БГУИР

Кафедра информатики

ОТЧЕТ

по лабораторной работе № 6

Метод сеток решения волнового уравнения.

Выполнил: Проверил:

ст. гр. 553504 Гербик А. И.

Криницин А. В.

Минск 2017

ОГЛАВЛЕНИЕ

[1. Введение 3](#_Toc465628474)

[1.1. Постановка задачи 3](#_Toc465628475)

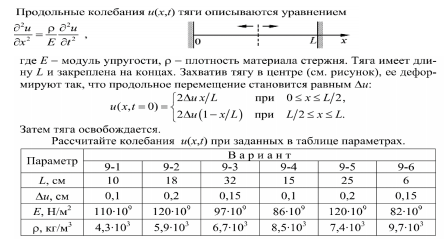
[1.2. Краткие теоретические сведения 3](#_Toc465628476)

2.  [Исходный код программы………….](#_Toc465628485) 6

3[. Анализ полученных результатов](#_Toc465628485) 7

# 1.Введение

## Постановка задачи



## 

## Краткие теоретические сведения

Волновое уравнение

Рассмотрим смешанную задачу для волнового урав­нения:

|  |  |  |
| --- | --- | --- |
|  | = *f*(*x*, *t*), 0 < *x* < 1, 0 < *t T*, | (1) |

|  |  |  |
| --- | --- | --- |
|  | *u*(*x*,0) =, =*q*(*x*), 0 *x*  1, | (2) |

|  |  |  |
| --- | --- | --- |
|  | *u*(0, *t*) = 0, *u*(1, *t*) = 0, 0 *t*  *T*, | (3) |

где *f*(*x*,*t*), , *q*(*x*) — заданные достаточно гладкие функции, причем (0) =(1) = *q*(0) = *q*(1) =0.

Будем предполагать, что задача (1) — (3) имеет реше­ние *u(х*, *t*) (), ={(*x*, *t*): 0*x*1, 0*tT*} — замкнутый прямоуголь­ник. Это решение единственно.

Разностная схема.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | ◦ |  |
|  | ◦ | ◦ | ◦ |
|  |  | ◦ |  |
|  |  |  |  |
|  |  |  |  |
|  | Рис. 25. | |  |

Будем использовать сетки, построенные на замкнутом прямоугольнике в § 31, и соответствующие обозначения сеточных функций. За­меняем в уравнении (1) частную производную приближенно второй разностной производной в направлении *t*, частную производную — аппроксимируем с помощью разностного оператора (31.5) и, переобозна­чив *u* на *у*, приходим к разностному уравнению

|  |  |  |
| --- | --- | --- |
|  | =, | (4) |

*k* = 1, 2, . . . , *N*-l, =1, 2, . . . , *M*-l.

Шаблон разностного уравнения (4) показан на рис. 25. Это уравнение можно разрешить явно относи­тельно . Но для того, чтобы находить значения разностного решения на -м слое, требуется иметь уже вычисленные значения искомого решения на двух предыдущих слоях. Поэтому нужно получить разностное решение сначала отдельно на слоях, отвечающих значениям = 0 и = l. В этом нам помогут началь­ные условия (2).

Прежде всего, используя первое начальное условие (2), задаем

|  |  |  |
| --- | --- | --- |
|  | =, *k* = 1, 2, . . . , *N*-l. | (5) |

Кроме того, полагаем при *k* = 1, 2, . . . , *N*-l

|  |  |  |
| --- | --- | --- |
|  | =++(). | (6) |

Правая часть формулы (6) аппроксимирует многочлен Тейлора ++, поскольку согласно (2) =, =, а из уравне­ния (1) для частных производных решения задачи (1) — (3) вытекает связь =+. Для аппроксимации = исполь­зуется оператор (31.5).

Наконец, согласно краевым условиям (3) имеем

|  |  |  |
| --- | --- | --- |
|  | , , = 0, 1, . . . , *M*. | (7) |

Теперь разностная схема (4) — (7) полностью опре­делена. Эта схема явная трехслойная (см. шаблон на рис. 25), условно устойчивая в некоторых естественных нормах.

Если *h*0, 0, причем /*hc*<1, *c* = const, то решение у разностной схемы (4) — (7) сходится к рас­сматриваемому решению *u* задачи (1) — (3) в следующем смысле:

|  |  |  |
| --- | --- | --- |
|  | =. | (8) |

где =. Схема (4) — (7) имеет второй порядок точности и по *h*, и по .

Понятие о методе прямых. Если в задаче (1) — (3) ввести дискретность только по *х*, то мы придем к системе линейных обыкновенных дифференциальных уравнений

|  |  |  |
| --- | --- | --- |
|  | =, | (9) |

*k* = 1, 2, . . . , *N*-l,

с начальными условиями

|  |  |  |
| --- | --- | --- |
|  | , , | (10) |

причем .

При сделанном предположении относительно глад­кости решения и задачи (1) — (3) имеем

|  |  |  |
| --- | --- | --- |
|  | =, | (11) |

где =, , , . . . , — решение задачи Коши (9), (10).

Данный метод называется *методом прямых*, по­скольку приближенное решение задачи (1) — (3) ищется на прямых , *k* = 1, 2, . . . , *N*-l, расположенных в плоскости *xt*.

Разностный же метод часто называется *методом сеток*.

# Исходный код программы

|  |
| --- |
| # 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')  y = sp.Symbol('y')  # In[175]:  T = 0.25  A = 1  B = 1  h = 0.01  # In[176]:  columns = np.arange(0, A + h, h)  index = np.arange(0, B + h, h)  values = np.zeros((len(columns), len(index)))  # In[177]:  for row\_pos in range(len(index)):  for col\_pos in range(len(columns)):  values[row\_pos][col\_pos] = 0    for row\_pos in range(len(index)):  values[row\_pos][len(columns)-1] = 0.85    for col\_pos in range(1, len(index) - 1):  values[0][col\_pos] = sin(columns[col\_pos])  # In[178]:  for col\_pos in range(1, len(index) - 1):  values[1][col\_pos] = (values[0][col\_pos] +  gamma/2 \* (values[0][col\_pos-1] - 2 \*values[0][col\_pos] + values[0][col\_pos+1]))  # In[179]:  for row\_pos in range(2, len(index) – 1):  for col\_pos in range(1, len(columns) – 1):  lamd = (values[row\_pos-1][col\_pos-1] - 2 \* values[row\_pos-1][col\_pos] + values[row\_pos-1][col\_pos+1]) / (h\*h)  values[row\_pos][col\_pos] = h\*h\*(0.5-lamd) - 2\*values[row\_pos-1][col\_pos]-values[row\_pos-2][col\_pos]    # In[180]:  pd.DataFrame(values).applymap(lambda x: round(x, 2))[:10]  # In[181]:  from IPython.html.widgets import \*  get\_ipython().magic(u'matplotlib inline')  x=index  def pltsin(f):  pylab.axis([0, 1,-100,100])  plot(x,values[f])  interact(pltsin, f=(0,100,1)); |

# Анализ полученных результатов

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

## 

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

