

Компьютерные технологии

Карусевич А. А.

22 июня 2021 г.

Задание 11 Создайте модель процесса остывания стеклянного стакана с горячим кофе при комнатных условиях. Постройте графики изменения температуры с учетом теплопроводности, конвекции и испарения, а также оцените точность интегрирования в зависимости от схемы интегрирования и величины шага интегрирования.

1 Алгоритм решения

Для моделирования процесса остывания стакана с кофе будет необходимо учесть несколько эффектов. Во-первых, теплопроводность.

Учет теплопроводности Теплопроводность описывается уравнением теплопроводности (Heat Equation), и в общем виде выглядит следующим образом:

$$\frac{\partial u}{\partial t} = a^2 \Delta u + f(r, t),$$

где u - температура в пространстве и времени a^2 — коэффициентом температуропроводности, Δ — оператор Лапласа и $f(r, t)$ — функция тепловых источников. В нашем случае кофе наливается горячим, а после остывает при комнатных температурах, где источников нету, поэтому $f = 0$.

Чтобы описать систему кофе-стакан-воздух, мы введем зависимость коэффициента температуропроводности a^2 от координат. Соответственно в тех местах, где расположено кофе, будет значение температуропроводности кофе, где стекло - там температуропроводность стекла и т.д.

Рассматривать задачу будем в двух измерениях X, Y , поскольку при использовании цилиндрической системы координат от угла ничего не будет зависеть. В двумерной декартовой системе координат оператор Лапласа запишется как

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

Для численного расчета Лапласиана будем применять метод конечных разностей, например для частной производной по x :

$$\frac{\partial^2 u(x_i, t)}{\partial x^2} = \frac{u(x_{i+1}, t) - 2u(x_i, t) + u(x_{i-1}, t))}{\Delta x^2},$$

где x_i - принадлежит предварительно распределенной координатной сетке с заданным шагом.

Учет конвекции Чтобы учесть конвекционную передачу тепла от жидкости стенке, воспользуемся законом Ньютона — Рихмана, описывающего теплопередачу от жидкости телу. Запишем в следующей форме:

$$\frac{\partial u}{\partial n} = \frac{\alpha}{\lambda}(u_s - u),$$

где $\frac{\partial u}{\partial n}$ - производная по нормали к поверхности тела на границе тело-жидкость, т.е. стенки стакана, u_s - температура поверхности, α - коэффициент теплоотдачи, λ - коэффициент теплопроводности.

В таком виде данный закон выступает граничным условием третьего рода для уравнения теплопроводности, описанного выше.

Учет испарения В общем случае учет испарения является достаточно трудоемкой задачей. В данной работе охлаждение жидкости засчет испарения будет учитываться в зависимости от скорости испарения массы жидкости, а также удельной теплоты испарения, откуда будет находится изменение температуры.

Скорость испарения W (кг/ч) описывается как

$$W = S(b + 0.0174 \cdot V)P_V(T)(1 - \frac{h}{100}),$$

где S - площадь испаряемой жидкости (м^2), b - фактор скорости подвижности окружающего воздуха, V - скорость воздуха на поверхности испаряемой жидкости, $P_V(T)$ - давление насыщенного пара, зависящее от температуры, h - влажность воздуха в процентах.

Зная массу испаряемой жидкости, а также удельную теплоту испарения $L \simeq 2260$ кДж/кг, можно найти теплоту, потраченную жидкостью на испарение, а значит и падение температуры в верхнем слое.

2 Результаты моделирования

Моделирование системы производится на языке Python. Используется библиотека SciPy, а также встроенный для решения дифференциальных уравнений метод `integrate.solve_ivp`. В данном методе используется алгоритм Рунге-Кутты 5-го порядка, а шаг интегрирования выбирается так, чтобы ошибка не превышала наперед заданного значения относительной $\varepsilon_r \leq 10^{-3}$ и абсолютной ошибок $\varepsilon_a \leq 10^{-6}$ интегрирования. Исходный код программы приведен в листинге 1.

Результаты моделирования приведены на рисунках 1, 2, 3, 4, 5, 6.

Кофе имеет изначальную температуру 350 К, а стекло и воздух вокруг 300 К. Изначальное распределение температуры приведено на рис. 1. На рисунках 2, 3, 4, 5 показаны распределения температуры в разные моменты времени - 10, 30, 120 секунд с начала моделирования

На рис. 6 приведена зависимость температуры верхней части кофе от времени.

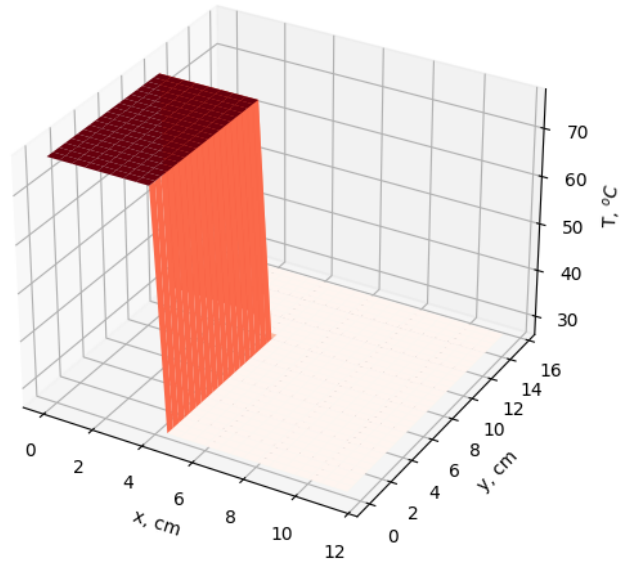


Рис. 1: Начальное распределение температуры. До $x = 4$ см расположено кофе, далее до $x = 5$ см стеклянный стакан. Далее пространство заполнено воздухом

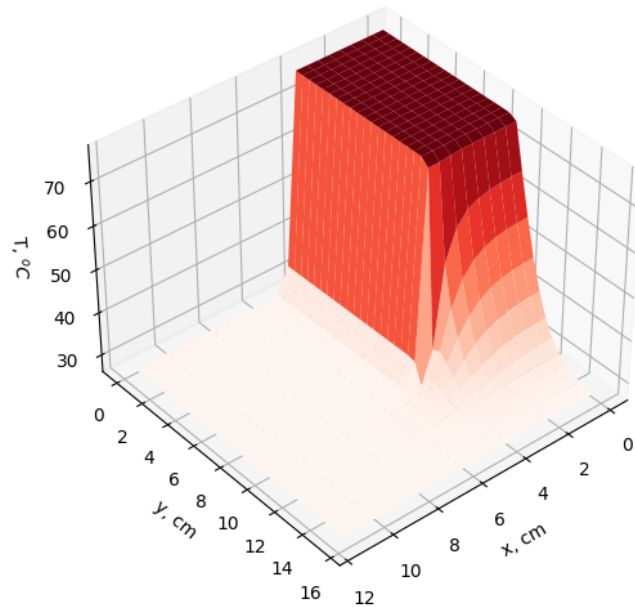


Рис. 2: Распределение температуры в $t = 10$ с

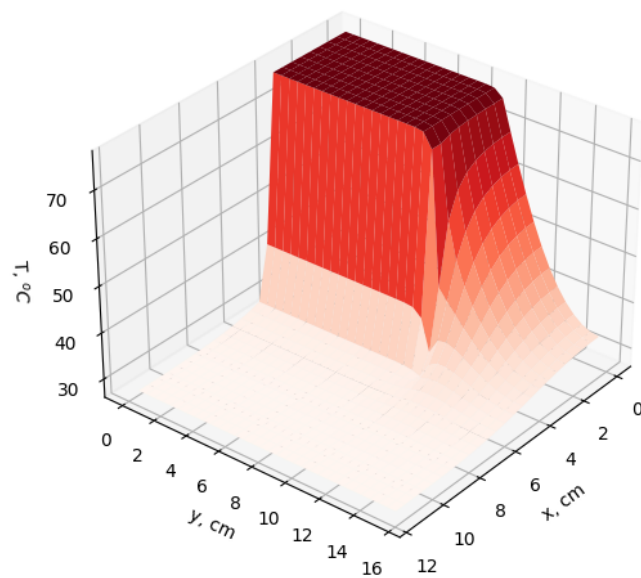


Рис. 3: Распределение температуры в $t = 30\text{s}$

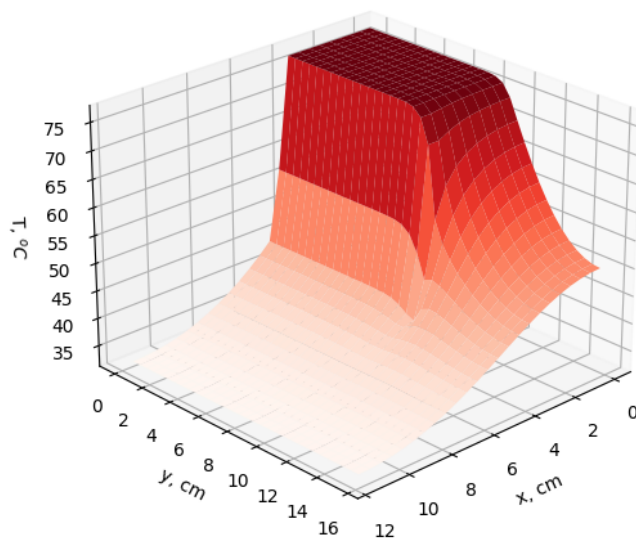


Рис. 4: Распределение температуры в $t = 120\text{s}$

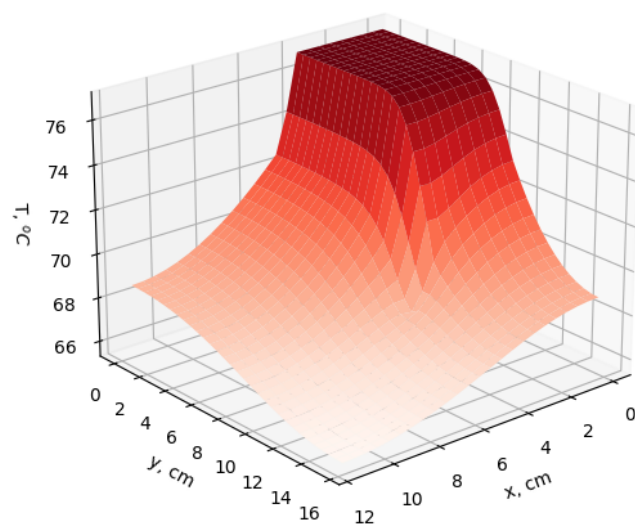


Рис. 5: Распределение температуры в $t = 600\text{c}$

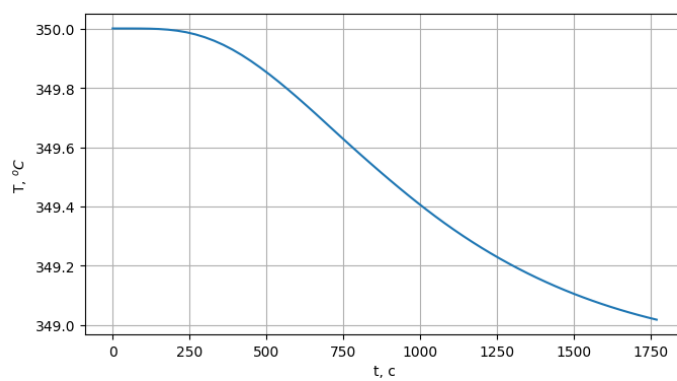


Рис. 6: Зависимость температуры в верхней части стакана кофе от времени ($x = 2, y = 0.97 \text{ cm}$).

3 Исходный код

```
1  # Задание 11. Создайте модель процесса остывания стеклянного стакана с горячим кофе при
2  # комнатных условиях. Постройте графики изменения температуры с учетом теплопроводности,
3  # конвекции и испарения, а также оцените точность интегрирования в зависимости от схемы
4  # интегрирования и величины шага интегрирования.
5
6  from matplotlib.colors import Colormap
7  import numpy as np
8  from scipy.integrate import solve_ivp
9  from scipy.optimize import minimize
10 import matplotlib.pyplot as plt
11 from mpl_toolkits.mplot3d import Axes3D
12
13 AMBIENT_TEMP = 300 # Комнатная температура, K
14 HUMIDITY = 85 # Влажность, %
15 V_AIR = 0.1 # Скорость воздуха m/s
16
17 # Размеры сетки, в метрах
18 X_ROOM = 0.12 # m
19 Y_ROOM = 0.16 # m
20
21 # Температуропроводности веществ
22 T_COND_WATER = 0.143*10**(-6)
23 T_COND_GLASS = 3.4*10**(-7)
24 T_COND_AIR = 1.9*10**(-5)
25
26 L = 2260000 # Удельная теплота парообразования Дж/кг
27 Cp = 4200 # Удельная теплоемкость, Дж/кгК
28 rho = 1000 # Плотность воды кг/м³
29
30 # Давление насыщенного пара воды, ммртст..
31 P_v_exp = np.array([4.585, 6.545, 9.212, 12.79, 17.54, 23.77, 31.84, 42.20, 55.37, 71.93, 92.59, 118.1])
32 P_v_exp_temps = np.array([0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55]) + 273
33
34 # Функция для расчета давления пара для заданной температуры
35 def get_Pv(T):
36     return np.interp(T, P_v_exp_temps, P_v_exp)
37
38 coffee_start_temp = 350 # Начальная температура кофе, K
39
40 # Внутренние размеры жидкости
41 x_coffee = 0.04 #m
42 y_coffee = 0.1 #m
43
44 # Координаты стакана
45 x_glass1 = 0.04
```

```

46 x_glass2 = 0.05
47 y_glass1 = 0
48 y_glass2 = y_coffee
49
50 # Шаг координатной сетки
51 x_step = 0.005
52 y_step = 0.005
53
54
55 def get_laplacian(T):
56     """ Расчет лапласиана для уравнения теплопроводности
57     """
58     d2Tdx2 = np.zeros_like(T)
59     d2Tdy2 = np.zeros_like(T)
60     size_x = len(T[0])
61     size_y = len(T[:,0])
62     for j in range(0, size_y):
63         for i in range(0, size_x):
64             if i==0:
65                 d2Tdx2[j, i] = ( 2*T[j, i+1] - 2*T[j, i]) / x_step**2
66             elif i==size_x-1:
67                 d2Tdx2[j, i] = ( -2*T[j, i] + 2*T[j, i-1] ) / x_step**2
68             else:
69                 d2Tdx2[j, i] = ( T[j, i+1] - 2*T[j, i] + T[j, i-1] ) / x_step**2
70
71             if j==0:
72                 d2Tdy2[j, i] = ( 2*T[j+1, i] - 2*T[j, i]) / y_step**2
73             elif j==size_y-1:
74                 d2Tdy2[j, i] = ( -2*T[j, i] + 2*T[j-1, i] ) / y_step**2
75             else:
76                 d2Tdy2[j, i] = ( T[j+1, i] - 2*T[j, i] + T[j-1, i] ) / y_step**2
77
78     return d2Tdx2 + d2Tdy2
79
80
81 def get_evaporation_speed(T):
82     # Скорость испарения, кгс/
83     W = x_coffee*x_step * (0.022 + 0.0174 * V_AIR)*get_Pv(T)/1000*(1 - HUMIDITY/100) / 3600
84     # W = np.pi*x_coffee**2 * (0.022 + 0.0174 * V_AIR)*get_Pv(T)*(1 - HUMIDITY/100) / 3600
85     return W
86
87
88 def heat_equation(t, params):
89     T = params[0:-1]
90     T = T.reshape(initial_shape)
91     m = params[-1]
92     # Учет испарения
93     dmdt = get_evaporation_speed(T[j_bnd, int(i_bnd/2)])

```



```

94     dT = np.zeros_like(T)
95     dT_evap = L*m/Cp/(rho*np.pi*x_coffee**2*y_step)
96     dT[j_bnd,0:i_bnd] = dT_evap
97     # Учет конвекции со стенкой
98     lamb = 0.6
99     for j in range(0, len(T[:,0])-1):
100         if j < j_bnd:
101             T[j, i_bnd] = (1 + x_step*120/lamb) * T[j, i_bnd-1] / 2
102
103     dTdt = T_COND*get_laplacian(T) - dT
104
105     return np.hstack((dTdt.flatten(), dmdt))
106
107
108     x = np.arange(0, X_ROOM, x_step)
109     y = np.arange(0, Y_ROOM, y_step)
110
111     i_bnd = None
112     j_bnd = None
113
114     X, Y = np.meshgrid(x, y)
115     start_temp_distr = np.zeros((len(y), len(x))) + AMBIENT_TEMP
116     T_COND = np.zeros_like(start_temp_distr)
117     # Задаем начальное распределение температуры, температуропроводности
118     for i, valx in enumerate(x):
119         if not i_bnd:
120             if valx>=x_coffee:
121                 i_bnd = i
122                 print(i_bnd)
123         for j, valy in enumerate(y):
124             if not j_bnd:
125                 if valy>=y_coffee:
126                     j_bnd = j
127                     print(j_bnd)
128             if valx<x_coffee and valy<y_coffee:
129                 start_temp_distr[j, i] = coffee_start_temp
130                 T_COND[j, i] = T_COND_WATER
131             elif valx>=x_glass1 and valx<=x_glass2 and valy>=y_glass1 and valy<=y_glass2:
132                 T_COND[j, i] = T_COND_GLASS
133             else:
134                 T_COND[j, i] = T_COND_AIR
135
136     # Время окончания моделирования, с
137     t_end = 60
138     time_points = np.arange(0, t_end, 30)
139     initial_shape = start_temp_distr.shape
140     print("Init shape", initial_shape)
141     sol1 = solve_ivp(heat_equation, [0, t_end], np.hstack((start_temp_distr.flatten(), 0)),

```

```

142         t_eval=time_points,
143         method='Radau')
144     T = sol1.y
145     t = sol1.t
146     Ts = sol1.y[:, -1]
147     Ts = Ts[0:-1].reshape(initial_shape)
148
149
150     fig = plt.figure(figsize=(6,6))
151     ax = fig.add_subplot(111, projection='3d')
152     ax.plot_surface(X*100, Y*100, Ts-273, cmap='Reds',
153                   linewidth=0, antialiased=True)
154
155     ax.set_ylabel('y, cm')
156     ax.set_xlabel('x, cm')
157     ax.set_zlabel('T, $^{\circ}$C')
158
159     plt.show()

```

Листинг 1: Исходный код задания