Лабораторная работа №6

Построение и исследование компьютерных моделей с использованием дифференциальных уравнений

№1 Остывание кофе

Постановка задачи:

Природа переноса тепла от кофе к окружающему пространству сложна и включает в себя механизмы конвекции, излучения, испарения и теплопроводности. Исследовать зависимость остывания кофе в чашке при следующих исходных данных: $T_{\text{среды}} = 22℃$, $T_{\text{жидкости}} = 83℃$, коэффициент остывания $r = 0.0373$.

1. Аналитическое исследование модели:

$$\frac{dT}{dt} = -r(T - T_s), \text{где}$$

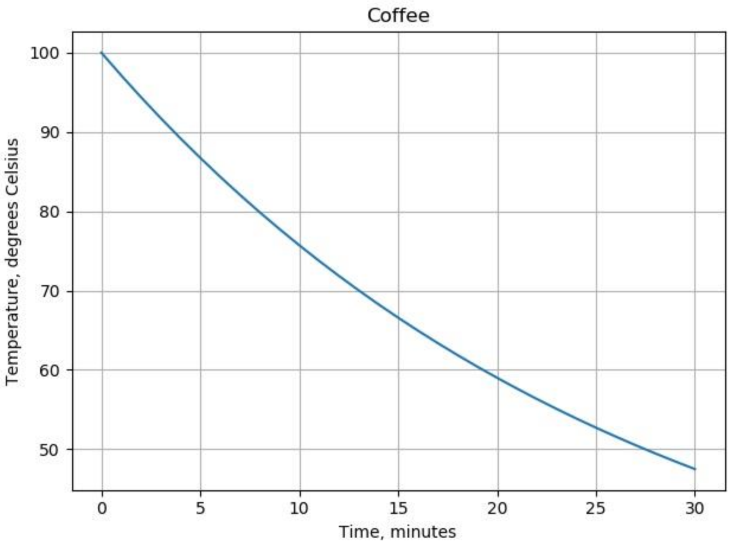
$T - \text{температура тела}, T_s - \text{температура окружающей среды},$
$r - \text{коэф. остывания}$

2. Программная реализация:

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

def f1(t, T):
    r = 0.0373
    Ts = 22
    return -r*(T - Ts)

def RungeKuttaMethod(f, a, b, n, y0):
    h = (b - a)/n
    x = a
    y = y0
    e = b - h
    h2 = h/2
    xTable = [a]
    yTable = [y]
    print('x =', x, ' y =', y)
    while (x <= e):
        k1 = h*f(x, y)
        k2 = h*f(x + h2, y + k1/2)
        k3 = h*f(x + h2, y + k2/2)
```

```
        k4 = h*f(x + h, y + k3)
        Fi = (k1 + 2*k2 + 2*k3 + k4)/6
        y += Fi
        x += h
        yTable += [y]
        xTable += [x]
        print('x =', x, ' y =', y)
    return (xTable, yTable)

def drawPlot(f, a, b, n, y0, textTitle, textXlbl, textYlbk):
    x, y = RungeKuttaMethod(f, a, b, n, y0)
    plt.plot(x, y)
    plt.grid(True)
    plt.title(textTitle)
    plt.xlabel(textXlbl)
    plt.ylabel(textYlbk)
    plt.show()

drawPlot(f1, 0, 30, 30, 100, 'Coffee', 'Time, minutes', 'Temperature, degrees
Celsius')
```

3. Вывод:
   Остывание кофе происходит по экспоненциальному закону.
   Для остывания кофе с 100℃ до 50℃ требуется больше 25 минут.
   Для остывания кофе с 83℃ до 50℃ требуется около 20 минут.



```
x = 0  y = 100
x = 1.0  y = 97.14419196446455
x = 2.0  y = 94.39294341015768
x = 3.0  y = 91.74242610878858
x = 4.0  y = 89.1889519946960 7
x = 5.0  y = 86.72896803308507
x = 6.0  y = 84.35905127615172
x = 7.0  y = 82.07590410021838
x = 8.0  y = 79.87634961725094
x = 9.0  y = 77.75732725437396
x = 10.0  y = 75.71588849523268
x = 11.0  y = 73.74919277727612
x = 12.0  y = 71.85450353925282
x = 13.0  y = 70.02918441341912
x = 14.0  y = 68.27069555716199
x = 15.0  y = 66.57659011893176
x = 16.0  y = 64.94451083356746
x = 17.0  y = 63.372186742277194
x = 18.0  y = 61.85743003270965
x = 19.0  y = 60.39813299471986
x = 20.0  y = 58.992265087593196
x = 21.0  y = 57.637870114646965
x = 22.0  y = 56.333063501278026
x = 23.0  y = 55.07602967266907
x = 24.0  y = 53.86501952750475
x = 25.0  y = 52.69834800418242
x = 26.0  y = 51.57439173613104
x = 27.0  y = 50.49158679297571
x = 28.0  y = 49.44842650440474
x = 29.0  y = 48.44345936371137
x = 30.0  y = 47.475287008092884
```

№2 Задача о распаде радия

Постановка задачи:

Установлено, что скорость распада радия прямо пропорциональна его количеству в каждый данный момент. Определить закон изменения массы радия в зависимости от времени, если при $t = 0$, масса радия была $m0$, $k = 0{,}00044$. Найти период полураспада радия.

1. Аналитическое исследование модели:
   Скорость распада:
   В момент t масса равна m, в момент $t + \Delta t$ масса равна $m + \Delta m$ => за Δt распалась масса Δm.
   Тогда $\frac{\Delta m}{\Delta t}$ – средняя скорость распада.
   Значит, $\lim_{\Delta t \to 0} \frac{\Delta m}{\Delta t} = \frac{dm}{dt}$ – скорость распада радия в момент t.
   По условию $\frac{dm}{dt} = -km,$ где k – коэф. пропорциональности.

$$\frac{dm}{dt} = -km$$
$$\frac{dm}{m} = -kdt$$
$$\int \frac{dm}{m} = \int -kdt$$
$$\ln m = -kt + \ln C$$
$$\ln \frac{m}{C} = -kt$$
$$e^{\ln \frac{m}{C}} = e^{-kt}$$

$$\frac{m}{C} = e^{-kt}$$
$$m = Ce^{-kt}$$

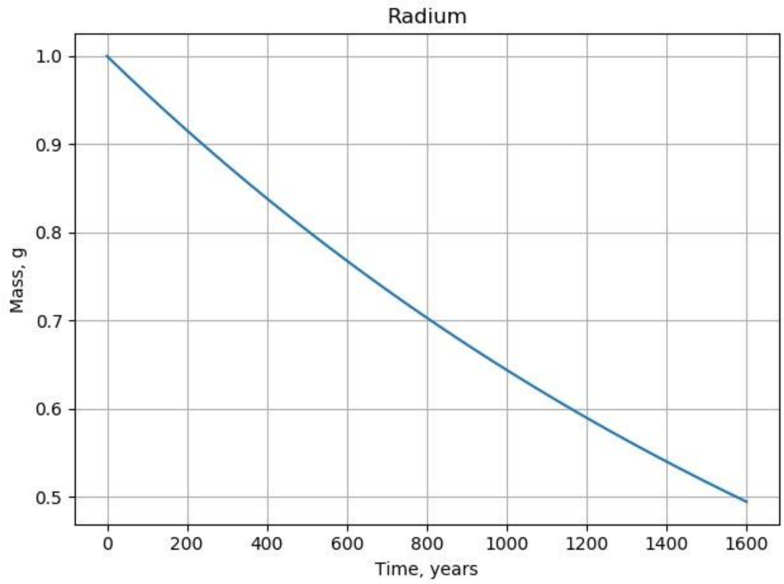
Т. к. при $t = 0$ масса m_0, то $m_0 = Ce^{-k*0} = C$

$$m = m_0 e^{-kt}$$
$$\frac{m_0}{2} = m_0 e^{-0.00044T}$$
$$T = \frac{\ln 2}{0.00044} \sim 1575 \text{ лет}$$

2. Программная реализация:

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

def f2(t, m):
    k = 0.00044
    return -k*m

def RungeKuttaMethod(f, a, b, n, y0):
    h = (b - a)/n
    x = a
    y = y0
    e = b - h
    h2 = h/2
    xTable = [a]
    yTable = [y]
    print('x =', x, ' y =', y)
    while (x <= e):
        k1 = h*f(x, y)
        k2 = h*f(x + h2, y + k1/2)
        k3 = h*f(x + h2, y + k2/2)
        k4 = h*f(x + h, y + k3)
        Fi = (k1 + 2*k2 + 2*k3 + k4)/6
        y += Fi
        x += h
        yTable += [y]
        xTable += [x]
        print('x =', x, ' y =', y)
    return (xTable, yTable)

def drawPlot(f, a, b, n, y0, textTitle, textXlbl, textYlbk):
    x, y = RungeKuttaMethod(f, a, b, n, y0)
    plt.plot(x, y)
    plt.grid(True)
    plt.title(textTitle)
    plt.xlabel(textXlbl)
    plt.ylabel(textYlbk)
    plt.show()

drawPlot(f2, 0, 1600, 160, 1, 'Radium', 'Time, years', 'Mass, g')
```

3. Вывод: для вычисления была выбрана масса $m = 1$ гр. Аналитически получено, что период полураспада радия равен 1575 годам, программно получено, что период полураспада равен 1570 годам.




```
x = 1470.0   y = 0.5237189989754382
x = 1480.0   y = 0.5214196975526221
x = 1490.0   y = 0.5191304908314366
x = 1500.0   y = 0.5168513344927682
x = 1510.0   y = 0.5145821844120789
x = 1520.0   y = 0.5123229966585523
x = 1530.0   y = 0.510073727494243
x = 1540.0   y = 0.5078343333732296
x = 1550.0   y = 0.505604770940772
x = 1560.0   y = 0.503384997032472
x = 1570.0   y = 0.5011749686734371
x = 1580.0   y = 0.49897464307744954
x = 1590.0   y = 0.49678397764613696
x = 1600.0   y = 0.49460292996814814
```

№3 Скорость ветра

Постановка задачи:

Проходя через лес и испытывая сопротивление деревьев, ветер теряет часть своей скорости. На бесконечно малом пути эта потеря пропорциональна скорости в начале этого пути и длине его. Найти скорость ветра, прошедшего в лесу 150 м, зная, что до вступления в лес начальная скорость ветра $v_0 = 12$ м/с; после прохождения в лесу пути $s = 1$ м, скорость ветра уменьшилась до величины $v_1 = 11{,}8$ м/с.

1. Аналитическое исследование модели:

На расстоянии S от начала леса скорость v, на расстоянии $S + \Delta S$ от начала леса скорость $v + \Delta v$ => за ΔS изменилась скорость на Δv.
Тогда $\frac{\Delta V}{\Delta S}$ – средняя движения ветра.
Значит, $\lim\limits_{\Delta S \to 0} \frac{\Delta v}{\Delta S} = \frac{dv}{dS}$ – скорость ветра на расстоянии S.
По условию $\frac{dm}{dt} = -km,$ где k – коэф. пропорциональности.

$$\frac{dv}{dS} = -kv$$
$$\frac{dv}{v} = -kdS$$
$$\int \frac{dv}{v} = \int -kdS$$
$$\ln v = -kS + \ln C$$
$$\ln \frac{v}{C} = -kS$$
$$e^{\ln \frac{v}{C}} = e^{-kS}$$
$$\frac{v}{C} = e^{-kS}$$

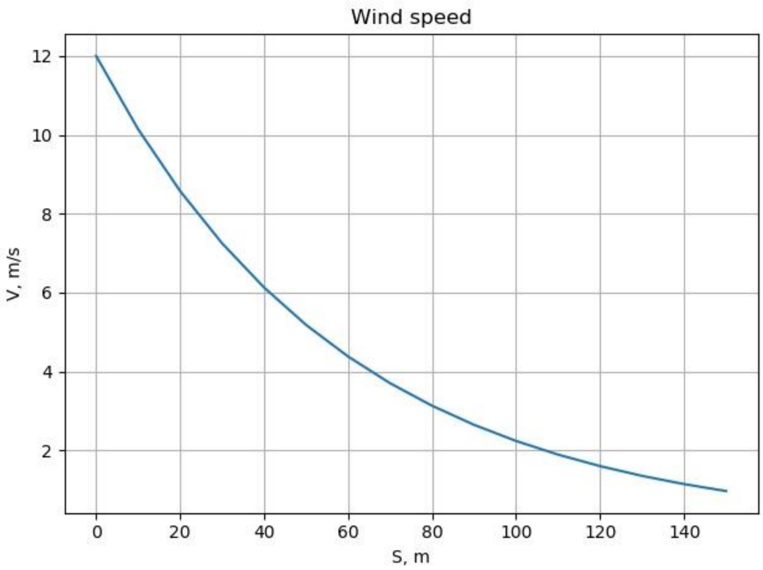
$$v = Ce^{-kS}$$
$$\text{Т. к. при } S = 0 \; v = v_0,$$
$$\text{то } v_0 = Ce^{-k*0} = C$$
$$v = v_0 e^{-kS}$$
$$\text{При } S = 1 \text{ м}, v_1 = 11{,}8 \frac{\text{м}}{\text{с}} = v$$
$$v_1 = v_0 e^{-k}$$
$$e^{-k} = \frac{v_1}{v_0} = \frac{11{,}8}{12} \sim 0{,}983$$
$$v = 12 * (0{,}983)^{150} \sim 0{,}92 \text{ м/с}$$

2. Программная реализация:

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

def f3(t, S):
    k = -np.log(11.8/12)
    return -k*S

def RungeKuttaMethod(f, a, b, n, y0):
    h = (b - a)/n
    x = a
    y = y0
    e = b - h
    h2 = h/2
    xTable = [a]
    yTable = [y]
    print('x =', x, ' y =', y)
    while (x <= e):
        k1 = h*f(x, y)
        k2 = h*f(x + h2, y + k1/2)
        k3 = h*f(x + h2, y + k2/2)
        k4 = h*f(x + h, y + k3)
        Fi = (k1 + 2*k2 + 2*k3 + k4)/6
```

```
            y += Fi
            x += h
            yTable += [y]
            xTable += [x]
            print('x =', x, ' y =', y)
    return (xTable, yTable)

def drawPlot(f, a, b, n, y0, textTitle, textXlbl, textYlbk):
    x, y = RungeKuttaMethod(f, a, b, n, y0)
    plt.plot(x, y)
    plt.grid(True)
    plt.title(textTitle)
    plt.xlabel(textXlbl)
    plt.ylabel(textYlbk)
    plt.show()

drawPlot(f3, 0, 150, 15, 12, 'Wind speed', 'S, m', 'V, m/s')
```

3. Вывод: аналитически получено, что скорость ветра, прошедшего через лес 150 м, равна 0,92 м/с, программно получено, что скорость равна 0,96 м/с.



```
x = 0  y = 12
x = 10.0  y = 10.14353698671066
x = 20.0  y = 8.574278550063932
x = 30.0  y = 7.247792633911112
x = 40.0  y = 6.126521054507203
x = 50.0  y = 5.178716076354617
x = 60.0  y = 4.377541505348014
x = 70.0  y = 3.7003128475298865
x = 80.0  y = 3.1278550192766708
x = 90.0  y = 2.643959423091791
x = 100.0  y = 2.23492501662448
x = 110.0  y = 1.8891703807212792
x = 120.0  y = 1.5969058025870462
x = 130.0  y = 1.3498560894028813
x = 140.0  y = 1.1410262641328948
x = 150.0  y = 0.9645035094200254
```
я

№4 Изменение тока

Постановка задачи:

В цепи поддерживается напряжение $E = 300$ В. Сопротивление цепи $R = 150$ Ом. Коэффициент самоиндукции равен $L = 30$ Гн. За какое время с момента замыкания цепи возникающий в ней ток I достигнет 99% своей предельной величины.

1. Аналитическое исследование модели:
   Характер изменения тока при размыкании цепи:
   $$I_0 = \frac{E}{R} - \text{постоянный ток.}$$
   Отключение источника тока в момент $t = 0$:
   (при убывании силы тока, возникает ЭДС самоиндукции, противодействующая этому убыванию)
   $$IR = E_s = -L\frac{dI}{dt} \text{ или } \frac{dI}{dt} + \frac{R}{L}I = 0$$
   $$\frac{dI}{I} = -\frac{R}{L}dt$$
   $$\ln I = -\frac{R}{L}t + \ln C$$
   $$I = Ce^{-\frac{R}{L}t}$$
   $$\text{При } t = 0\ I_0 = \frac{E}{R} => I_0 = C$$
   Замыкание цепи:

$$IR = E + E_s = E - L\frac{dI}{dt}$$

Общее решение:

$$I = I_0 + I_0 e^{-\frac{R}{L}t}$$

$$I_0\frac{99}{100} = I_0(1 - e^{-5t})$$

$$\frac{1}{100} = e^{-5t}$$

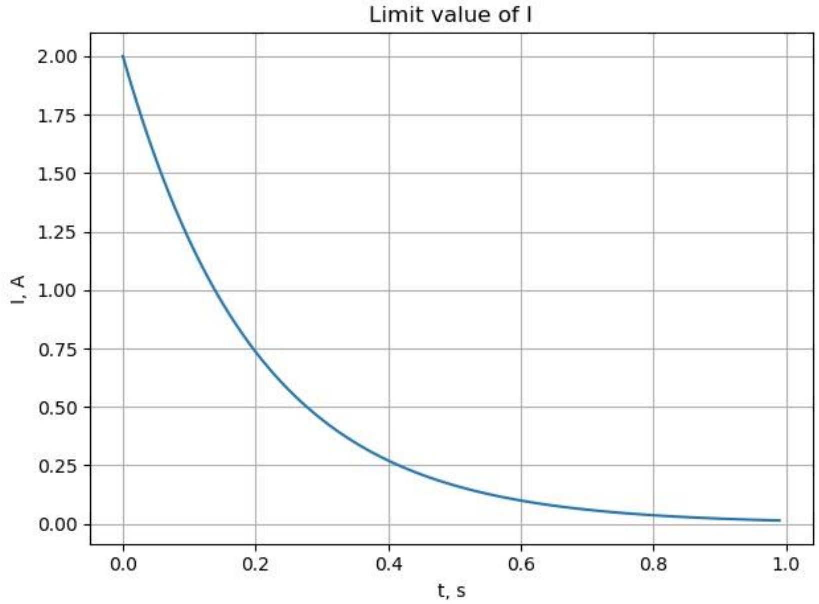
$$t = \frac{\ln 100}{5} \sim 0.92\ с$$

2. Программная реализация:

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

def f4(t, I):
    R = 150
    L = 30
    E = 300
    return -R*I/L

def RungeKuttaMethod(f, a, b, n, y0):
    h = (b - a)/n
    x = a
    y = y0
    e = b - h
    h2 = h/2
    xTable = [a]
    yTable = [y]
    print('x =', x, ' y =', y)
    while (x <= e):
        k1 = h*f(x, y)
        k2 = h*f(x + h2, y + k1/2)
        k3 = h*f(x + h2, y + k2/2)
        k4 = h*f(x + h, y + k3)
        Fi = (k1 + 2*k2 + 2*k3 + k4)/6
        y += Fi
        x += h
        yTable += [y]
        xTable += [x]
        print('x =', x, ' y =', y)
    return (xTable, yTable)

def drawPlot(f, a, b, n, y0, textTitle, textXlbl, textYlbk):
    x, y = RungeKuttaMethod(f, a, b, n, y0)
```

```
plt.plot(x, y)
plt.grid(True)
plt.title(textTitle)
plt.xlabel(textXlbl)
plt.ylabel(textYlbk)
plt.show()

drawPlot(f4, 0, 1, 100, 2, 'Limit value of I', 't, s', 'I, A')
```

3. Вывод: аналитически получилось, что ток I достигнет 99% своей предельной величины через 0.92 с, программно результат равен 0.99 с.




```
x = 0.8100000000000005  y = 0.034844756941964185
x = 0.8200000000000005  y = 0.033145358182762596
x = 0.8300000000000005  y = 0.03152884007466114
x = 0.8400000000000005  y = 0.029991160480821958
x = 0.8500000000000005  y = 0.02852847440173658
x = 0.8600000000000005  y = 0.027137124360725427
x = 0.8700000000000006  y = 0.025813631258342018
x = 0.8800000000000006  y = 0.024554685672813104
x = 0.8900000000000006  y = 0.023357139584761342
x = 0.9000000000000006  y = 0.022217998505517977
x = 0.9100000000000006  y = 0.02113441398934222
x = 0.9200000000000006  y = 0.020103676510823985
x = 0.9300000000000006  y = 0.019123208689659766
x = 0.9400000000000006  y = 0.018190558845860082
x = 0.9500000000000006  y = 0.017303394869273146
x = 0.9600000000000006  y = 0.016459498388095385
x = 0.9700000000000006  y = 0.015656759221787022
x = 0.9800000000000006  y = 0.014893170104522164
x = 0.9900000000000007  y = 0.014166821665979245
```

№5 Остывание хлеба

Постановка задачи:

Исследовать зависимость остывания хлеба при следующих исходных данных : $T_{\text{среды}} = 25℃$, $T_{\text{хлеба}} = 100℃$. Известно, что хлеб остыл до температуры $T_{\text{хлеба}} = 60℃$ за $t = 20$ минут.

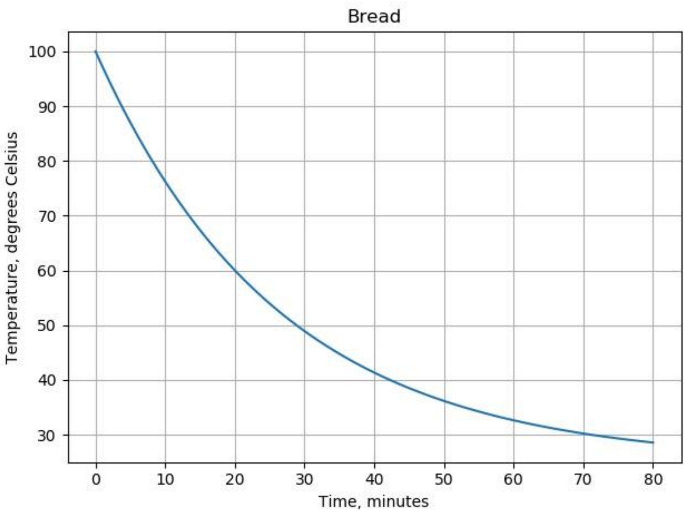
1. Аналитическое исследование модели:

2. Программная реализация:

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

def f12(t, T):
    r = -np.log(7/15)/20
    Ts = 25
    return -r*(T - Ts)

def RungeKuttaMethod(f, a, b, n, y0):
    h = (b - a)/n
    x = a
    y = y0
    e = b - h
    h2 = h/2
    xTable = [a]
    yTable = [y]
    print('x =', x, ' y =', y)
    while (x <= e):
        k1 = h*f(x, y)
        k2 = h*f(x + h2, y + k1/2)
        k3 = h*f(x + h2, y + k2/2)
        k4 = h*f(x + h, y + k3)
        Fi = (k1 + 2*k2 + 2*k3 + k4)/6
        y += Fi
        x += h
        yTable += [y]
        xTable += [x]
        print('x =', x, ' y =', y)
    return (xTable, yTable)

def drawPlot(f, a, b, n, y0, textTitle, textXlbl, textYlbk):
    x, y = RungeKuttaMethod(f, a, b, n, y0)
    plt.plot(x, y)
```

```
    plt.grid(True)
    plt.title(textTitle)
    plt.xlabel(textXlbl)
    plt.ylabel(textYlbk)
    plt.show()

drawPlot(f12, 0, 80, 80, 100, 'Bread', 'Time, minutes', 'Temperature, degrees Celsius')
```

3. Вывод:
   Остывание хлеба происходит по экспоненциальному закону.
   Для остывания хлеба с 100℃ до 30℃ требуется примерно 71 минута.



```
x = 58.0  y = 33.22585302642548
x = 59.0  y = 32.91828784114584
x = 60.0  y = 32.62222253835768
x = 61.0  y = 32.33722713669885
x = 62.0  y = 32.06288773183858
x = 63.0  y = 31.79880589535624
x = 64.0  y = 31.544598096096028
x = 65.0  y = 31.299895143157265
x = 66.0  y = 31.064341649711313
x = 67.0  y = 30.83759551686641
x = 68.0  y = 30.619327436830847
x = 69.0  y = 30.409220414652882
x = 70.0  y = 30.206969307842858
x = 71.0  y = 30.012280383208857
x = 72.0  y = 29.824870890262318
x = 73.0  y = 29.644468650574023
x = 74.0  y = 29.470811662484117
x = 75.0  y = 29.30364772059202
x = 76.0  y = 29.14273404947366
x = 77.0  y = 28.987836951094025
x = 78.0  y = 28.83873146540299
x = 79.0  y = 28.69520104362149
x = 80.0  y = 28.557037233743547
```