Курсовая работа по вычислительной математики "Задча Стефана"

Курохтин Александр, Б06-101

22 августа 2024 г.

1 Введение и постановка задачи

Задача Стефана – задача о распространении волны фазового перехода.

Пусть некоторое вещество может находиться (при разных температурах) в разных фазовых состояниях. Рассматривается цилиндрическая область и в ней известно начальное распределение температуры. Для определенности фазы – «жидкая» и «твердая». Определяющие уравнения:

Твердая фаза:

$$C_{solid}\frac{\partial u}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}(rK_{solid}\frac{\partial u}{\partial r}) \tag{1}$$

Жидкая фаза:

$$C_{liquid}\frac{\partial u}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}(rK_{liquid}\frac{\partial u}{\partial r}) \tag{2}$$

Условие Стефана:

$$\frac{ds}{dt} = -\frac{\partial u}{\partial x}(s(t)-0,t) \tag{3}$$

Начальные и граничные условия:

$$u(x,0) = u_0 \tag{4}$$

$$u(R,t) = f(t),\ \left.\frac{\partial u}{\partial r}\right|_{r=0} = 0 \tag{5}$$

2 Результаты и обсуждение

Решать задачу численно будем без явного выделения границы раздела фаз, но на каждом этапе её можно легко восстановить из условия $u_{border} = u_{melting}$. Что позволит автоматически привязать границу раздела к температуре плавления, по которой будет проходить разрыв в параметрах задачи

2.1 Сведение задачи к нелинейному уравнению теплопроводности

Будем считать, что коэффициент теплопроводности и теплоемкость - разрывные функции, зависящие только от температуры, то есть:

$$C(u)\frac{\partial u}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}(rK(u)\frac{\partial u}{\partial r}), \tag{6}$$

где $C(u) = C_{liquid}$ при $u > u_{melting}$, $C(u) = C_{solid}$ иначе (для коэффициента теплопроводности аналогично).

Введем энергию единицы в-ва W, которая при $u = u_{melting}$ испытывает скачок на величину, равную удельной теплоте плавления (λ):

$$W(u) = \int_0^u (C(u) + \lambda\cdot\delta(u-u_{melting}))\,du \tag{7}$$

Тогда подставляя энергию в уравнение (6):

$$\boxed{(\mathrm{C(u)} + \lambda\cdot\delta(u-u_{melting}))\frac{\partial u}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}(rK(u)\frac{\partial u}{\partial r})}, \tag{8}$$

Полученное уравнение и будет являтся искомой формой нелинейного уравнения теплопроводности. В нем нет зависимости от количества фаз, поэтому одновременно без усложнения задачи может находится несколько фазовых разделов.

2.2 Сглаживание коэффициентов

Для прехода к разностным схемам будем использовать следующие приближения для δ и θ функций с помощью непрерывной и бесконечно дифференцируемой функции гиперболического тангенса:

$$\theta(x,\alpha) = \frac{tanh(\alpha x)+1}{2} \tag{9}$$

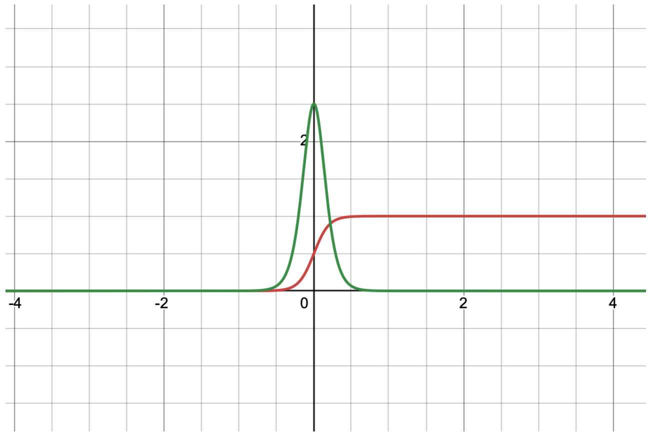

Рисунок 1 – Приближение дельта и тета функций при $\alpha = 5$

$$\delta(x, \alpha) = \frac{1}{2} \frac{dtanh(\alpha x)}{dx} = \frac{\alpha}{2 \cdot cosh^2(\alpha x)} \tag{10}$$

Воспользуемя тем фактом, что дельта функция является производной от тета функции, и заметим, что данные приближения зависят от одного управляющего параметра масштаба α.

Из этого получим приближение для коэффициента теплопроводности и эффективной теплоемкости:

$$K(u) = \frac{K_{solid} + K_{liquid}}{2} + tanh(\alpha x) \cdot \frac{K_{liquid} - K_{solid}}{2} \tag{11}$$

$$C(u) = \frac{C_{solid} + C_{liquid}}{2} + tanh(\alpha x) \cdot \frac{C_{liquid} - C_{solid}}{2} + \frac{\lambda \cdot \alpha}{2 \cdot cosh^2(\alpha x)} \tag{12}$$

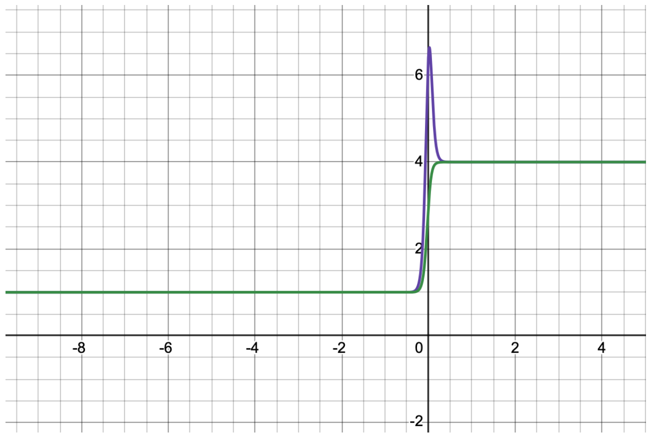

Рисунок 2 – Приближение эффективной теплоемкости при большой (фиолетовая) и малой теплоте плавления

2.3 Реализация численного метода

Рассмотрим разностную схему, в которой температура определяется в целых точках, а коэффициент теплопроводности в полуцелых:

$$C_m^n \frac{u_m^{n+1} - u_m^n}{\tau} = \frac{1}{h \cdot mh}[(m+1/2)h \cdot K_{m+1/2}^n \frac{u_{m+1}^{n+1} - u_m^{n+1}}{h} - (m-1/2)h \cdot K_{m-1/2}^n \frac{u_m^{n+1} - u_{m-1}^{n+1}}{h}] \tag{13}$$

- воспользуемся неявной схемой с нелинейностью на нижнем слое

Получим выражение, которое позволяет привести матрицу системы к трехдиагональной и воспользоваться методом прогонки:

$$-\sigma(m-1/2)K^n_{m-1/2}u^{n+1}_{m-1}+(C^n_m+\sigma[(m+1/2)]K^n_{m+1/2}+(m-1/2)K^n_{m-1/2}])u^{n+1}_m-\sigma(m+1/2)K^n_{m+1/2}u^{n+1}_{m+1}=C^n_m u^n_m \quad (14)$$

где введен параметр $\sigma = \frac{\tau}{h^2}$, аналогичный аналогу числа Куранта для параболических задач.

Реализация метода приведена в приложении.

Посмотрим на работоспособность схемы на примере реальных коэффициентов для системы "вода-лед"и граничного условия в виде постоянного значения (использовал -5):

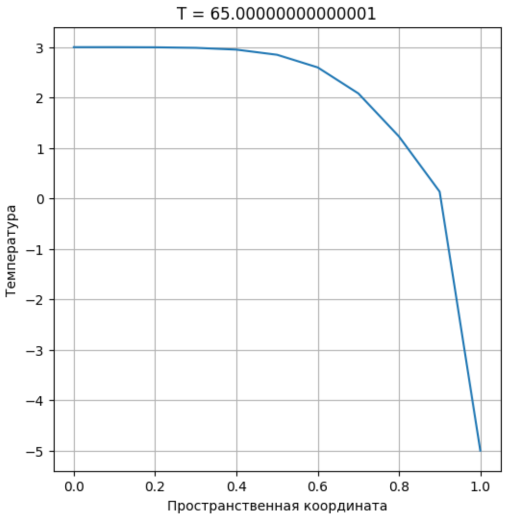

Рисунок 3 – Результат расчета для реальных параметров

Рельная система обладает слишком большой инертностью и установление температуры происходит за слишком большое время, но при этом это не является принципиальным моментом в исследовании закономерностей (все параметры задачи приведены в единицах СИ). Поэтому уменьшим значения для удельной теплоты плавления и теплоемкости.

2.4 Исследование задачи от параметров

Заметим, что результирующее решение зависит от: 1) параметра масшата α (задача о поиске оптимального гиперпараметра модели), 2) отношение между удельной теплотой плавления и теплоемкостями (успевает ли происходить релаксация в фазе, на которую набегает волна), 3) функции, задающей граничные условия (определяет количество фаз, одновременно находящихся в системе, и параметры скорости на границе, которые дальше передаются в систему)

2.4.1 Зависимость от модельных граничных условий

На систему подадим:

1) Граничное значение в виде постоянного значения (рис. 4) - в системе происходит быстрая релаксация в пограничном слое и последующее медленное промораживание стержня вглубь.

2) Граничное условие в виде синусоиды (рис. 5) - проверим возможность существовать в системе нескольких границ раздела фаз

Схема остаётся монотонной при всех значениях числа Куранта, поэтому схема остается устойчивой и в ней не появляются паразитные осцилляции. Теоретически возможно, что из-за большого временно шага можно пропустить какие-то эффекты, связанные с высокочастотным сигналом, но сама система не располагает к ним, обладая большой инертностью.

2.4.2 Зависимость от отношения параметров задачи

Если удельная теплота плавления становится малой по сравнению с коэффициентами теплопроводности, то уменьшается разрыв в тепловом потоке на границе раздела фаз, (не очень аккуратная

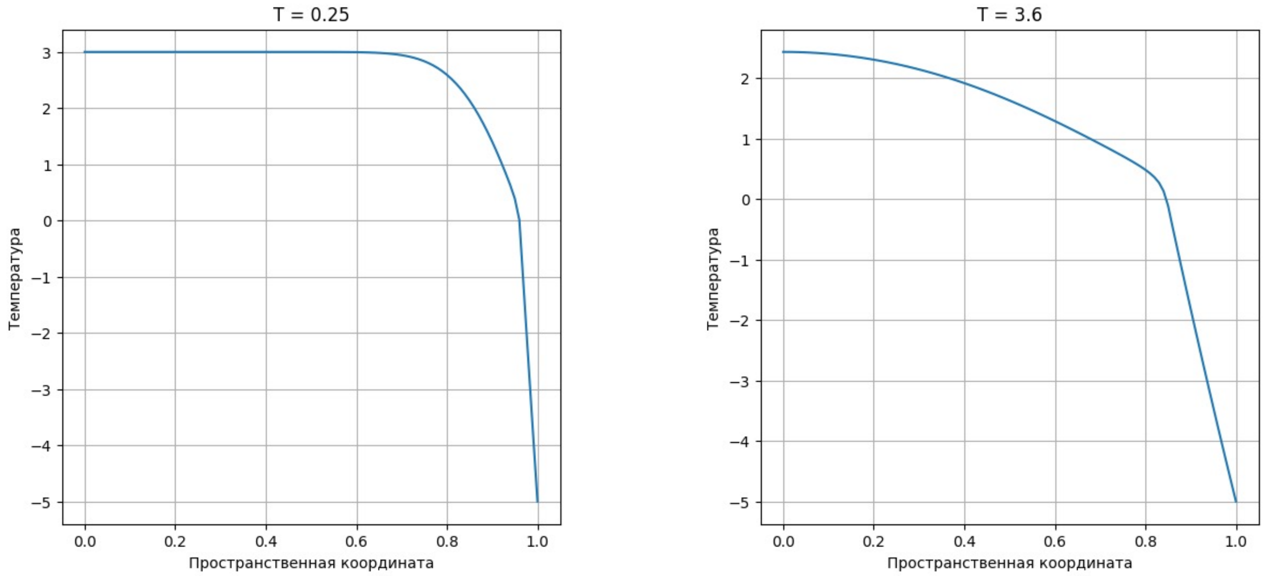

Рисунок 4 – Распространение теплоты в системе с сигналом в виде константы

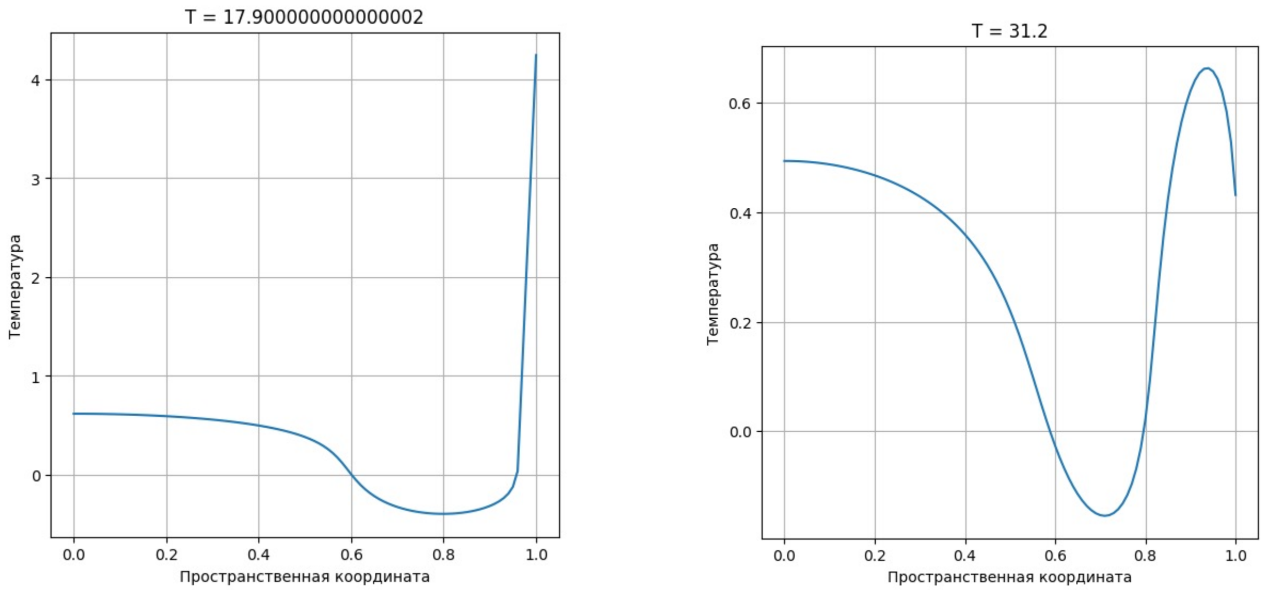

Рисунок 5 – Распространение теплоты в системе с сигналом в виде синусоиды

формулировка) система начинает жить как бы вместе с согласованным распростронением волным тепла, то есть сжимается пограничный релаксационный слой в жидкой фазе.

2.4.3 Зависимость от параметра масштаба сглаживания

При малом параметре α происходит сильное сглаживание параметров и на графике становится незаметным разрыв теплового потока (разрыв производной), то есть отдаляемся от точного решения. Однако при большом значении параметра может происходить то, что при тех значениях температуры на сетки можем не заметить то, что там есть дельта пик в эффективной теплоемкости.

Однако параметр сглаживание влияет на то, какое численное значение температуры увидем в итоге

3 Вывод

Метод без явного выделения фазовой границы позволяет упростить задачу тем, что автоматически привязывает границу фазового раздела к температуре плавления (с чем могут возникнуть проблемы при других реализациях) и не накладывает ограничение на количество фаз в системе.

Однако вместе с этим получили нечеткое положение границы раздела и зависимость решения от гиперпараметра сглаживания модели.

4 Литература

- Федоренко Р.П. "Введение в вычислительную физику: учебное пособие для вузов под ред. Лобанов А.И., 2008
- Самарский А.А., Моисеенко Б.Д. "ЭКОНОМИЧНАЯ СХЕМА СКВОЗНОГО СЧЕТА ДЛЯ МНОГОМЕРНОЙ ЗАДАЧИ СТЕФАНА 1965
- Лобанов А.И., Петров И.Б. "Вычислительная математика. Курс лекций 2021
- Б.М. БУДАК, Е.Н. СОЛОВЬЕВА, А.Б. УСПЕНСКИЙ "РАЗНОСТНЫЙ МЕТОД СО СГЛАЖИВАНИЕМ КОЭФФИЦИЕНТОВ ДЛЯ РЕШЕНИЯ ЗАДАЧ СТЕФАНА 1965

5 Приложение

```
# thermal conductivity as temperature function
def k(u, alpha):
    '''
    u - temperature
    alpha - scale parameter
    '''
    K_solid = 2.22  # thermal conductivity in solis [W/m*k]
    K_liquid = 0.596  # thermal conductivity in liquid [W/m*k]

    # apply smoothing with the function tanh
    return 0.5*(K_solid + K_liquid) + np.tanh(alpha*u)*0.5*(K_liquid - K_solid)

def c(u, alpha):
    c_solid = 4200 # J/kg*K
    c_liquid = 2100
    melt_const = 3.33*10**5 # J/kg
    return 0.5*(c_solid + c_liquid) + np.tanh(alpha*u)*0.5*(c_liquid - c_solid)
    + 0.5*melt_const*alpha/np.cosh(alpha*u)**2


def thomas_algorithm(N, a, b, c, d):
    '''
    solves Ax = d, where A is a tridiagonal matrix consisting of vectors a, b, c
    N = number of equations
    a[] = subdiagonal
    b[] = main diagonal
    c[] = superdiagonal
    d[] = right part of equasion
    '''
    P, Q = np.zeros(N), np.zeros(N)
    P[0], Q[0] = c[0]/b[0], d[0]/b[0]

    for i in range(1, N):
```

```
        P[i] = c[i]/(b[i] - a[i]*P[i-1])
        Q[i] = (d[i] - a[i]*Q[i-1])/(b[i] - a[i]*P[i-1])

    ans = np.zeros(N)
    ans[N-1] = Q[N-1]
    for i in range(N-2, -1, -1):
        ans[i] = Q[i] - P[i]*ans[i+1]
    return ans

def implicit_scheme(u, c, k, R, T, h, courant, alpha):
    '''
    courant - in our problem accept as h^2/tau
    R - cylinder radius
    T - time
    h - step in space
    u - initial condition
    c - heat capacity function
    k - thermal conductivity function
    alpha - smoothing scale paramether
    '''

    # method parameters
    tau = courant * h ** 2  # time step
    N = len(u)  # number of net points

    show_progress(u, R, h, 0)
    for i in range(1, int(T / tau) + 1):
        '''
        a[] = subdiagonal
        b[] = main diagonal
        c[] = superdiagonal
        d[] = right part of equasion

        left border condition (r=0) on gradient (symmetry condition)
        right border condition (r=R) on meaning
        '''
        a_thomas = [0] + [-courant * (1 - 1 / (2*m)) * k((u[m-1]+u[m])/2, alpha)
            for m in range(1, N - 1)] + [0]
        b_thomas = [-1] + [c(u[m], alpha) + courant * ((1 - 1 / (2*m)) * k((u[m-1]+u[m])/2
            + (1 + 1 / (2*m)) * k((u[m]+u[m+1])/2, alpha)) for m in range(1, N - 1)] + [1]
        c_thomas = [1] + [-courant * (1 + 1 / (2*m)) * k((u[m]+u[m+1])/2, alpha)
            for m in range(1, N - 1)] + [0]
        d_thomas = [0] + list(u[1:-1]*c(u[1:-1], alpha)) + [f_border(i * tau)]
        u = thomas_algorithm(N, a_thomas, b_thomas, c_thomas, d_thomas)


        if i % 100 == 0:
            show_progress(u, R, h, i*tau)
    return u
```