Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Лабораторная работа №2 по дисциплине “Математические модели систем с распределёнными параметрами”

Выполнил студент гр. 3530904/91002 Афанасьев Е. Д.
Преподаватель Воскобойников С. П.

Санкт-Петербург 2021

Оглавление

Постановка задачи 3

Дискретная модель 4

Алгоритм решения 6

Тесты для модели 8

Результаты тестирования 10

Выводы 11

Код программы 12

Постановка задачи

Разработка программ для моделирования стационарного одномерного распределения температуры интегро-интерполяционным методом (методом баланса).

Постановка задачи. Вариант D1. Используя интегро-интерполяционный метод (метод баланса) разработать программу для моделирования нестационарного распределения температуры в шаре, описываемого математической моделью вида

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(k(x,t)\frac{\partial u}{\partial x}\right) - q(x,t)u + f(x,t), \qquad x \in [0, L] \quad t \in [0, T]$$

$$0 < c_1 \le k(x,t) \le c_2, \qquad 0 \le q(x,t)$$

С начальным условием $u|_{t=0} = \varphi(r)$ и граничными условиями

$u|_{x=0} = \nu_1,$ $\qquad$ $u|_{x=L} = \nu_2,$

Дискретная модель

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(k(x,t)\frac{\partial u}{\partial x}\right) - q(x,t)u + f(x,t), x \in [0, L], t \in [0, T]$$

$$0 < c_1 \le k(x,t) \le c_2, \quad 0 \le q(x,t)$$

$$u|_{x=0} = \nu_1(t), \quad u|_{x=L} = \nu_2(t), \quad u|_{t=0} = \varphi(x)$$

N - число разбиений инт. $[0, L]$

$$x_0 < x_1 < \ldots < x_N, \quad x_0 = 0, \quad x_N = L$$

$$u = u(x,t), \quad k = k(x,t), \quad q = q(x,t), \quad f = f(x,t)$$

$$h_i = x_i - x_{i-1} \qquad x_{i-\frac{1}{2}} = \frac{x_i + x_{i-1}}{2}, \quad i = 1, 2, \ldots, N$$

$$\hbar_i = \begin{cases} \frac{h_{i+1}}{2}, & i = 0 \\ \frac{h_i + h_{i+1}}{2}, & i = 1, 2, \ldots, N-1 \\ \frac{h_i}{2}, & i = N \end{cases}$$

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \frac{\partial u}{\partial t} dx = \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \left[\frac{\partial}{\partial x}\left(k\frac{\partial u}{\partial x}\right) - qu\right] dx + \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} f\, dx$$

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \frac{\partial u}{\partial t} dx = \left[k\frac{\partial u}{\partial x}\Big|_{x = x_{i+\frac{1}{2}}} - k\frac{\partial u}{\partial x}\Big|_{x = x_{i-\frac{1}{2}}} + \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} qu\, dx\right] + \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} f dx$$

$$k\frac{\partial u}{\partial x}\Big|_{x = x_{i-\frac{1}{2}}} \approx k_{i-\frac{1}{2}} \frac{u_i - u_{i-1}}{2\frac{h_i}{2}} = k_{i-\frac{1}{2}} \frac{u_i - u_{i-1}}{h_i}$$

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \varphi(x) dx \approx \hbar_i \varphi_i$$

$u|_{x=0} = \nu_1(t)$, где $i=0$

$\hbar_i \frac{dv_i}{dt} = \left[k_{i+\frac{1}{2}} \frac{v_{i+1}-v_i}{h_{i+1}} - k_{i-\frac{1}{2}} \frac{v_i - v_{i-1}}{h_i} - \hbar_i q_i v_i\right] + \hbar_i f_i, \quad i=1,2,\dots,N-1$

$u|_{x=L} = \nu_2(t)$, где $i=N$

Алгоритм решения

$$D\frac{dv}{dt} = Av + g, \quad A - (N+1)(N+1), \quad v, g \in R^{(N+1)}$$

D - диаг. матрица коэф. перед $\frac{dv}{dt}$

A - матр. коэф. реш-я

g - вектор ~~реш~~ правой части

коэф. A и g

a	bc	b	g	i
0	-1	0	$v_1(t)$	0
$\frac{k_{i-\frac{1}{2}}}{h_i}$	$-\frac{k_{i+\frac{1}{2}}}{h_{i+1}} - \frac{k_{i-\frac{1}{2}}}{h_i} - \hbar q_i$	$\frac{k_{i+\frac{1}{2}}}{h_{i+1}}$	$\hbar_i f_i$	$1, 2, \dots, N-1$
0	-1	0	$v_2(t)$	N

Выразим старшую произв.

$$\frac{dv}{dt} = \tilde{A} v + \tilde{g}, \quad \tilde{A} = D^{-1} A, \quad \tilde{g} = D^{-1} g$$

Выразим реш. для явного мет. Эйлера ($i = 1, \dots, N+1$)

$$v_i = v_{i-1} + \tau (\tilde{A}_{i-1} v_{i-1} + \tilde{g}_{i-1})$$

Будем решать пошаг. методом

Огр-е на шаг $\tau = \frac{2}{\max_i |\lambda_i|}$, $\max_i |\lambda_i| < \|A\| \sim \frac{1}{h^2}$

$\Rightarrow \tau < \frac{1}{h^2}$

Выразим реш-я для неявного м. Эйлера ($i = 1, \dots, N$)

$$\left(\frac{1}{\tau} E - \tilde{A}_i\right) v_i = \frac{v_{i-1}}{\tau} + \tilde{g}_i$$

огр-я $\tau > 0$, будем решать методом прогонки

Тесты для модели

$T=1,\ L=1$

конст. случай $\quad k=1,\ q=1,\ u=1,\ f=1,\ \varphi(x)=1$

$$\frac{\partial u}{\partial t}=\frac{\partial}{\partial x}\left(k\frac{\partial u}{\partial x}\right)-qu+f$$

$$\frac{\partial u}{\partial t}=0-1+1,\quad \frac{\partial u}{\partial t}=0$$

$$u|_{x=0}=1,\quad u|_{x=L}=1,\quad \frac{\partial \nu_1(t)}{\partial t}=0,\quad \frac{\partial \nu_2(t)}{\partial t}=0$$

$$\nu_1(t)=0,\ \nu_2(t)=1$$

зависим. от x $\quad k=x^2+1,\ q=x^2,\ u=x^2,\ \varphi(x)=x^2$

$$\frac{\partial u}{\partial t}=\frac{\partial}{\partial x}\left(k\frac{\partial u}{\partial x}\right)-qu+f$$

$$0=\frac{\partial}{\partial x}\left((x^2+1)\frac{\partial x^2}{\partial x}\right)-x^4+f$$

$$f=x^4-6x^2-2$$

$$u|_{x=0}=0,\quad u|_{x=L}=L^2,\quad \frac{\partial \nu_1(t)}{\partial t}=0,\quad \frac{\partial \nu_2(t)}{\partial t}=0$$

$$\nu_1(t)=0,\ \nu_2(t)=L^2$$

завис. от x и t $\quad k = x e^{-t} + 1, \; q = x e^{-t}, \; u = x e^{-t}, \; \varphi(x) = x$

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(k \frac{\partial u}{\partial x}\right) - q u + f$$

$$\frac{\partial x e^{-t}}{\partial t} = \frac{\partial}{\partial x}\left((x e^{-t} + 1) \frac{\partial x e^{-t}}{\partial x}\right) - x^2 e^{-2t} + f$$

$$-x e^{-t} = \frac{\partial}{\partial x}\left(x e^{-2t} + e^{-t}\right) - x^2 e^{-2t} + f$$

$$f = x^2 e^{-2t} - x e^{-t} - e^{-2t}$$

$$\nu_1(t) = 0, \; \nu_2(t) = L^2 e^{-t}$$

$$u|_{x=0} = 0, \; u|_{x=L} = L^2 e^{-t}$$

$$\frac{\partial \nu_1(t)}{\partial t} = 0, \; \frac{\partial \nu_2(t)}{\partial t} = -L^2 e^{-t}$$

Результаты тестирования

Приведу таблицы результатов работы программы, n – число разбиений по L, m – по T.

Константный случай

m	10	10	100	100	1000	1000
n	Явный	Неявный	Явный	Неявный	Явный	Неявный
4	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
8	0.000000e+00	4.440892e-16	0.000000e+00	2.220446e-16	0.000000e+00	0.000000e+00
16	0.000000e+00	6.661338e-16	0.000000e+00	2.220446e-16	0.000000e+00	0.000000e+00
32	0.000000e+00	4.440892e-16	0.000000e+00	2.220446e-16	0.000000e+00	4.440892e-16
64	0.000000e+00	3.330669e-15	0.000000e+00	2.664535e-15	0.000000e+00	4.329870e-15
128	0.000000e+00	5.107026e-15	0.000000e+00	1.598721e-14	0.000000e+00	7.560619e-14
256	0.000000e+00	1.532108e-14	0.000000e+00	4.185541e-14	0.000000e+00	4.929390e-13

Случай зависимости от x

m	10	10	100	100	1000	1000
n	Явный	Неявный	Явный	Неявный	Явный	Неявный
4	3.425449e-04	5.662792e-03	2.631765e-03	5.664298e-03	5.655033e-03	5.664298e-03
8	8.587160e-05	1.422578e-03	6.754210e-04	1.422892e-03	1.421025e-03	1.422892e-03
16	2.147621e-05	3.618794e-04	1.717394e-04	3.619561e-04	3.615042e-04	3.619561e-04
32	1.253291e-05	9.049524e-05	1.705466e+20	9.051418e-05	inf	9.051418e-05
64	2.566010e+01	2.262540e-05	1.336639e+89	2.263012e-05	inf	2.263012e-05
128	4.659225e+06	5.656449e-06	7.495436e+153	5.657628e-06	inf	5.657628e-06
256	3.952653e+11	1.414118e-06	2.819183e+215	1.414413e-06	inf	1.414413e-06

Случай зависимости от x и t

m	10	10	100	100	1000	1000
n	Явный	Неявный	Явный	Неявный	Явный	Неявный
4	6.320945e-01	1.566764e-01	6.318918e-01	9.953298e-02	6.312714e-01	9.354425e-02
8	6.320692e-01	1.251078e-01	6.317240e-01	5.965954e-02	6.311885e-01	5.272882e-02
16	6.320206e-01	1.087644e-01	6.315077e-01	3.643168e-02	6.311533e-01	2.880078e-02
32	6.319312e-01	9.896647e-02	1.459913e+22	2.365282e-02	inf	1.553787e-02
64	3.016150e+04	9.516258e-02	1.608542e+94	1.687069e-02	inf	8.466089e-03
128	3.794026e+09	9.516258e-02	1.166862e+159	1.339357e-02	inf	7.673391e-03
256	2.925673e+14	9.516258e-02	5.478455e+219	1.162847e-02	inf	7.673549e-03

Выводы

Для константного случая явный метод не имеет погрешности, поэтому лучше применять его для данной задачи.

Для зависимости от x для малого количества разбиений явный метод дает точность схожую с неявным методом, однако начиная с некоторого n, начинает расходиться и выдавать огромную ошибку. Причем, чем больше m, тем быстрее расходится явный метод. Неявный же метод сходится быстро при любых m, поэтому в случае зависимости от x следует применять его.

Для зависимости от x и от t получены схожие с предыдущим случаем результаты.

Системы с плохой обусловленностью нельзя решать с помощью явного метода, он начинает быстро расходиться. Для их решения следует выбрать неявный метод. Однако системы с небольшой обусловленностью можно решать явным методом

Код программы

Программа разработана на python 3.7

main.py

```
import numpy as np
from solve import *

#case = Cases.LINEAR
case = Cases.CONST
#case = Cases.NON_LINEAR
debug = True
debug = False
n = 4
m = 10
#for case in range(3):
for i in range(3):
    n = 4
    print(m)
    for j in range(7):
        ans1 = solve_explicit_euler(n, m, case)
        ans2 = solve_not_explicit_euler(n, m, case)
        print(n, m)
        print("explicit", end=' ')
        print_error(n, m, ans1, case, debug)
        print("not explicit", end=' ')
        print_error(n, m, ans2, case, debug)
        n *= 2
    m *= 10
```

solve.py

```
import numpy as np
from coef import *


def solve_matrix(n, aa, cc, bb, gg):
    x = np.zeros(n)
    a = aa.tolist()
    c = cc.tolist()
    b = bb.tolist()
    g = gg.tolist()
    for i in range(1, n):
        m = a[i] / c[i-1]
        c[i] = c[i] - m * b[i - 1]
        g[i] = g[i] - m * g[i - 1]

    x[n-1] = g[n - 1]/c[n-1]

    for i in range(n-2, -1, -1):
        x[i] = (g[i] - b[i] * x[i + 1]) / c[i]

    return x


def solve_explicit_euler(n, m, case):
    step_t = get_T() / m / 1000
    t = 0
    ans = np.zeros((m + 1, n + 1))
    inv_D = np.linalg.inv(get_D(n))
    _, _, _, _, fi = init_coef(n, 0, case)
    ans[0] = fi
```

```
    A, g = get_A_and_g(n, t, case)
    A_w = np.dot(inv_D, A)
    g_w = np.dot(inv_D, g)

    for i in range(1, m + 1):
        t += step_t

        b = np.dot(A_w, ans[i - 1])
        ans[i] = ans[i - 1] + t * (b + g_w)

        A, g = get_A_and_g(n, t, case)
        A_w = np.dot(inv_D, A)
        g_w = np.dot(inv_D, g)

    return ans


def solve_not_explicit_euler(n, m, case):
    step_t = get_T() / m
    t = 0
    inv_D = np.linalg.inv(get_D(n))
    ans = np.zeros((m + 1, n + 1))
    _, _, _, _, fi = init_coef(n, 0, case)
    ans[0] = fi

    A, g = get_A_and_g(n, t, case)
    A_w = np.dot(inv_D, A)
    g_w = np.dot(inv_D, g)

    for i in range(1, m + 1):
        t += step_t
        B = np.eye(n + 1) / step_t - A_w
        C = ans[i - 1] / step_t + g_w

        a, c, b = get_coefs_from_A(n, B)
        ans[i] = solve_matrix(n + 1, a, c, b, C)

        A, g = get_A_and_g(n, t, case)
        A_w = np.dot(inv_D, A)
        g_w = np.dot(inv_D, g)

    return ans


def check_error(n, m, ans, case, debug=False):
    step_T = get_T() / m
    h = get_L() / n
    error = []
    max_error = 0
    t = 0
    gr = np.zeros((m + 1, n + 1))
    for a in ans:
        x = 0
        er = []
        for v in a:
            max_error = max(max_error, abs(v - get_u(x, t, case)))
            er.append(abs(v - get_u(x, t, case)))
            x += h
        t += step_T
        error.append(er)

    if debug:
```

```
        x = 0
        t = 0
        for i in range(m + 1):
            for j in range(n + 1):
                gr[i, j] = get_u(x, t, case)
                x += h
            t += step_T

    return max_error, error, gr


def print_error(n, m, ans, case, debug=False):
    m, e, gt = check_error(n, m, ans, case, debug)

    print('{0:3e}'.format(m))
    if debug:
        print("answer ", *ans)
        print("error ", *e)
        print("real answer ", *gt)


def solve(debug, case):
    n = 4
    m = 10
    th = []
    td = []

    for i in range(3):
        n = 4
        th.append(m)
        th.append(m)
        for j in range(7):
            ans1 = solve_explicit_euler(n, m, case)
            ans2 = solve_not_explicit_euler(n, m, case)
            m1, _, _ =check_error(n, m, ans1, case, debug)
            m2, _, _ =check_error(n, m, ans2, case, debug)
            td.append(m1)
            td.append(m2)
            n *= 2
        m *= 10
```
coef.py

```
import enum
import math

import numpy as np


class Cases(enum.Enum):
    CONST = 0
    LINEAR = 1
    NON_LINEAR = 2


def get_k(x, t, case):
    if case == Cases.CONST:
        return 1
    elif case == Cases.LINEAR:
        return x**2 + 1
    elif case == Cases.NON_LINEAR:
        return x * math.exp(-t) + 1
```

```python
def get_q(x, t, case):
    if case == Cases.CONST:
        return 1
    elif case == Cases.LINEAR:
        return x**2
    elif case == Cases.NON_LINEAR:
        return x * math.exp(-t)


def get_u(x, t, case):
    if case == Cases.CONST:
        return 1
    elif case == Cases.LINEAR:
        return x ** 2
    elif case == Cases.NON_LINEAR:
        return x * math.exp(-t)


def get_f(x, t, case):
    if case == Cases.CONST:
        return 1
    elif case == Cases.LINEAR:
        return x ** 4 - 6 * x ** 2 - 2
    elif case == Cases.NON_LINEAR:
        return x ** 2 * math.exp(-t) - x * math.exp(-t) - math.exp(-2*t)


def get_fi(x, t, case):
    if case == Cases.CONST:
        return 1
    elif case == Cases.LINEAR:
        return x ** 2
    elif case == Cases.NON_LINEAR:
        return x * math.exp(-t)


def get_L():
    return 1

def get_T():
    return 1


def get_v1(case):
    if case == Cases.CONST:
        return 1
    elif case == Cases.LINEAR:
        return 0
    elif case == Cases.NON_LINEAR:
        return 0


def get_dv1(case):
    return 0


def get_v2(t, case):
    if case == Cases.CONST:
        return 1
    elif case == Cases.LINEAR:
        return get_L() ** 2
```

```python
    elif case == Cases.NON_LINEAR:
        return get_L() ** 2 * math.exp(-t)


def get_dv2(t, case):
    if case == Cases.CONST:
        return 0
    elif case == Cases.LINEAR:
        return 0
    elif case == Cases.NON_LINEAR:
        return -get_L() ** 2 * math.exp(-t)


def get_str(case):
    if case == Cases.CONST:
        return "CONST"
    elif case == Cases.LINEAR:
        return "LINEAR"
    elif case == Cases.NON_LINEAR:
        return "NON LINEAR"


def get_D(n):
    h = get_L() / n
    D = np.zeros((n + 1, n + 1))
    D[0, 0] = h / 2
    D[n, n] = h / 2
    for i in range(1, n):
        D[i, i] = h

    return D


def init_coef(n, t, case):
    k = np.zeros(n + 1)
    q = np.zeros(n + 1)
    u = np.zeros(n + 1)
    f = np.zeros(n + 1)
    fi = np.zeros(n + 1)
    h = get_L() / n
    x = 0
    for i in range(n + 1):
        k[i] = get_k(x, t, case)
        q[i] = get_q(x, t, case)
        u[i] = get_u(x, t, case)
        f[i] = get_f(x, t, case)
        fi[i] = get_fi(x, t, case)
        x += h
    return k, q, u, f, fi


def init_coef_of_A(n, t, case):
    step = get_L() / n
    k, q, u, f, fi = init_coef(n, t, case)
    a = np.zeros(n + 1)
    c = np.zeros(n + 1)
    b = np.zeros(n + 1)
    g = np.zeros(n + 1)

    a[0] = 0
    c[0] = -1
    b[0] = 0
```

```
    g[0] = get_v1(case)

    a[1] = 0
    c[1] = -(k[1] + k[0]) / (2 * step) - (k[1] + k[2]) / (2 * step) - step * q[1]
    b[1] = (k[1] + k[2]) / (2 * step)
    g[1] = step * f[1] + get_v1(case) * (k[1] + k[0]) / (2 * step)

    for i in range(2, n-1):
        a[i] = (k[i] + k[i-1]) / (2 * step)
        c[i] = -(2 * k[i] + k[i - 1] + k[i + 1]) / (2 * step) - step * q[i]
        b[i] = (k[i] + k[i + 1]) / (2 * step)
        g[i] = step * f[i]

    a[n - 1] = (k[n - 1] + k[n - 2]) / (2 * step)
    c[n - 1] = -(k[n - 1] + k[n - 2]) / (2 * step) - (k[n - 1] + k[n]) / (2 * step) -
step * q[n - 1]
    b[n - 1] = 0
    g[n - 1] = step * f[n - 1] + get_v2(t, case) * (k[n - 1] + k[n]) / (2 * step)

    a[n] = 0
    c[n] = -1
    b[n] = 0
    g[n] = get_v2(t, case)

    return a, b, c, g


def get_A_and_g(n, t, case):
    a, b, c, g = init_coef_of_A(n, t, case)
    A = np.zeros((n + 1, n + 1))
    A[0, 0] = c[0]
    A[0, 1] = b[0]

    for i in range(1, n):
        A[i, i - 1] = a[i]
        A[i, i] = c[i]
        A[i, i + 1] = b[i]

    A[n, n] = c[n]
    A[n, n - 1] = a[n]

    return A, g


def get_coefs_from_A(n, A):
    a = np.zeros(n + 1)
    c = np.zeros(n + 1)
    b = np.zeros(n + 1)

    a[0] = 0
    c[0] = A[0, 0]
    b[0] = A[0, 1]

    for i in range(1, n):
        a[i] = A[i, i - 1]
        c[i] = A[i, i]
        b[i] = A[i, i + 1]

    c[n] = A[n, n]
    b[n] = 0
    a[n] = A[n, n - 1]
    return a, c, b
```