить линии кратности и нейтральности. Дописав для системы стационаров (1),(2) условие вырожденности матрицы Якоби, получим линию кратности. Если допишем условие равенства нулю следа матрицы Якоби, получим линию нейтральности. Решение не выходит на колебательный режим.

Ниже представлены параметрический портрет, фазовый портрет и колебания соответственно.





0.5
0.4
0.3
0.2
0.1
0.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x



0.12
0.10
0.08
0.06
0.04
0.02
0.00
x
0.35
0.30
0.25
0.20
0.15
0.10
y
0
20
40
60
80
100
t

Приложение

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.legend_handler import HandlerLine2D
from sympy import Symbol, solve, lambdify, Matrix, simplify
import sympy as smp
from scipy.integrate import odeint

#переменные
x = Symbol("x", Positive=True)
y = Symbol("y", Positive=True)
k1 = Symbol('k1', Positive=True)
k2 = Symbol('k2', Positive=True)
k_1 = Symbol('k_1', Positive=True)
k_2 = Symbol('k_2', Positive=True)
k3 = Symbol('k3', Positive=True)
alpha = Symbol('alpha', Positive=True)

#значения
alpha_value = 18.0
k1_value = 0.03
k2_value = 0.05
k_1_value = 0.01
k_2_value = 10**(-9)
k3_value = 10.0 #k30

#Однопараметрический анализ 2 - значения
alpha_range = [10,15,18,20,25]
k3_range = [1,5,10,50,100]

#Система уравнений для поиска стационарных состояний
equation_1 = k1 * (1 - x - y)- k_1 * x - x * y * k3 *(1-x)**alpha
equation_2 = k2 * (1 - x - y) ** 2 - k_2 * y ** 2 - x * y * k3 *(1-x)**alpha
solution = solve([equation_1, equation_2], y, k2)
y_sol = solution[0][0] #y(x)
k2_sol = solution[0][1] #k2(x)
a11 = -k_1 - k1 - k3*y*((1-x)**alpha- x*alpha*(1-x)**(alpha-1))
a12 = -k1- k3*x*(1-x)**alpha
```

```
a21 = -k2 - k3*y*((1-x)**alpha - x*alpha*(1-x)**(alpha-1))
a22 = -2*k_2*y - k2 - k3*x*(1-x)**alpha
#print(y_sol)
#eq1 = k1 * (1 - x - y)- 0.01* x - x * y * 10 *(1-x)**18
#eq2 = k2 * (1 - x - y) ** 2 - 10**(-9) * y ** 2 - x * y * 10 *(1-x)**18
#sol = solve([eq1, eq2],y,k2)
#ysol = sol[0][0]
#k2sol = sol[0][1]
print('k2sol',k2_sol)

#матрица Якоби
#matrix_A = Matrix([equation_1, equation_2])
matrix_A = Matrix([equation_1, equation_2])
matrix_variables = Matrix([x, y]) #матрица Якоби
jacobian_ = matrix_A.jacobian(matrix_variables) #Якобиан
det_A = jacobian_.det() #определитель матрицы на стационаре
trace_A = jacobian_.trace() #след матрицы
S_A = lambdify((x,y,k1,k_1,k2,k_2,k3,alpha),trace_A)
delta_A = lambdify((x,y,k1,k_1,k2,k_2,k3,alpha),det_A)

y_func = lambdify((x,k1,k_1,k3,alpha),y_sol)
k2_func = lambdify((x,y,k1,k_1,k_2,k3,alpha),k2_sol)
x_range=np.linspace(0.001,0.987,1000)
N = np.size(x_range)
k2_f = np.zeros(N)
y_f = np.zeros(N)

bifurcation_point = []

def parametres_portrait():
   sol_k2_trace = solve(trace_A.subs(y,y_sol),k2)[0] #выражаем k2 из следа = 0
   k1_sol_new = solve(sol_k2_trace - k2_sol,k1)[0] #приравниваем k2 из одного
уравнение к k2 из следа, получаем k1
   k2_sol_new = k2_sol.subs(k1,k1_sol_new)
   k1_ot_x_trace = lambdify((x, k_1, k_2, k3, alpha),k1_sol_new, 'numpy')
   k2_ot_x_trace = lambdify((x, k_1, k_2, k3, alpha), k2_sol_new, 'numpy')

plt.plot(k1_ot_x_trace(x_range,k_1_value,k_2_value,k3_value,alpha_value),k2_ot_x_
```

```
trace(x_range,k_1_value,k_2_value,k3_value,alpha_value),linestyle='--',
linewidth=1.5, label='neutral')
   plt.xlabel(r'$k_1$')
   plt.ylabel(r'$k_2$')
   plt.show()
   det_A_ = det_A.subs(y,y_sol)
   sols = solve(det_A_,k2) #выражаем k1 из определителя = 0
   sol_k2_det___ = sols[0]
   sol_k1_det = solve(sol_k2_det___ - k2_sol, k1)[0]
   sol_k2_det = k2_sol.subs(k1,sol_k1_det)
   sol_k1_det_func = lambdify((x,k_1,k_2,k3,alpha),sol_k1_det)
   sol_k2_det_func= lambdify((x,k_1,k_2,k3,alpha),sol_k2_det)

plt.plot(sol_k1_det_func(x_range,k_1_value,k_2_value,k3_value,alpha_value),sol_k2
_det_func(x_range,k_1_value,k_2_value,k3_value,alpha_value), linestyle='--',
linewidth=1.5, label='neutral')
   plt.show()

def analysis_k2(elem, k3a):
   global k3_value
   global alpha_value
   if k3a == 0:
       k3_value = elem
   if k3a == 1:
       alpha_value = elem
   fl = 0
   fl_delta = 0
   fl_di = 0
   for i in range(N):
       cur_y = y_func(x_range[i],k1_value,k_1_value,k3_value,alpha_value)
       cur_k2 =
k2_func(x_range[i],cur_y,k1_value,k_1_value,k_2_value,k3_value,alpha_value)
       k2_f[i] = cur_k2
       y_f[i] = cur_y
       sa = S_A(x_range[i], cur_y,
k1_value,k_1_value,k2_value,k_2_value,k3_value,alpha_value)
       deltaa =
delta_A(x_range[i],cur_y,k1_value,k_1_value,k2_value,k_2_value,k3_value,alpha_val
ue)
```

```
        di = sa*sa - 4*deltaa
        if sa < 0:
            if fl != 0 and fl == 1:
                #print(x_range[i], cur_y)
                bifurcation_point.append([x_range[i],cur_y])
                plt.plot(k2_f[i],x_range[i],'r', marker='o', label="hopf_x")
                plt.plot(k2_f[i],y_f[i],'r', marker='o', label="hopf_y")
            fl = -1
            #print ('меньше нуля',i, sa, cur_y, cur_k2, fl)
        else:
            if fl != 0 and fl == -1:
                #print(x_range[i], cur_y)
                bifurcation_point.append([x_range[i],cur_y])
                plt.plot(k2_f[i],x_range[i],'r', marker='o', label="hopf_x")
                plt.plot(k2_f[i],y_f[i],'r', marker='o', label="hopf_y")
            fl = 1
            #print('больше нуля',i, sa, cur_y, cur_k2, fl)

        if deltaa < 0:
            if fl_delta != 0 and fl_delta == 1:
                #print ('определитель меньше нуля',i, deltaa, cur_y, cur_k2, fl)
                plt.plot(k2_f[i],x_range[i],'k*', marker='o',
label="saddle-node_x")
                plt.plot(k2_f[i],y_f[i],'k*', marker='o', label="saddle-node_y")
            fl_delta = -1
        else:
            if fl_delta != 0 and fl_delta == -1:
                #print('определитель больше нуля',i, deltaa, cur_y, cur_k2, fl)
                plt.plot(k2_f[i],x_range[i],'k*', marker='o',
label="saddle-node_x")
                plt.plot(k2_f[i],y_f[i],'k*', marker='o', label="saddle-node_y")
            fl_delta = 1
    line1, = plt.plot(k2_f, x_range, 'b--', label="x")
    line2, = plt.plot(k2_f, y_f, 'k', label="y")

    plt.legend(handler_map={line1: HandlerLine2D(numpoints=4)})
    if k3a == 0:
        plt.title('Однопараметрический анализ. Завсимость стационарных решений от
параметра k2, k3 = ' + str(k3_value))
```

```
    else:
        plt.title('Однопараметрический анализ. Завсимость стационарных решений от
параметра k2, alpha = ' + str(alpha_value))
    plt.plot(k2_f,x_range, 'b--')
    plt.plot(k2_f,y_f,'k')
    plt.xlabel('k2')
    plt.ylabel('x,y')
    plt.show()


def solve_system(init, dt, iterations):

    func_1 = lambdify((x, y, k1, k_1, k3,alpha), equation_1)
    func_2 = lambdify((x, y, k2, k_2, k3,alpha), equation_2)

    def list_of_functions(xy,t):
        return
[func_1(xy[0],xy[1],k1_value,k_1_value,k3_value,alpha_value),func_2(xy[0],xy[1],k
2_value,k_2_value,k3_value,alpha_value)]

    t = np.arange(iterations) * dt
    return odeint(list_of_functions, init, t), t

def phase_portrait():
    func_1 = lambdify((x, y, k1, k_1, k3,alpha), equation_1)
    func_2 = lambdify((x, y, k2, k_2, k3,alpha), equation_2)
    Y, X = np.mgrid[0:.5:1000j, 0:1:2000j]
    U = func_1(X, Y, k1_value, k_1_value, k3_value,alpha_value)
    V = func_2(X, Y, k2_value, k_2_value, k3_value,alpha_value)
    velocity = np.sqrt(U*U + V*V)
    plt.streamplot(X, Y, U, V, density = [2.5, 0.8], color=velocity)
    plt.xlabel('x')
    plt.ylabel('y')
    plt.show()
    print('тут4')

#----------------------ОДНОПАРАМЕТРИЧЕСКИЙ АНАЛИЗ ПО
K2--------------------------------
#for elem in k3_range:
```

```
   #analysis_k2(elem,0)

#for elem in alpha_range:
#    analysis_k2(elem,1)
#----------------------Двухпараметрический АНАЛИЗ ПО
K1,K2--------------------------------
parametres_portrait()
phase_portrait()
#колебания
res, times = solve_system([0.1, 0.1], 1e-2, 1e4)
ax = plt.subplot(211)
plt.plot(times, res[:, 0])
plt.setp(ax.get_xticklabels(), visible=False)
plt.ylabel('x')
plt.grid()
ax1 = plt.subplot(212, sharex=ax)
plt.plot(times, res[:, 1], color='red')
plt.xlabel('t')
plt.ylabel('y')
plt.grid()
plt.show()
```