Московский авиационный институт

(национальный исследовательский университет)

Институт №8 «Информационные технологии и прикладная математика»

**Лабораторная работа №3 по численным методам**

**7 семестр**

Студент: Сухотин Игорь

Группа: М8O-402Б

Москва, 2020

# Постановка задачи

Пункт 2.1:

Решить начально-краевую задачу для ДифУрЧап параболического типа. Использовать схему: Кранка-Никлсона. Осуществить реализацию варианта аппроксимации граничных условий, содержащих производные:

– двухточечная с первым порядком точности

# Метод решения

Рассмотрим уравнение теплопроводности в общем виде:

1. Сделаем *дискретизацию* пространственной области с малым шагом :

а также – временной области (для этого нужно выбрать конечный момент времени ) с малым шагом

Теперь для решения задачи (1)-(4) необходимо определить значение искомой функции в узлах дискретизации .

Далее, для краткости будем писать

1. Если применить двухточечную аппроксимацию производных:
2. *Смешанная схема*.

Аппроксимируем уравнение (1) конечными разностями следующим образом:

и повторим те же шаги, что и в неявной схеме. После необходимых действий получится довольно громоздкая, но всё же линейная система с трёхдиагональной матрицей, которую можно решать методом прогонки на каждом временном слое .

При этом глобальный порядок аппроксимации по останется первым, а не вторым, из-за аппроксимации граничных условий. Схема Кранка-Никлсона получится, если положить .

Но у нас задача с ненулевой левой границей:

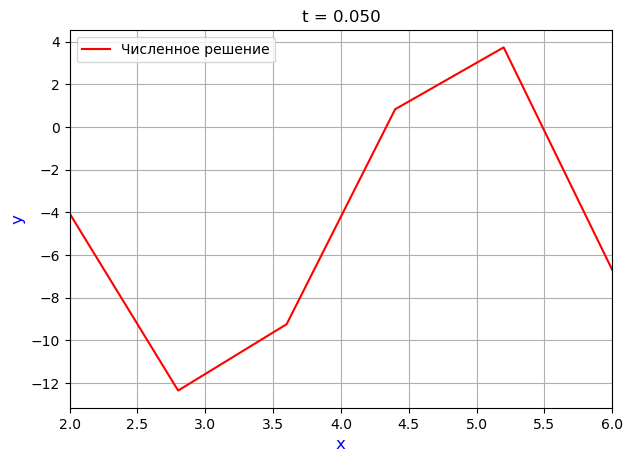
Чтобы перейти от задачи с ненулевой левой границей к задаче с нулевой левой границей нужно сделать замены:

После решения задачи выше, можно легко перейти обратно по формулам:

# Результат решения

Написана дпрограмма на языке *Python* с применением фреймворков *numpy и matplotlib. Среда программирования: Spyder 3.*





# Выводы

Решена задача уравнения математической физики, а именно одномерное параболическое уравнение с краевыми условиями третьего рода. Физический смысл этого уравнения и искомой функции заключается в распределении температур внутри ограниченного стержня.

# Листинг

***main.py***

*# -\*- coding: utf-8 -\*-*

*#%% Либы + настройка*

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**from** num\_methods **import** heat\_eq\_solve

**from** pandas **import** DataFrame

*#%% Ввод параметров варианта*

a = 2

b = 0

c = 1.25

f = **lambda** x, t: (x + 1) / (t + 4)

l = 4

h = 0.8

tau = 0.025

T = 0.1

N = int((6 - 2) / h + 1)

K = int(T / tau + 1)

phi0 = **lambda** t: -4 + t / (-1)

phil = **lambda** t: -39.41592654 + t \*\* 2 / (-2) - 0.4 \* t

alpha = [[0, 1],

[2, 2]]

psi = **lambda** x: -4 + 10 \* np.sin(1.57079633 \* x)

F = **lambda** x, t: f(x + 2, t)

Psi = **lambda** x: psi(x + 2)

*#%% Решение*

*# N = 20*

*# K = 100*

x, t, u = heat\_eq\_solve(a, Psi, phi0, phil,

l, T,

N=N, K=K,

b=b, c=c, f=F,

alpha=alpha,

method="comb", order=1, tetta=0.5)

x = x + 2;

tau = t[1] - t[0]

h = x[1] - x[0]

u\_table = DataFrame(u, columns=["t = {:.3f}".format(t\_) **for** t\_ **in** t ])

**print**(u\_table)

*#%% График при фиксированном времени*

t\_ = 0.05

ti = int(np.round(t\_ \* (K - 1) / T))

plt.figure(figsize=(7, 5), dpi=100) *# 700x500*

plt.plot(x, u[:, ti], "r", markersize = 3, label="Численное решение")

plt.xlim((x[0], x[-1]))

plt.xlabel('x', fontsize=12, color='blue')

plt.ylabel('y', fontsize=12, color='blue')

plt.title("t = {:.3f}".format(t[ti]))

plt.legend()

plt.grid(True)

plt.savefig("plot.png", dpi=100)

***num\_methods.py***

*# -\*- coding: utf-8 -\*-*

**import** numpy **as** np

**def** prog\_solve(el, d, eu, b):

"""

Метод прогонки решения системы с трёхдиагональной матрицей

"""

n = len(d)

eu = np.concatenate((eu, [0]))

P = np.zeros(n)

Q = np.zeros(n)

P[0] = -eu[0] / d[0]

Q[0] = b[0] / d[0]

**for** i **in** range(1, n):

P[i] = -eu[i] / (d[i] + el[i - 1] \* P[i - 1])

Q[i] = (b[i] - el[i - 1] \* Q[i - 1]) / (d[i] + el[i - 1] \* P[i - 1])

x = np.zeros(n)

x[-1] = Q[-1]

**for** i **in** range(n - 2, -1, -1):

x[i] = P[i] \* x[i + 1] + Q[i]

**return** x

**def** heat\_eq\_solve(a, psi, phi0, phil,

l, T,

N=10, K=20, sigma=None,

b=0, c=0, f=np.vectorize(**lambda** x, t: 0),

alpha=[[0, 1], [0, 1]], order=3,

method="comb", tetta=0.5):

**if** method == "comb":

**if** tetta == 0:

method = "explicit"

**elif** method == 1:

method = "implicit"

x = np.linspace(0, l, N)

h = x[1] - x[0]

*# пользователь указал сигму*

**if** sigma != None:

tau = sigma \* h \*\* 2 / a

t = np.arange(0, T+1e-9, tau)

K = len(t)

**else**:

t = np.linspace(0, T, K)

tau = t[1] - t[0]

a1 = alpha[0][0]

b1 = alpha[0][1]

a2 = alpha[1][0]

b2 = alpha[1][1]

u = np.zeros((N, K))

f = np.array([f(x\_i, t) **for** x\_i **in** x])

phi0 = phi0(t)

phil = phil(t)

u[:, 0] = psi(x) *# 1 слой*

**if** method == "explicit":

"""

Явная схема

"""

**if** order == 1:

**for** k **in** range(1, K):

u[1:N-1, k] = (a \* (u[2:N, k-1] - 2 \* u[1:N-1, k-1] + u[0:N-2, k-1]) / h \*\* 2 +

b \* (u[2:N, k-1] - u[0:N-2, k-1]) / (2 \* h) +

c \* u[1:N-1, k-1] +

f[1:N-1, k-1]) \* tau + \

u[1:N-1, k-1]

u[0, k] = (a1 \* u[1, k] - h \* phi0[k]) / (a1 - h \* b1)

u[N-1, k] = (a2 \* u[N-2, k] + h \* phil[k]) / (a2 + h \* b2)

**elif** order == 2:

**for** k **in** range(1, K):

u[1:N-1, k] = (a \* (u[2:N, k-1] - 2 \* u[1:N-1, k-1] + u[0:N-2, k-1]) / h \*\* 2 +

b \* (u[2:N, k-1] - u[0:N-2, k-1]) / (2 \* h) +

c \* u[1:N-1, k-1] +

f[1:N-1, k-1]) \* tau + \

u[1:N-1, k-1]

mul0 = - 3 \* a1 / (2 \* h) + b1

mull = 3 \* a2 / (2 \* h) + b2

u[0, k] = (- 4 \* a1 \* u[1, k] / (2 \* h) + a1 \* u[2, k] / (2 \* h) + phi0[k]) / mul0

u[N-1, k] = (4 \* a2 \* u[N-2, k] / (2 \* h) - a2 \* u[N-3, k] / (2 \* h) + phil[k]) / mull

**elif** order == 3:

**for** k **in** range(1, K):

u[1:N-1, k] = (a \* (u[2:N, k-1] - 2 \* u[1:N-1, k-1] + u[0:N-2, k-1]) / h \*\* 2 +

b \* (u[2:N, k-1] - u[0:N-2, k-1]) / (2 \* h) +

c \* u[1:N-1, k-1] +

f[1:N-1, k-1]) \* tau + \

u[1:N-1, k-1]

mul0 = 2 \* a1 \* a / (2 \* a - b \* h)

mull = 2 \* a2 \* a / (2 \* a + b \* h)

u[0, k] = (phi0[k] - mul0 \* (h \* u[0, k - 1] / (2 \* a \* tau) + h \* f[0, k] / (2 \* a)) - u[1, k] \* mul0 / h) / \

(b1 + mul0 \* (- 1 / h - h / (2 \* a \* tau) + c \* h / (2 \* a)))

u[N-1, k] = (phil[k] - mull \* (- h \* u[N-1, k-1] / (2 \* a \* tau) - h \* f[N-1, k] / (2 \* a)) + u[N-2, k] \* mull / h) / \

(b2 + mull \* (1 / h + h / (2 \* a \* tau) - c \* h / (2 \* a)))

**elif** method == "implicit":

"""

Неявная схема

"""

**if** order == 1:

**for** k **in** range(1, K):

el = np.concatenate((np.ones(N - 2) \* (a / h \*\* 2 - b / (2 \* h)),

[-a2 / h]))

d = np.concatenate(([-a1 / h + b1],

np.ones(N - 2) \* (- 1 / tau - 2 \* a / h \*\* 2 + c),

[a2 / h + b2]))

eu = np.concatenate(([a1 / h],

np.ones(N - 2) \* (a / h \*\* 2 + b / (2 \* h))))

rb = np.concatenate(([phi0[k]],

-u[1:N-1, k-1] / tau - f[1:N-1, k],

[phil[k]]))

u[:, k] = prog\_solve(el, d, eu, rb)

**elif** order == 2:

**for** k **in** range(1, K):

mul0 = h \* a1 / (2 \* a + b \* h)

mull = - a2 \* h / (2 \* a - b \* h)

el = np.concatenate((np.ones(N - 2) \* (a / h \*\* 2 - b / (2 \* h)),

[(-1 / tau - 2 \* a / h \*\* 2 + c) \* mull - 4 \* a2 / (2 \* h)]))

d = np.concatenate(([(a / h \*\* 2 - b / (2 \* h)) \* mul0 - 3 \* a1 / (2 \* h) + b1],

np.ones(N - 2) \* (- 1 / tau - 2 \* a / h \*\* 2 + c),

[(a / h \*\* 2 + b / (2 \* h)) \* mull + 3 \* a2 / (2 \* h) + b2]))

eu = np.concatenate(([(- 1 / tau - 2 \* a / h \*\* 2 + c) \* mul0 + 4 \* a1 / (2 \* h)],

np.ones(N - 2) \* (a / h \*\* 2 + b / (2 \* h))))

rb = np.concatenate(([(-u[1, k-1] / tau - f[1, k]) \* mul0 + phi0[k]],

-u[1:N-1, k-1] / tau - f[1:N-1, k],

[(-u[N-2, k-1] / tau - f[N-2, k]) \* mull + phil[k]]))

u[:, k] = prog\_solve(el, d, eu, rb)

**elif** order == 3:

**for** k **in** range(1, K):

mul0 = 2 \* a1 \* a / (2 \* a - b \* h)

mull = 2 \* a2 \* a / (2 \* a + b \* h)

el = np.concatenate((np.ones(N - 2) \* (a / h \*\* 2 - b / (2 \* h)),

[- mull / h]))

d = np.concatenate(([b1 + mul0 \* (- 1 / h - h / (2 \* a \* tau) + c \* h / (2 \* a))],

np.ones(N - 2) \* (- 1 / tau - 2 \* a / h \*\* 2 + c),

[b2 + mull \* (1 / h + h / (2 \* a \* tau) - c \* h / (2 \* a))]))

eu = np.concatenate(([mul0 / h],

np.ones(N - 2) \* (a / h \*\* 2 + b / (2 \* h))))

rb = np.concatenate(([phi0[k] - mul0 \* (h \* u[0, k - 1] / (2 \* a \* tau) + h \* f[0, k] / (2 \* a))],

-u[1:N-1, k-1] / tau - f[1:N-1, k],

[phil[k] - mull \* (- h \* u[N-1, k-1] / (2 \* a \* tau) - h \* f[N-1, k] / (2 \* a))]))

u[:, k] = prog\_solve(el, d, eu, rb)

**elif** method == "comb":

"""

Смешанная схема

"""

**if** order == 1:

**for** k **in** range(1, K):

el = np.concatenate((np.ones(N - 2) \* (a \* tetta / h \*\* 2 - b \* tetta / (2 \* h)),

[-a2 / h]))

d = np.concatenate(([-a1 / h + b1],

np.ones(N - 2) \* (- 1 / tau - 2 \* a \* tetta / h \*\* 2 + c \* tetta),

[a2 / h + b2]))

eu = np.concatenate(([a1 / h],

np.ones(N - 2) \* (a \* tetta / h \*\* 2 + b \* tetta / (2 \* h))))

rb = np.concatenate(([phi0[k]],

-u[1:N-1, k-1] / tau - (1 - tetta) \* a \* (u[2:N, k-1] - 2 \* u[1:N-1, k-1] + u[0:N-2, k-1]) / h \*\* 2 - b \* (1 - tetta) \* (u[2:N, k-1] - u[0:N-2, k-1]) / (2 \* h) - c \* (1 - tetta) \* u[1:N-1, k-1] - tetta \* f[1:N-1, k] - (1 - tetta) \* f[1:N-1, k-1],

[phil[k]]))

u[:, k] = prog\_solve(el, d, eu, rb)

**elif** order == 2:

mul0 = h \* a1 / ((2 \* a + b \* h) \* tetta)

mull = - a2 \* h / ((2 \* a - b \* h) \* tetta)

**for** k **in** range(1, K):

el = np.concatenate((np.ones(N - 2) \* (a \* tetta / h \*\* 2 - b \* tetta / (2 \* h)),

[(- 1 / tau - 2 \* a \* tetta / h \*\* 2 + c \* tetta) \* mull - 4 \* a2 / (2 \* h)]))

d = np.concatenate(([(a \* tetta / h \*\* 2 - b \* tetta / (2 \* h)) \* mul0 - 3 \* a1 / (2 \* h) + b1],

np.ones(N - 2) \* (- 1 / tau - 2 \* a \* tetta / h \*\* 2 + c \* tetta),

[(a \* tetta / h \*\* 2 + b \* tetta / (2 \* h)) \* mull + 3 \* a2 / (2 \* h) + b2]))

eu = np.concatenate(([(- 1 / tau - 2 \* a \* tetta / h \*\* 2 + c \* tetta) \* mul0 + 4 \* a1 / (2 \* h)],

np.ones(N - 2) \* (a \* tetta / h \*\* 2 + b \* tetta / (2 \* h))))

rb = np.concatenate(([(-u[1, k-1] / tau - (1 - tetta) \* a \* (u[2, k-1] - 2 \* u[1, k-1] + u[0, k-1]) / h \*\* 2 - b \* (1 - tetta) \* (u[2, k-1] - u[0, k-1]) / (2 \* h) - c \* (1 - tetta) \* u[1, k-1] - tetta \* f[1, k] - (1 - tetta) \* f[1, k-1]) \* mul0 + phi0[k]],

-u[1:N-1, k-1] / tau - (1 - tetta) \* a \* (u[2:N, k-1] - 2 \* u[1:N-1, k-1] + u[0:N-2, k-1]) / h \*\* 2 - b \* (1 - tetta) \* (u[2:N, k-1] - u[0:N-2, k-1]) / (2 \* h) - c \* (1 - tetta) \* u[1:N-1, k-1] - tetta \* f[1:N-1, k] - (1 - tetta) \* f[1:N-1, k-1],

[(-u[N-2, k-1] / tau - (1 - tetta) \* a \* (u[N-1, k-1] - 2 \* u[N-2, k-1] + u[N-3, k-1]) / h \*\* 2 - b \* (1 - tetta) \* (u[N-1, k-1] - u[N-3, k-1]) / (2 \* h) - c \* (1 - tetta) \* u[N-2, k-1] - tetta \* f[N-2, k] - (1 - tetta) \* f[N-2, k-1]) \* mull + phil[k]]))

u[:, k] = prog\_solve(el, d, eu, rb)

**elif** order == 3:

mul0 = 2 \* a1 \* a / (2 \* a - b \* h)

mull = 2 \* a2 \* a / (2 \* a + b \* h)

**for** k **in** range(1, K):

el = np.concatenate((np.ones(N - 2) \* (a \* tetta / h \*\* 2 - b \* tetta / (2 \* h)),

[- mull / h]))

d = np.concatenate(([b1 + mul0 \* (- 1 / h - h / (2 \* a \* tau) + c \* h / (2 \* a))],

np.ones(N - 2) \* (- 1 / tau - 2 \* a \* tetta / h \*\* 2 + c \* tetta),

[b2 + mull \* (1 / h + h / (2 \* a \* tau) - c \* h / (2 \* a))]))

eu = np.concatenate(([mul0 / h],

np.ones(N - 2) \* (a \* tetta / h \*\* 2 + b \* tetta / (2 \* h))))

rb = np.concatenate(([phi0[k] - mul0 \* (h \* u[0, k - 1] / (2 \* a \* tau) + h \* f[0, k] / (2 \* a))],

-u[1:N-1, k-1] / tau - (1 - tetta) \* a \* (u[2:N, k-1] - 2 \* u[1:N-1, k-1] + u[0:N-2, k-1]) / h \*\* 2 - b \* (1 - tetta) \* (u[2:N, k-1] - u[0:N-2, k-1]) / (2 \* h) - c \* (1 - tetta) \* u[1:N-1, k-1] - tetta \* f[1:N-1, k] - (1 - tetta) \* f[1:N-1, k-1],

[phil[k] - mull \* (- h \* u[N-1, k-1] / (2 \* a \* tau) - h \* f[N-1, k] / (2 \* a))]))

u[:, k] = prog\_solve(el, d, eu, rb)

**return** x, t, u