Министерство образования Республики Беларусь

Учреждение образования «Белорусский Государственный Университет Информатики и Радиоэлектроники»

Кафедра «Информатика»

Лабораторная работа № 2

«Метод сеток решения одномерного нестационарного уравнения теплопроводности »

по курсу «Методы численного анализа»

**Выполнил:**

Волчецкий А.М. гр. 253504

**Проверила:**

Ероминек К.Р.

Минск 2015

**Задание**

Найти приближенное решение начально-краевой задачи для уравнения теплопроводности:

*,*

используя разностную схему. Взять h=(b-a)/10; шаг ῖ выбрать из условия устойчивости. Изобразить графики зависимости приближенного решения от х***.***

******



Теоретические сведения

Рассмотрим численные методы решения уравнений параболического типа.

Напомним постановку соответствующей смешанной задачи:

(1)

Здесь *a = a(x, t) > 0*. Для того чтобы задача была поставлена корректно, необходимо задать начальное условие

*u(0, x) = u0(x), t = 0,*

и граничные условия

Одномерное квазилинейное уравнение теплопроводности (диффузии):

Уравнения такого вида встречаются в теории горения, астрофизике, физике плазмы, динамике популяций и других приложениях. Здесь a(u) > 0 при любых значениях u, кроме того, . Для глобальной ограниченности решения также требуется выполнение условия

. Для корректной постановки задачи необходимо задать одно начальное и два граничных условия.

Двухмерное линейное уравнение теплопроводности (диффузии):

Для численного решения уравнения (1), по-видимому, наиболее известной является параметрическая двухслойная шести точечная разностная схема вида

, (4)

где

При имеем **явную схему**

, устойчивую при , с порядком аппроксимации

При имеем **неявную схему**

устойчивую при любых , с порядком аппроксимации

При разностный метод называется схемой Кранка-Никольсон:

Схема устойчива при любых шагах и имеет порядок аппроксимации Эта схема, в отличие от двух предыдущих, не является **монотонной**, т.е. она может давать осцилляции разностного происхождения на решениях, имеющих большие градиенты.

Рассмотрим одномерный однородный стержень длины *L*. Пусть *U(x,t)*− температура в точке стержня с абсциссой *x* в момент времени *t*. Из математической физики известно, что распределение температуры в точках стержня в зависимости от времени описывается дифференциальным уравнением

. (5)

Добавим к уравнению естественные граничные условия:

(6)

которые описывают температуру, измеряемую на концах стержня и начальное распределение температуры в точках стержня.

Построим сетку в области D (см. Рис. 1)

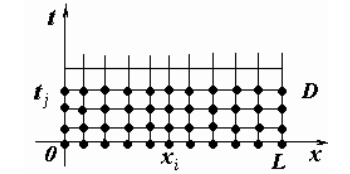


Рис. 1.

Заменим в дифференциальном уравнении значения частных производных во внутренних узлах их разностными аппроксимациями:

(7)

Получим разностное уравнение соответствующее исходному дифференциальному уравнению:

(8)

или

(9)

Положим

Тогда граничные условия аппроксимируются следующим образом

Зная значения на нижнем (нулевом) слое и на границе слева и справа, вычисляем для :

Таким образом, мы получили разностную краевую задачу (9)-(10), которая очевидно имеет решение, причем единственное.

Рассматривая уравнения (9), построим его шаблон.

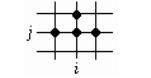


Рис. 2.

Вопрос о сходимости к решению дифференциальной задачи зависит от соотношения между и *h* в уравнении (5). Известно, что если

имеет место сходимость.

Рассмотрим подробнее ситуацию:

В этом случае уравнение (9) существенно упрощается и принимает вид:

На рис. 3 показан его шаблон

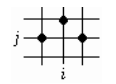


Рис. 3.

Пусть – функция двух переменных, график которой построен на основе аппроксимации плоскостями найденных при решении задачи (9)-(10)

значений . Можно показать, что при выполнении условия

будет сходиться к решению дифференциальной краевой задачи (5)-(6) и скорость сходимости будет:

Код программы

class Program:

def \_\_init\_\_(self, a, b, k, T, fi, g1, g2, f):

self.fi = fi

self.g1 = g1

self.g2 = g2

self.f = f

self.x, self.h = self.\_\_get\_x\_h(a, b)

self.t, self.taw = self.\_\_get\_t\_raw(a, b, k, T)

self.\_\_init\_table\_pattern()

def solve(self):

for index\_t in range(1, len(self.t)):

row = []

for index\_x in range(len(self.x)):

if (index\_x == 0):

row.append(self.\_\_get\_g1(self.t[index\_t]))

continue

if (index\_x == len(self.x)-1):

row.append(self.\_\_get\_g1(self.t[index\_t]))

continue

y = 0

y += self.taw/(self.h\*\*2)\*self.table\_pattern[index\_t-1][index\_x-1]

y += (1-2\*self.taw/(self.h\*\*2))\*self.table\_pattern[index\_t-1][index\_x]

y += self.taw/(self.h\*\*2)\*self.table\_pattern[index\_t-1][index\_x+1]

y += self.taw\*self.\_\_get\_f(self.x[index\_x], self.t[index\_t])

row.append(round(y, 5))

self.table\_pattern.append(row)

def \_\_init\_table\_pattern(self):

self.table\_pattern = [[0]\*len(self.x)]

self.table\_pattern[0][0] = self.\_\_get\_g1(self.t[0])

self.table\_pattern[0][-1] = self.\_\_get\_g2(self.t[-1])

for index in range(1, len(self.x)-1):

self.table\_pattern[0][index] = self.\_\_get\_fi(self.x[index])

def \_\_get\_f(self, x, t):

return eval(self.f)

def \_\_get\_fi(self, x):

return eval(self.fi)

def \_\_get\_g1(self, t):

return eval(self.g1)

def \_\_get\_g2(self, t):

return eval(self.g2)

def \_\_get\_x\_h(self, a, b):

h = (b-a)/10.0

value = a

x = []

while value<=b:

x.append(round(value, 5))

value += h

return x, h

def \_\_get\_t\_raw(self, a, b, k, T):

taw = ((b-a)/10.0)\*\*2/k/128

t = []

value = 0

while value <= T+2\*taw:

t.append(round(value, 5))

value += 2\*taw

return t, taw

class PlotHandler:

def \_\_init\_\_(self, x, rows):

self.x = x

self.rows = rows

def draw(self):

fig = plt.figure()

ax = plt.axes(xlim=(0, 2), ylim=(0, .2))

line, = ax.plot([], [], lw=5)

time\_text = ax.text(0.05, 0.9, '', transform=ax.transAxes)

def init():

line.set\_data([], [])

time\_text.set\_text('')

return line, time\_text

def animate(i):

i = i%len(self.rows)

x = self.x

y = self.rows[i]

line.set\_data(x, y)

time\_text.set\_text("Frame %d"%i)

return line, time\_text

anim = animation.FuncAnimation(fig, animate, init\_func=init, frames=len(self.rows), interval=1000, blit=True)

plt.show()

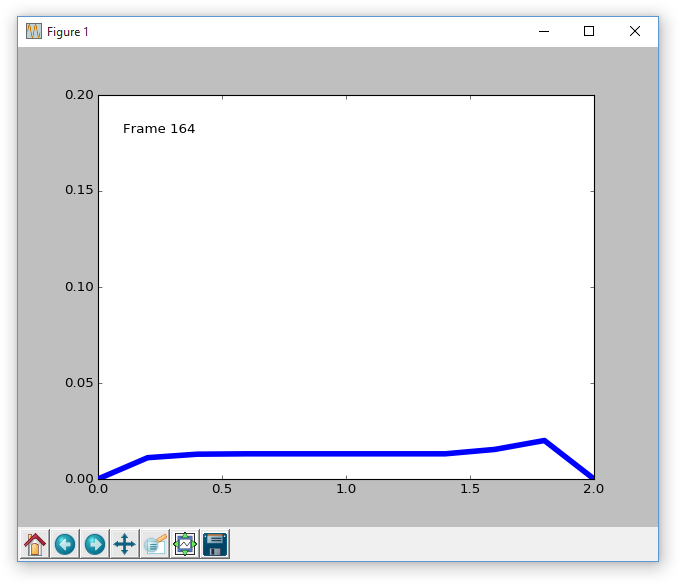
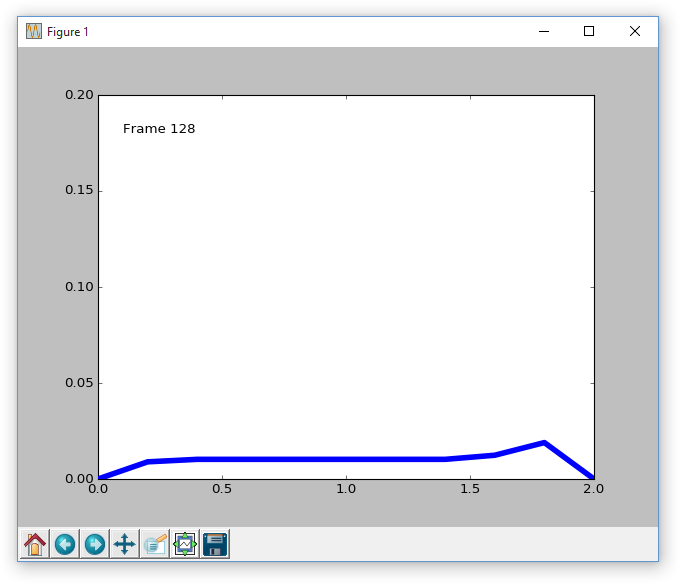
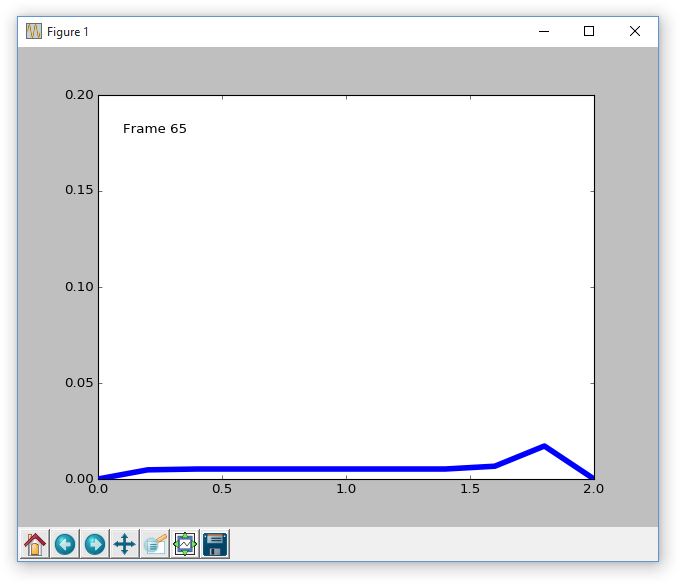
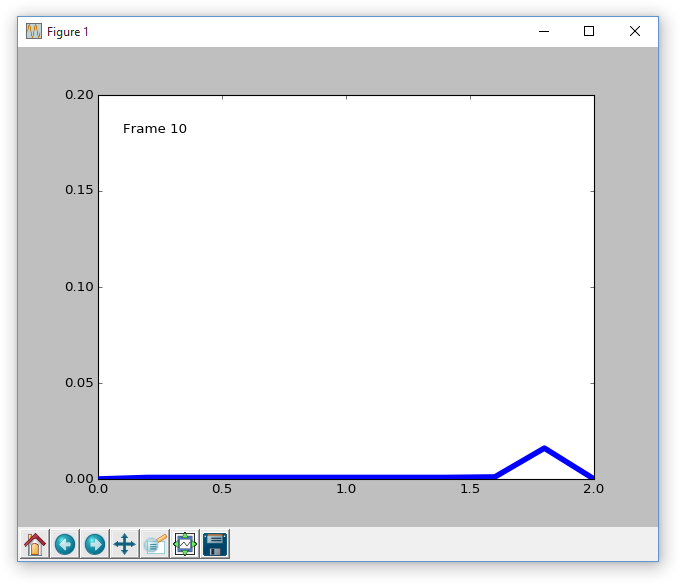
program = Program(-1, 1, 0.5, 0.4, "1+x\*\*2", "0", "0", "x")

program.solve()

plotHandler = PlotHandler(program.x, program.table\_pattern)

plotHandler.draw()

В результате выполнения программы была построена анимация, которая демонстрирует зависимость приближенного решения от х (температуру стержня в точках х) в различных промежутках времени. Численно программа выводит вектор приближенных решений в момент времени T.



**Вывод**

В данной работе была рассмотрена начально-краевая задача для уравнения теплопроводности и один из численных методов ее решения – метод сеток. Прежде всего, было дано понятие начально-краевой задачи для уравнения теплопроводности и выделены основные идеи метода сеток, приведен его алгоритм на языке программирования python и создана анимация демонстрирующая распределение тепла в стержне в различные моменты времени. Результаты показали, что значения температуры во взятой точке в разные промежутки времени значительно разнятся. Значительное влияние на результат оказали точность аппроксимации производных и шаг конечно-разностной сетки.