БГУИР

Кафедра информатики

ОТЧЕТ

по лабораторной работе № 4

Метод сеток решения одномерного

Аппроксимации граничных условий второго рода в методе конечных

разностей на примере уравнения теплопроводности

Выполнил: Проверил:

ст. гр. 553504 Гербик А. И.

Криницин А. В.

Минск 2017

ОГЛАВЛЕНИЕ

[1. Введение 3](#_Toc465628474)

[1.1. Постановка задачи 3](#_Toc465628475)

[1.2. Краткие теоретические сведения 3](#_Toc465628476)

2.  [Исходный код программы………….](#_Toc465628485) 5

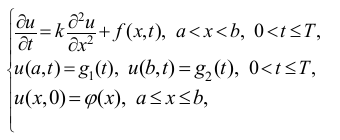
3[. Анализ полученных результатов](#_Toc465628485) 8

# 1.Введение

## 1.1 Постановка задачи

Найти приближенное решение начально-краевой задачи

для уравнения теплопроводности:



используя явную и неявную разностные схемы.

Взять h = ( b − a ) /10 ; шаг τ выбрать из условия устойчивости. Изобразить графики зависимости приближенного решения от x при t= 0 , 2 τ , 4 τ ,...T.

УКАЗАНИЕ. Условие устойчивости для явной разностной схемы имеет

вид τ ≤ 0.5( h 2 / k ) .

Вариант 13

a = 13

b = 0

k = 0.2

T = 0.25 f(x, t) = 1- x

## 1.2Краткие теоретические сведения

Задачи, которые будут использоваться для анализа свойств численных

решений с ГУ второго рода, формулируются так: в стержне длиной L с теплоизолированной боковой поверхностью торец x=0 поддерживается при

постоянной температуре T0 (ГУ первого рода), а торец x=1– теплоизолирован (ГУ второго рода); температуропроводность материала стержня постоянна и равна a; в начальный момент времени t=0 стержень нагрет до температуры Tнач(x) (координата x отсчитывается от левого торца стержня; см. рис.1). Найти распределение температуры по стержню в любой момент времени, т.е. найти функцию T(x,t) для 0<x≤L и t>0.

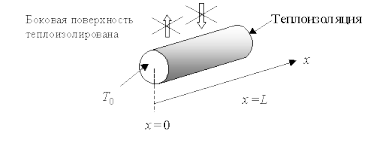


Рис. 1. Система координат и обозначения. (Стержень круглого сечения

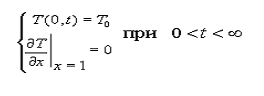
нарисован условно – сечение может иметь любую форму и если боковая поверхность теплоизолирована, то температура любой точки стержня может зависеть только от координаты x и не будет зависеть от координаты поперек стержня).

Искомая функция T(x,t) является решением одномерного уравнения

теплопроводности, которое в безразмерных координатах имеет вид:



Граничные условия:



на границе x=0 граничное условие первого рода, а при x=1 – второго).

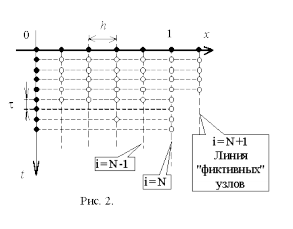
Начальные условия: T(x,0)=Tнач(x) при

*Способы реализации ГУ второго рода*

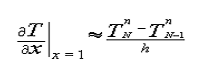
Методы конечных разностей, применяемые для численного решения задач с

граничными условиями второго (и третьего) рода, не имеют никаких

принципиальных отличий от методов, применяемых для задач с ГУ первого рода. Для решения поставленной задачи методом конечных разностей необходимо представить граничное условие второго рода в "естественном" для этого метода виде, т.е. с использованием численного решения (величин). Иными словами, производную в граничном условии надо заменить её разностной аппроксимацией, а это можно сделать многими способами.



Первый способ. Приближенно значение производной при x=1 можно записать, используя аппроксимацию производной по x левой разностью:



а поскольку значение этой производной в рассматриваемой задаче равно нулю, то аппроксимация граничного условия выглядит так:



# 2. Исходный код программы

# Явная разностная схема:

|  |
| --- |
| # coding: utf-8  # In[3]:  import numpy as np  import sympy as sp  import pandas as pd  # In[4]:  get\_ipython().magic(u'pylab inline')  # In[13]:  x = sp.Symbol('x')  t = sp.Symbol('t')  # In[14]:  T = 0.25  a = 0  b = 1  k = 0.2  s = sp.sin(x \*1.0)  #s(x) | u(x, 0) = s(x)  g1 = 0 \* t  g2 = sp.sin(1.0 + 2 \* t)  f = 1 - x  h = (b - a) / 10.0  tetta = (h \* h \* k) / 2  # In[15]:  columns = np.arange(a, b + h, h)  index = np.arange(0, T + tetta, tetta)  values = pd.DataFrame(columns=range(len(columns)), index=range(len(index)))  # In[22]:  values.iloc[0] = [s.subs(x, i) for i in columns]  values.iloc[:,0] = [g1.subs(t, i) for i in index]  values.iloc[:,-1] = [g2.subs(t, i) for i in index]  # In[25]:  values[len(columns)-1][0] = values[len(columns)-2][0]  for row\_pos in range(1, len(index)):  for col\_pos in range(1, len(columns) - 1):  values[col\_pos][row\_pos] = (  (tetta / (h \* h)) \* (values[col\_pos-1][row\_pos-1] + values[col\_pos+1][row\_pos-1]) +  (1 - 2 \* tetta / (h \* h)) \* (values[col\_pos][row\_pos-1]) +  tetta \* f.subs(x, columns[col\_pos])  )  values[len(columns)-1][row\_pos] = values[len(columns)-2][row\_pos]  # In[26]:  values.applymap(lambda x: round(x, 6)).head()  # In[27]:  x = np.arange(10)  fig = plt.figure()  ax = plt.subplot(111)  for i in np.linspace(0, len(index) - 1, 10):  line, = ax.plot(columns, values.iloc[int(i)], label='t={}'.format(index[int(i)]))  ax.legend(loc='best', bbox\_to\_anchor=(0.5, 1.05),  ncol=3, fancybox=True, shadow=True)  plt.show() |

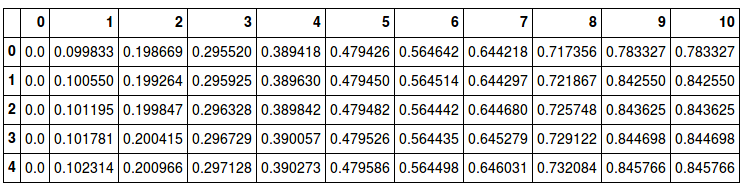
# Неявная разностная схема:

|  |
| --- |
| # coding: utf-8  # In[1]:  import numpy as np  import sympy as sp  import pandas as pd  # In[2]:  get\_ipython().magic(u'pylab inline')  # In[3]:  x = sp.Symbol('x')  t = sp.Symbol('t')  # In[6]:  T = 0.25  a = 0  b = 1  k = 0.2  s = sp.sin(x \*1.0)  #s(x) | u(x, 0) = s(x)  g1 = 0 \* t  g2 = sp.sin(1.0 + 2 \* t)  f = 1 - x  h = (b - a) / 10.0  tetta = (h \* h \* k) / 2  # In[7]:  columns = np.arange(a, b + h, h)  index = np.arange(0, T + tetta, tetta)  values = pd.DataFrame(columns=range(len(columns)), index=range(len(index)))  # In[8]:  values.iloc[0] = [s.subs(x, i) for i in columns]  values.iloc[:,0] = [g1.subs(t, i) for i in index]  values.iloc[:,-1] = [g2.subs(t, i) for i in index]  # In[9]:  values[len(columns)-1][0] = values[len(columns)-2][0]  for row\_pos in range(1, len(index)):  size = len(columns) - 1  a\_ = [tetta / h / h] \* (size -2) + [0] + [0]  c\_ = [0]+ [tetta / h / h] \* (size-2) + [0]  b\_ = [1]  d\_ = [values[0][row\_pos]]  for col\_pos in range(1, len(columns) - 2):  b\_.append( -(1 + 2 \* tetta / h / h ))  d\_.append(-values[col\_pos][row\_pos-1] - tetta \* f.subs(x, columns[col\_pos]))  b\_.append(1)  d\_.append(values[len(columns)-1][row\_pos])  res = TDMA(a\_, b\_, c\_, d\_)  res = np.append(res, res[-1])  values.iloc[row\_pos] = res  # In[10]:  values.applymap(lambda x: round(x, 6)).head()  # In[11]:  x = np.arange(10)  fig = plt.figure()  ax = plt.subplot(111)  for i in np.linspace(0, len(index) - 1, 10):  line, = ax.plot(columns, values.iloc[int(i)], label='t={}'.format(index[int(i)]))  ax.legend(loc='best', bbox\_to\_anchor=(0.5, 1.05),  ncol=3, fancybox=True, shadow=True)  plt.show()  # In[5]:  def TDMA(a,b,c,d):  n = len(d)  w= np.zeros(n-1,float)  g= np.zeros(n, float)  p = np.zeros(n,float)  w[0] = c[0]/b[0]  g[0] = d[0]/b[0]  for i in range(1,n-1):  w[i] = c[i]/(b[i] - a[i-1]\*w[i-1])  for i in range(1,n):  g[i] = (d[i] - a[i-1]\*g[i-1])/(b[i] - a[i-1]\*w[i-1])  p[n-1] = g[n-1]  for i in range(n-1,0,-1):  p[i-1] = g[i-1] - w[i-1]\*p[i]  return p |

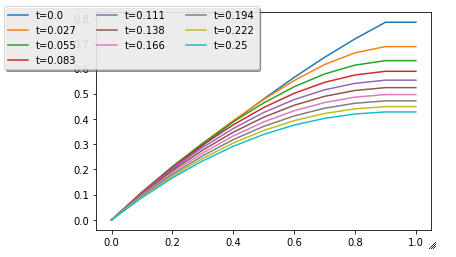
# 3. Анализ полученных результатов

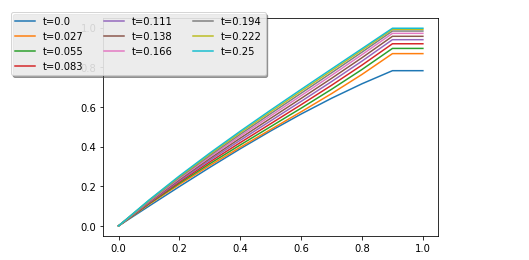
## В результате работы алгоритма получена таблица приближений функции f(x, t) а заданных диапазонах изменения t и х:

## Для явной схемы:

 Для неявной схемы:

Графики поведения f(x ,t) при фиксированном значении t:

Для явной схемы:



Для неявной схемы: