Московский государственный университет имени Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математической физики

Отчёт по программе

“Построение параметрического портрета системы. Нахождение областей множественности стационарных состояний и автоколебаний”

студента кафедры математической физики
Колодяжного А. В.

Москва, 2017

Содержание:

Условие задачи.

Рассмотреть автокаталитическую химическую реакцию, происходящую на поверхности катализатора. Математическая модель, описывающая изменение концентраций адсорбированных веществ [X] и [Y], x и y:

$$\frac{dx}{dt}=k_1 z-k_{-1}x-k_2 z^2 x,$$

$$\frac{dy}{dt}=k_3 z^2-k_{-3}y^2,$$

где $z=1-x-y$, $0\leqslant x\leqslant 1$, $0\leqslant y\leqslant 1$ и $0\leqslant x+y\leqslant 1$.

Базовый набор параметров : $k_1=0.12$, $k_{-1}=0.005$, $k_2=1.05$, $k_3=0.0032$ и $k_{-3}=0.002$.

1. Используя аналитические методы продолжения по параметру, построить зависимость стационарных решений x_c и y_c от параметра k_1 для нескольких значений k_{-1} : 0.001, 0.005, 0.01, 0.015 и 0.02.
2. Используя аналитические методы продолжения по параметру, построить зависимость стационарных решений x_c и y_c от параметра k_1 для нескольких значений k_{-3} : 0.0005, 0.001, 0.002, 0.003 и 0.004.
3. Исследуя след и определитель матрицы Якоби на стационаре, найти точки бифуркаций, уточнить их и отметить на графиках.
4. На плоскости параметров (k_{-1},k_1) построить параметрический портрет системы, провести линии кратности и нейтральности. Найти и отметить точки бифуркации коразмерности-2: C (трехкратный корень) и ТВ (Такенса−Богданова, когда два собственных значения равны нулю).
5. Задать параметры из области автоколебаний. С помощью стандартных программ численного интегрирования систем ОДУ в среде Python решить систему, задав некоторые начальные данные. Нарисовать графики установившихся колебаний x(t) и y(t). На фазовой плоскости построить фазовый портрет системы: отметить стационарную точку, нарисовать предельный цикл, нарисовать несколько траекторий, которые наматываются на цикл.

Однопараметрический и двухпараметрический анализ.

Стационарные состояния удовлетворяют системе уравнений:

$$\begin{cases} k_1 z-k_{-1}x-k_2 z^2 x=0 \\ k_3 z^2-k_{-3}y^2=0 \end{cases} \qquad (1)$$

Рассмотрим 2-ое уравнение:

$$(\sqrt{k_3}z-\sqrt{k_{-3}}y)(\sqrt{k_3}z+\sqrt{k_{-3}}y)=0$$

Так как выражение во второй скобке больше нуля, то её можно отбросить. Выразим y через x:

$$y=\frac{\sqrt{k_3}(1-x)}{\sqrt{k_3}+\sqrt{k_{-3}}}$$

Пусть

$$c=\frac{\sqrt{k_3}}{\sqrt{k_3}+\sqrt{k_{-3}}}$$

Теперь подставим y в первое уравнение системы (1):

$$k_1(1-x-c(1-x))-k_{-1}x-k_2(1-x-c(1-x))^2x=0$$

Выразим параметр k_1 :

$$k_1=\frac{k_{-1}x+k_2(1-x-c(1-x))^2x}{1-x-c(1-x)} \qquad (2)$$

Выпишем элементы матрицы Якоби:

$$A=\begin{pmatrix}\frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y}\\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y}\end{pmatrix}$$

$$a_{11}=-k_1-k_{-1}-k_2(1-x-y)^2+2k_2x(1-x-y)$$
$$a_{12}=-k_1+2k_2x(1-x-y)$$
$$a_{21}=-2k_3(1-x-y)$$
$$a_{22}=-2k_3(1-x-y)-2k_{-3}y$$

Вычислим определитель матрицы Якоби:

$$\Delta A=(k_1+k_{-1}+k_2(1-x-y)^2-2k_2x(1-x-y))(2k_3(1-x-y)+2k_{-3}y)$$
$$-(k_1-2k_2x(1-x-y))(2k_3(1-x-y))$$

Теперь вычислим её след:

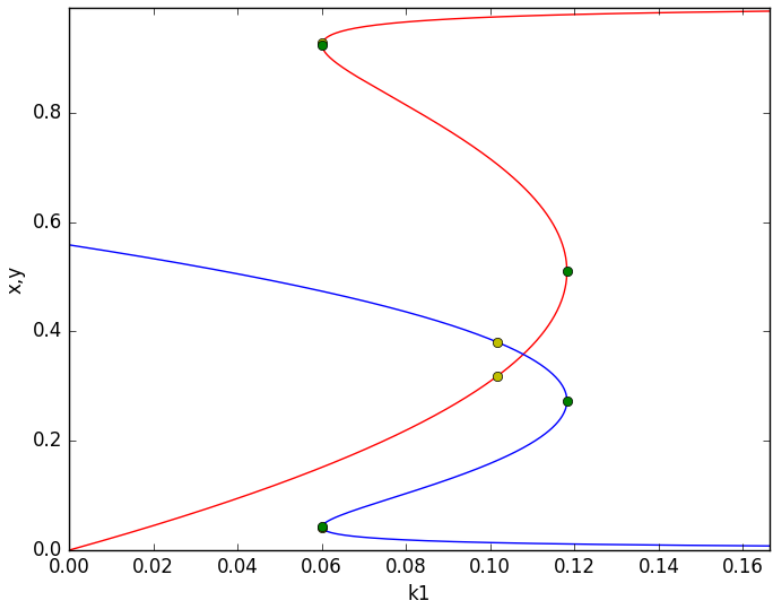
$$S_A(x,y)=-k_1-k_{-1}-k_2(1-x-y)^2+2k_2x(1-x-y)-2k_3(1-x-y)-2k_{-3}y$$

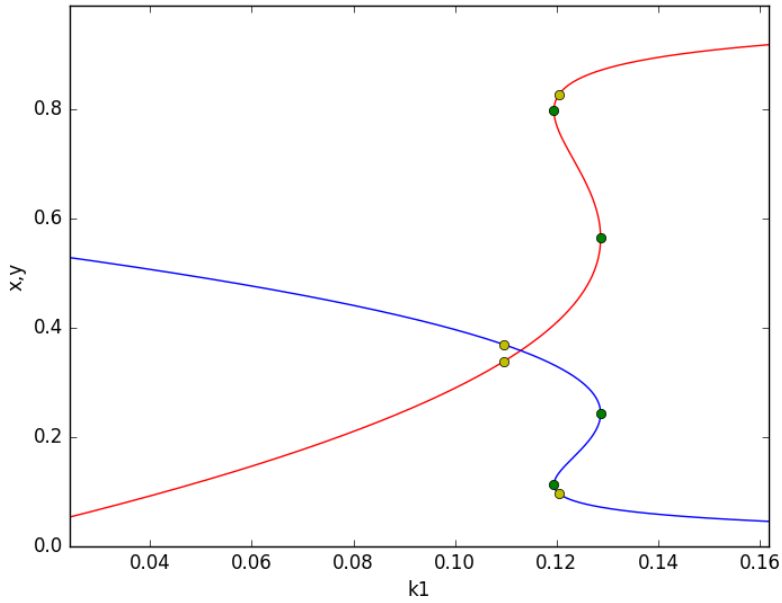
В точках смены знака определителя мы имеем седло-узловую бифуркацию, в точках смены знака следа – бифуркацию Хопфа. Точки бифуркации будем уточнять по формулам:

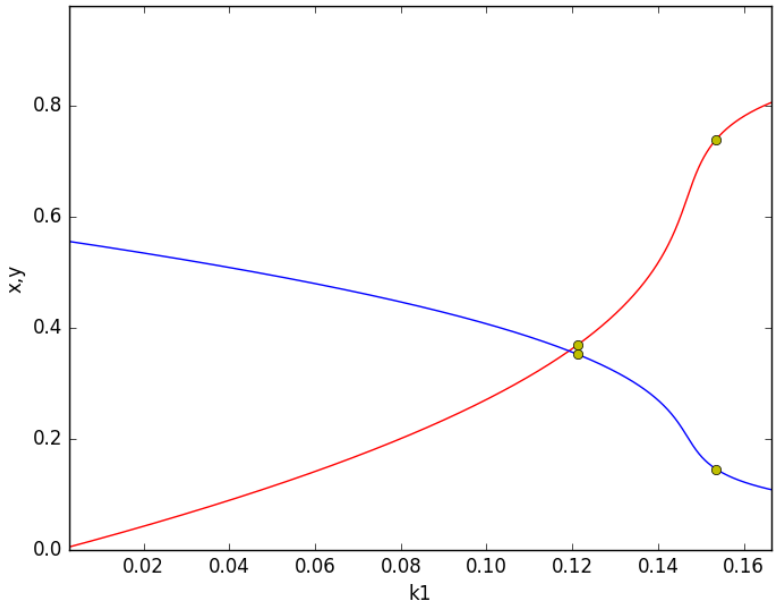
$$x=x_{i-1}-det_{i-1}(A)\frac{x_i-x_{i-1}}{det_i(A)-det_{i-1}(A)}$$

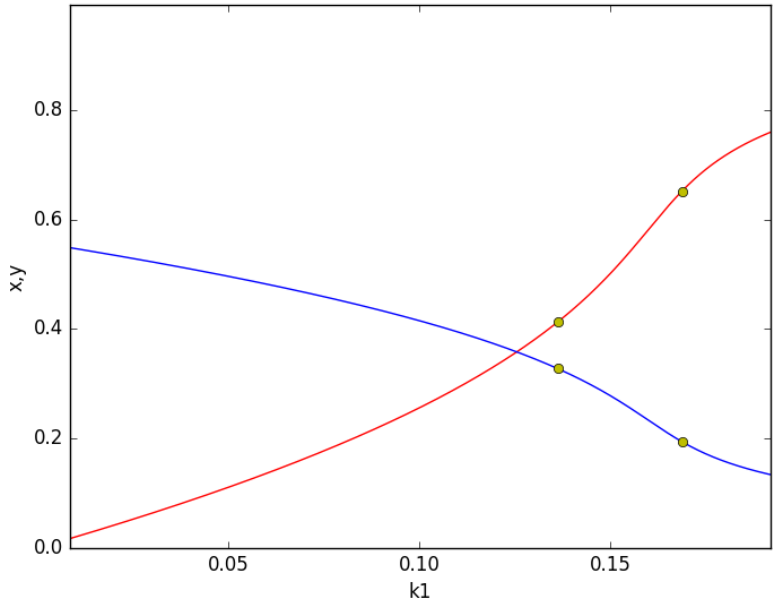
$$x=x_{i-1}-S_{i-1}(A)\frac{x_i-x_{i-1}}{S_i(A)-S_{i-1}(A)}$$

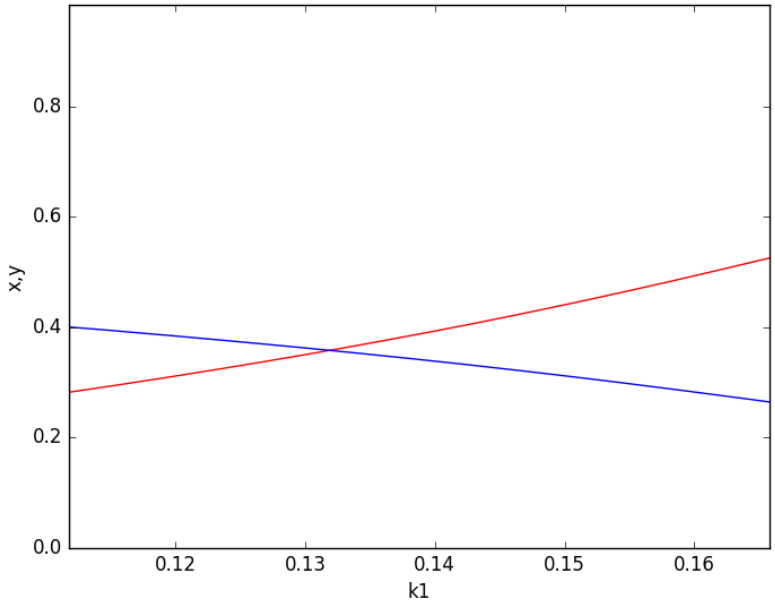
На рисунках 1-10 представлены графики x(k_1) и y(k_1) для нескольких значений параметров k_{-1} и k_{-3} соответственно(желтым помечены точки бифуркации Хопфа, зеленым – седло-узловой бифуркации).

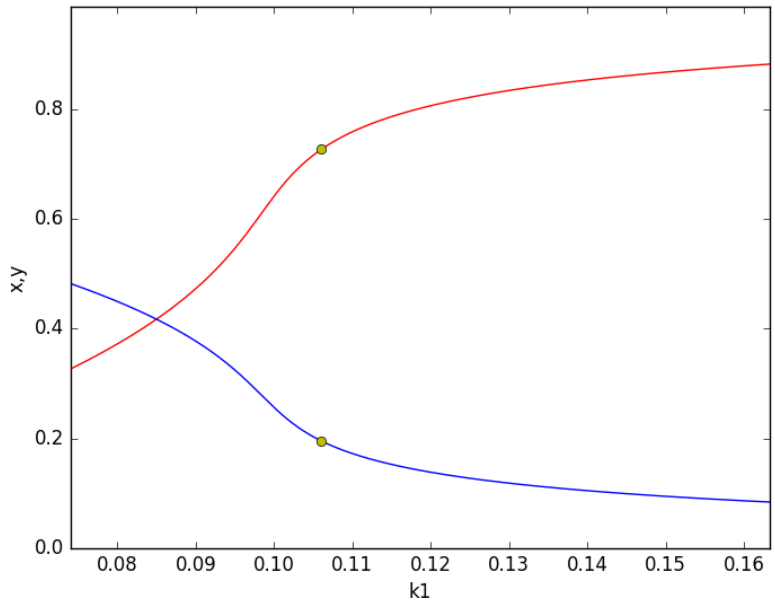


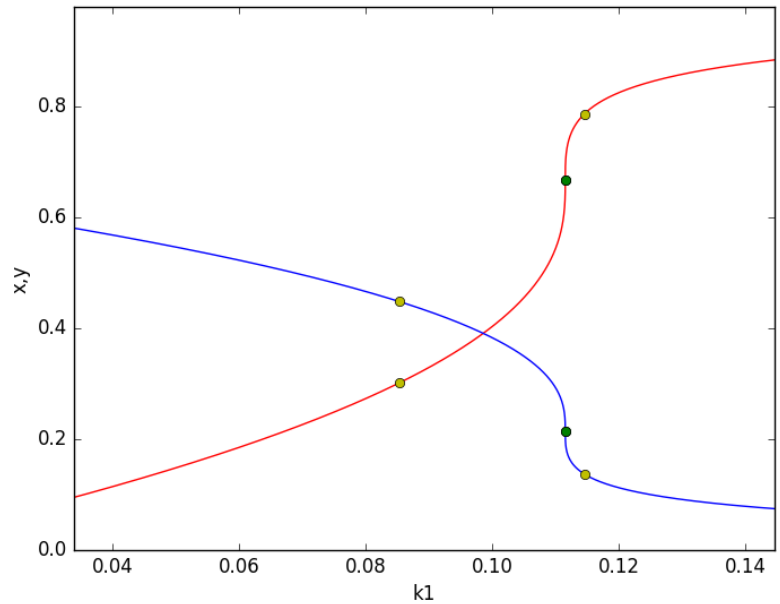

0.8
0.6
0.4
0.2
0.0
x,y
0.04
0.06
0.08
0.10
0.12
0.14
0.16
k1

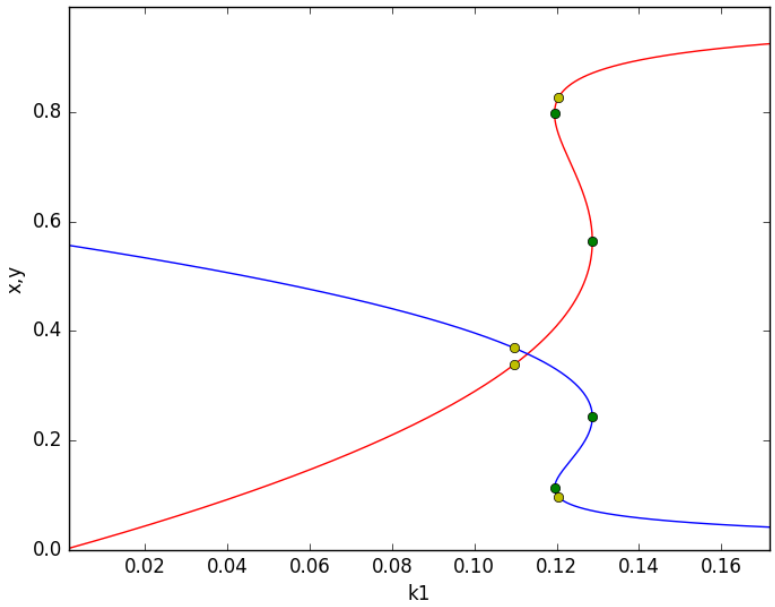

0.8
0.6
0.4
0.2
0.0
x,y
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
k1

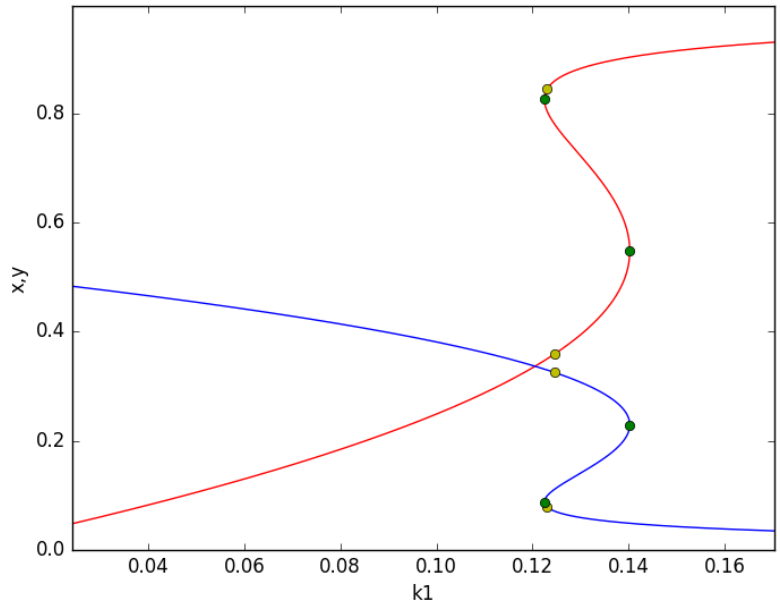

0.8
0.6
0.4
0.2
0.0
x,y
0.05
0.10
0.15
k1

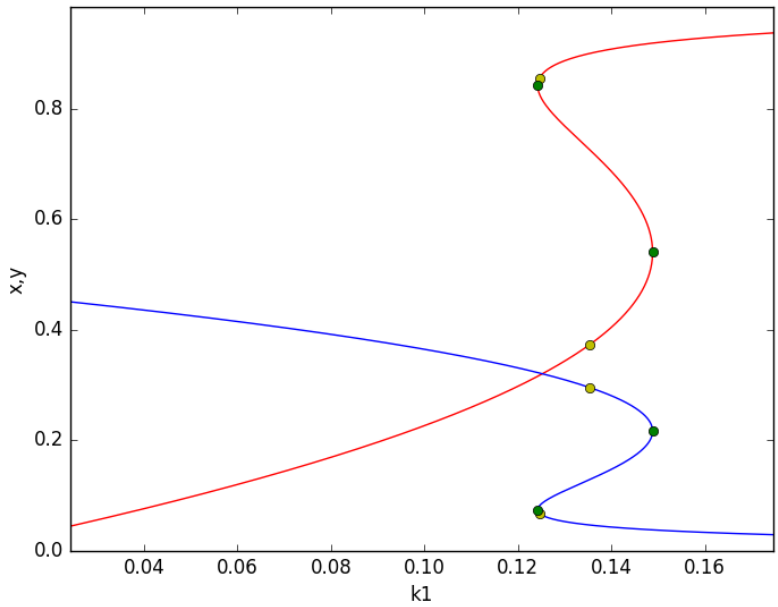

0.8
0.6
0.4
0.2
0.0
x,y
0.12
0.13
0.14
0.15
0.16
k1


0.8
0.6
0.4
0.2
0.0
x,y
0.08
0.09
0.10
0.11
0.12
0.13
0.14
0.15
0.16
k1


0.8
0.6
0.4
0.2
0.0
x,y
0.04
0.06
0.08
0.10
0.12
0.14
k1


0.8
0.6
0.4
0.2
0.0
x,y
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
k1


0.8
0.6
0.4
0.2
0.0
x,y
0.04
0.06
0.08
0.10
0.12
0.14
0.16
k1

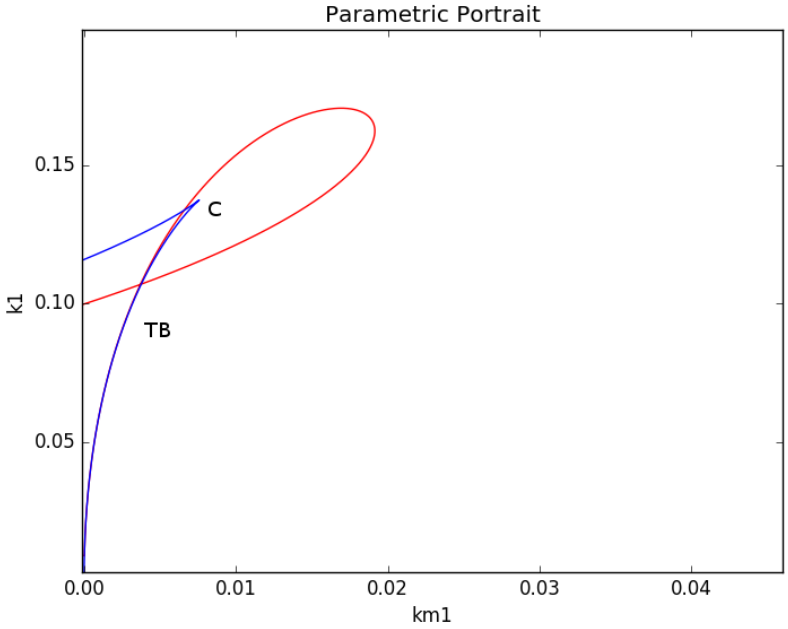


Допишем к системе (1) сначала уравнение $\Delta A=0$, потом $S_A=0$. Выразим k_{-1}:

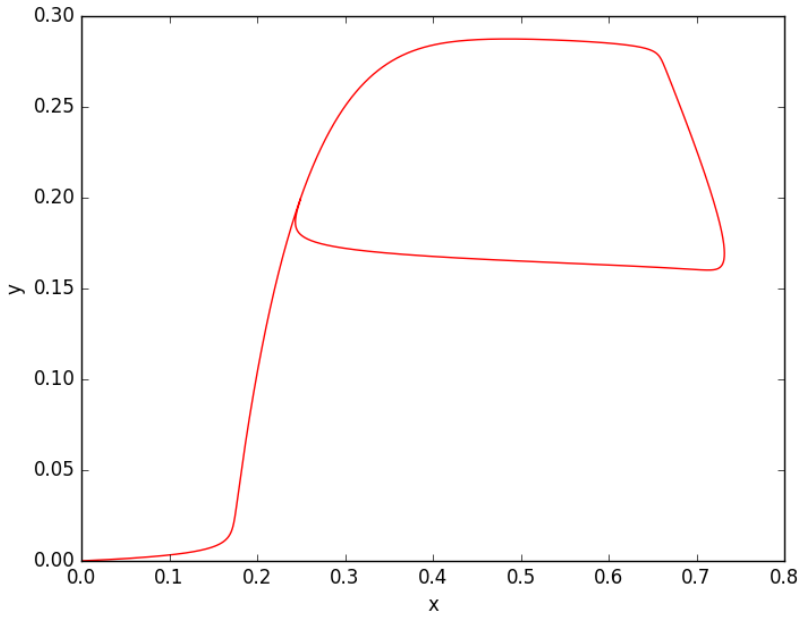
$$k_{-1}=\frac{(k_2z^2x+k_2z^3-2k_2xz^2)(2k_3z+2k_{-3}y)+2k_3k_2z^3x}{x2k_3z-(x+z)(2k_3z+2k_{-3}y)} \qquad для\ \Delta A=0$$

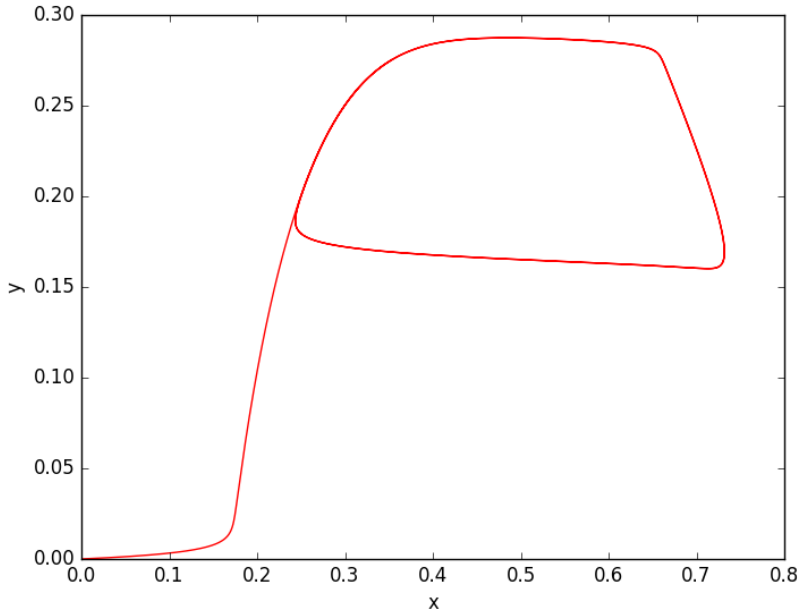
$$k_{-1}=\frac{-k_2z^2x-k_2z^3+2k_2xz^2-2k_3z^2-2k_{-3}yz}{x+z} \qquad для\ S_A=0$$

Возьмём k_1 из (2) и построим линии кратности(синяя) и нейтральности.

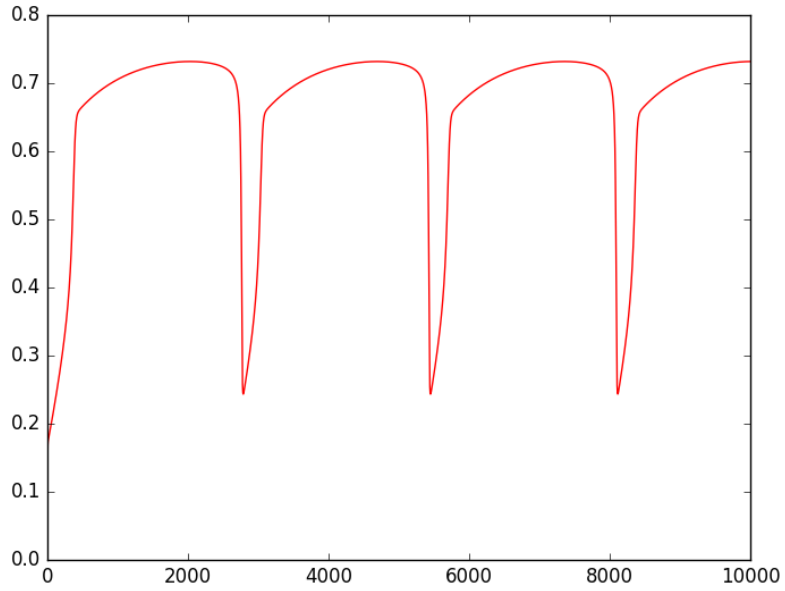


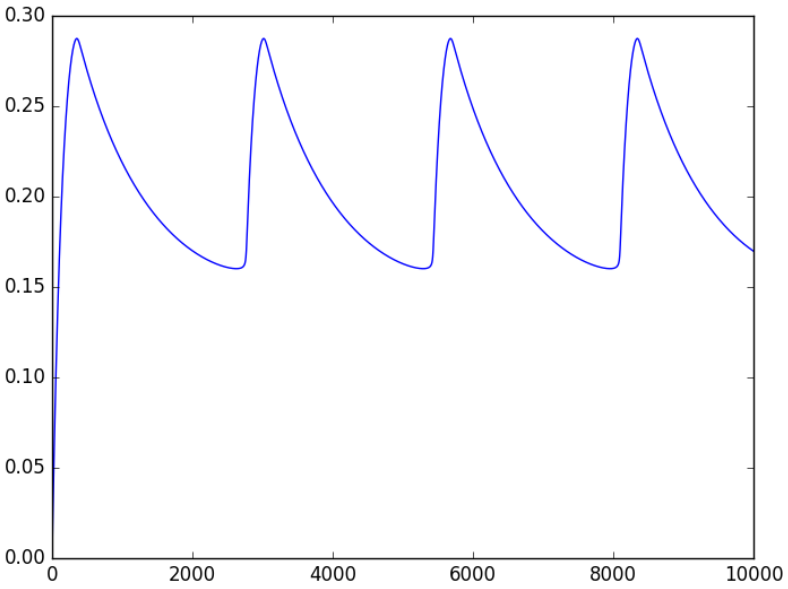
Теперь возьмём любую точку в области автоколебаний и решим ОДУ с помощью odeint. Построим графики y(x) для $t=\overline{0,2800}$ и $t=\overline{0,5600}$:


0.30
0.25
0.20
0.15
0.10
0.05
0.00
y
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
x


0.30
0.25
0.20
0.15
0.10
0.05
0.00
y
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
x

Теперь построим y(t) и x(t):





Код программы.

```
import numpy as np
import math
import matplotlib.pyplot as plt
import scipy.integrate
import numpy as nm
def getXcYc(x, k1m, k3, k3m, k2, n):
    k1 = [0.0] * n
    y = [0.0] * n
    z = [0.0] * n
    alk = math.sqrt(k3) / (math.sqrt(k3m) + math.sqrt(k3))
    for i in range(n):
        y[i] = (1 - x[i]) * alk
        z[i] = 1 - x[i] - y[i]
        k1[i] = (k1m*x[i]+k2*z[i]**2*x[i])/(z[i])
    i = 0
    DI = [0.0] * 1000
    yh = []
    xh = []
    k1h = []
    ysn = []
    xsn = []
    k1sn = []
    ydi = []
    xdi = []
    k1di = []
    k1[i] = (k1m * x[i] + k2 * z[i] ** 2 * x[i]) / (z[i])
    a11 = -k1[i] - k1m - k2 * z[i] ** 2 + 2 * k2 * x[i] * z[i]
    a12 = -k1[i] + 2 * k2 * x[i] * z[i]
    a21 = -2 * k3 * z[i]
    a22 = -2 * k3 * z[i] - 2 * k3m * y[i]
    sp[i] = a11 + a22
    det[i] = a11 * a22 - a12 * a21
    DI[i] = sp[i] ** 2 - 4 * det[i]
    for i in range(1, n):
        k1[i] = (k1m * x[i] + k2 * z[i] ** 2 * x[i]) / (z[i])
        a11 = -k1[i] - k1m - k2 * z[i] ** 2 + 2 * k2 * x[i] * z[i]
        a12 = -k1[i] + 2 * k2 * x[i] * z[i]
        a21 = -2 * k3 * z[i]
        a22 = -2 * k3 * z[i] - 2 * k3m * y[i]
        sp[i] = a11 + a22
        det[i] = a11 * a22 - a12 * a21
```

```
        DI[i] = sp[i] ** 2 - 4 * det[i]
        if sp[i] * sp[i - 1] <= 0:
            x1 = x[i-1] - sp[i-1]*(x[i]-x[i-1])/(sp[i]-sp[i-1])
            y1 = (1-x1)*alk
             k11 = (k1m * x1 + k2 * (1-x1-y1) ** 2 * x1) / (1-x1-
y1)
            yh.append(y1)
            xh.append(x1)
            k1h.append(k11)
        if det[i] * det[i - 1] <= 0:
               x1 = x[i - 1] - det[i - 1] * (x[i] - x[i - 1]) /
(det[i] - det[i - 1])
            y1 = (1 - x1) * alk
            k11 = (k1m * x1 + k2 * (1 - x1 - y1) ** 2 * x1) / (1 -
x1 - y1)
            ysn.append(y1)
            xsn.append(x1)
            k1sn.append(k11)
        if DI[i] * DI[i - 1] <= 0:
            x1 = x[i - 1] - DI[i - 1] * (x[i] - x[i - 1]) / (DI[i]
- DI[i - 1])
            y1 = (1 - x1) * alk
            k11 = (k1m * x1 + k2 * (1 - x1 - y1) ** 2 * x1) / (1 -
x1 - y1)
            ydi.append(y1)
            xdi.append(x1)
            k1di.append(k11)
    figure, axes = plt.subplots()
    plt.xlabel("k1")
    plt.ylabel("x,y")
    axes.plot(k1, x, 'r')
    axes.plot(k1, y, 'b')
    for i in range(len(xh)):
        axes.plot(k1h[i], xh[i], 'oy')
        axes.plot(k1h[i], yh[i], 'oy')
    for i in range(len(xsn)):
        axes.plot(k1sn[i], xsn[i], 'og')
        axes.plot(k1sn[i], ysn[i], 'og')
    plt.show()
def scipySolve(r, t, k1, k1m, k2, k3, k3m):
       return  [k1*(1  -  r[0]  -  r[1])  -  k1m*r[0]  -  k2*(1-r[0]-
r[1])**2*r[0],
```

```
            k3*(1-r[0]-r[1])**2 - k3m*r[1]**2]
if __name__ == "__main__":
    # двухпараметрический анализ на плоскости (k1,k1m)
    k2 = 1.05
    k3m = 0.002
    k3 = 0.0032
    alk = math.sqrt(k3) / (math.sqrt(k3m) + math.sqrt(k3))
    x = np.linspace(0, 0.999, 1000)
    n = len(x)
    y = [0.0]*1000
    z = [0.0]*1000
    k1 = [0.0]*1000
    k1m = [0.0]*1000
    sp = [0.0]*1000
    det = [0.0]*1000
    K1 = [0.0]*1000
    K1m = [0.0]*1000
    Sp = [0.0]*1000
    Det = [0.0]*1000
    #зависимость от k1 для неск. значений параметра k1m
    getXcYc(x, 0.001, k3, k3m, k2, n)
    getXcYc(x, 0.005, k3, k3m, k2, n)
    getXcYc(x, 0.01, k3, k3m, k2, n)
    getXcYc(x, 0.015, k3, k3m, k2, n)
    getXcYc(x, 0.02, k3, k3m, k2, n)
    # зависимость от k1 для неск. значений параметра k3m
    getXcYc(x, 0.005, k3, 0.0005, k2, n)
    getXcYc(x, 0.005, k3, 0.001, k2, n)
    getXcYc(x, 0.005, k3, 0.002, k2, n)
    getXcYc(x, 0.005, k3, 0.003, k2, n)
    getXcYc(x, 0.005, k3, 0.004, k2, n)
    # расчет линии нейтральности
    for i in range(n):
        y[i] = (1 - x[i]) * alk
        z[i] = 1 - x[i] - y[i]
        k1m[i] = (-k2 * z[i] ** 2 * x[i] - k2 * z[i] ** 3 + 2 * k2
* x[i] * z[i] ** 2 - 2 * k3 * z[
            i] ** 2 - 2 * k3m * y[i] * z[i]) / (x[i] + z[i])
        k1[i] = (k1m[i] * x[i] + k2 * z[i] ** 2 * x[i]) / (z[i])
         a11 = -k1[i] - k1m[i] - k2 * z[i] ** 2 + 2 * k2 * x[i] *
z[i]
        a12 = -k1[i] + 2 * k2 * x[i] * z[i]
```

```
        a21 = -2 * k3 * z[i]
        a22 = -2 * k3 * z[i] - 2 * k3m * y[i]
        det[i] = a11 * a22 - a12 * a21
        sp[i] = a11 + a22
        if det[i] <= 0:
            k1m[i] = -1
    # расчет линии кратности
    for i in range(n):
        K1m[i] = ((k2 * z[i] ** 2 * x[i] + k2 * z[i] ** 3 - 2 * k2
* x[i] * z[i]) * (
        2 * k3 * z[i] + 2 * k3m * y[i]) - (k2 * z[i] ** 2 * x[i] -
2 * k2 * x[i] * z[i] ** 2) * 2 * k3 * z[i]) / (
                    x[i] * 2 * k3 * z[i] - (x[i] + z[i]) * (2 * k3
* z[i] + 2 * k3m * y[i]))
        K1[i] = (K1m[i] * x[i] + k2 * z[i] ** 2 * x[i]) / (z[i])
         a11 = -K1[i] - K1m[i] - k2 * z[i] ** 2 + 2 * k2 * x[i] *
z[i]
        a12 = -K1[i] + 2 * k2 * x[i] * z[i]
        a21 = -2 * k3 * z[i]
        a22 = -2 * k3 * z[i] - 2 * k3m * y[i]
        Det[i] = a11 * a22 - a12 * a21
        Sp[i] = a11 + a22
    #построение фазового портрета
    figure, axes = plt.subplots()
    axes.set_title("Parametric Portrait")
    plt.xlabel("km1")
    plt.ylabel("k1")
    axes.plot(k1m, k1, 'r')
    axes.plot(K1m, K1, 'b')
    plt.show()
    #odeint
    r = [0, 0]
    t = nm.linspace(0, 10000, 1000000)
     result = scipy.integrate.odeint(scipySolve, r, t, args=(0.15,
0.01, k2, k3, k3m))
    x11 = []
    y11 = []
    for i in range(len(result)):
        x11.append(result[i][0])
        y11.append(result[i][1])
    figure, axes = plt.subplots()
    plt.xlabel("x")
```

```
plt.ylabel("y")
axes.plot(x11, y11, 'r')
plt.show()
figure, axes = plt.subplots()
axes.plot(t, x11, 'r')
axes.plot(t, y11, 'b')
plt.show()
```