**Московский авиационный институт**

**(национальный исследовательский университет)**

**Институт информационных технологий и прикладной математики**

**Кафедра вычислительной математики и программирования**

Лабораторная работа №5

«Численное решение уравнений с частными производными параболического типа»

Вариант №2

Студент: Ганцева Е. С.

Группа: М8О-409Б-19

Руководитель: Пивоваров Д.Е.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 12.01.2023

**Москва 2023**

**Лабораторная работа №5**

Численное решение уравнений с частными производными параболического типа. Понятие о методе конечных разностей. Конечно-разностные схемы.

**Задача**

Используя явную и неявную конечно-разностные схемы, а также схему Кранка - Николсона, решить начально-краевую задачу для дифференциального уравнения параболического типа. Осуществить реализацию трех вариантов аппроксимации граничных условий, содержащих производные: двухточечная аппроксимация с первым порядком, трехточечная аппроксимация со вторым порядком, двухточечная аппроксимация со вторым порядком. В различные моменты времени вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением U(x, t) . Исследовать зависимость погрешности от сеточных параметров τ, h.

**Описание метода**

Рассматривается решение уравнения теплопроводности с граничными условиями второго рода на обоих концах интервала, т.е. рассматривается следующая задача:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание



Для решения такой задачи применяют метод конченых разностей. Для этого вводят понятие разностной сетки



с пространственным шагом h и шагом по времени τ.

Введём понятие сеточной функции. Сеточной функцией называют следующее отображение целых аргументов j и k:

Затем происходит аппроксимация производной по времени и второй производной по пространству.

Если аппроксимировать вторую производную по пространству на нижнем временном слое, то получим **явную** конечно-разностную схему.

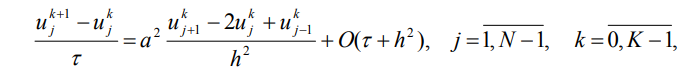
Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Явная конечно-разностная схема:



где для каждого j-го уравнения неизвестна только одна величина , которая может быть явно выражена из этого уравнения:



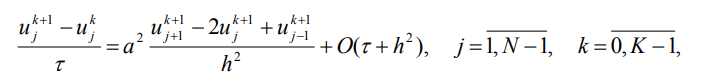
Для устойчивости данной схемы накладывается ограничение на σ: σ <= 1/2.

Если аппроксимировать производную по пространству на верхнем временном слое, то получим **неявную** конечно-разностную схему

Изображение выглядит как текст, антенна

Автоматически созданное описание

Неявная схема:



где для нахождения сеточной функции на верхнем временном слое необходимо решать СЛАУ с трёхдиагональной матрицей:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

**Явно-неявная** конечно-разностная схема имеет следующий вид:

Изображение выглядит как текст, антенна

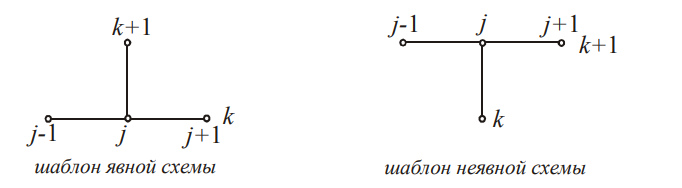
Автоматически созданное описание

где 0ϴ1.

При ϴ = 0 получается явная схема, при ϴ = 1 – неявная, при ϴ = 1/2 - схема Кранка-Николсона.

Здесь так же, как и в неявной схеме для нахождения на каждом шаге необходимо решать СЛАУ с трёхдиагональной матрицей.

Шаблоны трёх схем:



Шаблон схемы Кранка-Николсона:

Изображение выглядит как часы, датчик

Автоматически созданное описание

**Аппроксимация граничных условий**

Рассмотрим 3 способа аппроксимации граничных условий 2Т1П, 3Т2П, 2Т2П.

Двухточечная, с первым порядком:



Трёхточечная, со вторым порядком:

Изображение выглядит как текст

Автоматически созданное описание

При использовании неявной и явно-неявной схемы СЛАУ теряет трёхдиагональность, поэтому сначала необходимо привести СЛАУ к трёхдиагональному виду линейной комбинацией первой строки со второй, предпоследней с последней.

Двухточечная, со вторым порядком:

Для того, чтобы получить эти формулы для начала раскладывают в ряд Тейлора в окрестности x = 0 и в окрестности x = l:

Изображение выглядит как текст

Автоматически созданное описание

Далее берут информацию из исходного уравнения, выражая оттуда вторую производную и подставляя это выражение, получают:

Изображение выглядит как текст, стол

Автоматически созданное описание

**Вариант**

Изображение выглядит как текст

Автоматически созданное описание

**Выводы**

В данной работе реализована явная и неявная конечно-разностные схемы, а также схема Кранка – Николсона для решения начально-краевую задачу для дифференциального уравнения параболического типа. Осуществлена реализацию трех вариантов аппроксимации граничных условий, содержащих производные: двухточечная аппроксимация с первым порядком, трехточечная аппроксимация со вторым порядком, двухточечная аппроксимация со вторым порядком.

**Приложение. Листинг программы.**

|  |
| --- |
| #!/usr/bin/env python |
|  | # coding: utf-8 |
|  |  |
|  | # In[5]: |
|  |  |
|  |  |
|  | import numpy as np |
|  | import matplotlib.pyplot as plt |
|  | import math |
|  |  |
|  | # %matplotlib notebook |
|  | get\_ipython().run\_line\_magic('matplotlib', 'inline') |
|  |  |
|  |  |
|  | # In[6]: |
|  |  |
|  |  |
|  | def Progonka(arr): |
|  | A = [[arr[i][j] for j in range(len(arr[i]))] for i in range(len(arr))] |
|  | n = len(A) |
|  |  |
|  | a, b, c, d = [0], [arr[0][0]], [arr[0][1]], [arr[0][2]] |
|  | for i in arr[1:-1]: |
|  | a.append(i[0]) |
|  | b.append(i[1]) |
|  | c.append(i[2]) |
|  | d.append(i[3]) |
|  |  |
|  | a.append(arr[-1][0]) |
|  | b.append(arr[-1][1]) |
|  | c.append(0) |
|  | d.append(arr[-1][2]) |
|  |  |
|  | for i in range(n): |
|  | if math.fabs(b[i]) < math.fabs(a[i]) + math.fabs(c[i]): |
|  | raise Exception |
|  |  |
|  | # Формирование массивов P, Q (Расчет значений) ((Прямой ход)) |
|  |  |
|  | P, Q = [-c[0] / b[0]], [d[0] / b[0]] |
|  |  |
|  | for i in range(1, n): |
|  | P.append(-c[i] / (b[i] + a[i] \* P[i - 1])) |
|  | Q.append((d[i] - a[i] \* Q[i - 1]) / (b[i] + a[i] \* P[i - 1])) |
|  |  |
|  | # Вычисление решения системы (Обратный ход) |
|  | x = [Q[n - 1]] |
|  | for i in range(1, n): |
|  | x.append(P[n - 1 - i] \* x[i - 1] + Q[n - 1 - i]) |
|  |  |
|  | x = reversed(x) |
|  | return x |
|  |  |
|  |  |
|  | # In[7]: |
|  |  |
|  |  |
|  | a = 0.01 |
|  |  |
|  |  |
|  | # In[10]: |
|  |  |
|  |  |
|  | N = 100 |
|  | T\_end = 3 |
|  | X\_end = math.pi |
|  | X = np.linspace(0, X\_end, N) |
|  | T = np.linspace(0, T\_end, N) |
|  | u = U(X, T) |
|  |  |
|  | h = X\_end / N |
|  | tau = T\_end / N |
|  | sigma = a\*a\*tau/h/h |
|  | sigma |
|  |  |
|  |  |
|  | # In[14]: |
|  |  |
|  |  |
|  | U = lambda x, t: x + np.exp(-math.pi\*math.pi\*a\*t)\*np.sin(math.pi\*x) |
|  |  |
|  | phi\_0 = lambda t: 0 |
|  | phi\_l = lambda t: 1 |
|  | xi = lambda x: x + math.sin(math.pi\*x) |
|  |  |
|  |  |
|  | # аналитическое решение |
|  |  |
|  | # In[15]: |
|  |  |
|  |  |
|  | u2 = np.array([[U(i, j) for j in T] for i in X]) |
|  |  |
|  | fig = plt.figure() |
|  | ax = fig.add\_subplot(111, projection='3d') |
|  |  |
|  | Q, W = np.meshgrid(X, T) |
|  | ax.plot\_surface(W, Q, np.array(u2)) |
|  |  |
|  | ax.set\_xlabel('T Label') |
|  | ax.set\_ylabel('X Label') |
|  | ax.set\_zlabel('u Label') |
|  | None |
|  |  |
|  |  |
|  | # явная конечноразностная схема |
|  |  |
|  | # In[18]: |
|  |  |
|  |  |
|  | u = [[0 for j in range(len(T))] for i in range(len(X))] |
|  |  |
|  | for j in range(len(X)): |
|  | u[j][0] = xi(X[j]) |
|  |  |
|  | for k in range(len(T) - 1): |
|  | u[0][k+1] = phi\_0(T[k+1]) |
|  |  |
|  | for k in range(len(T) - 1): |
|  | u[-1][k+1] = phi\_l(T[k+1]) |
|  |  |
|  | for j in range(1, len(X) - 1): |
|  | for k in range(len(T) - 1): |
|  | u[j][k+1] = sigma\*u[j+1][k] + (1 - 2\*sigma)\*u[j][k] + sigma\*u[j-1][k] |
|  |  |
|  | fig = plt.figure() |
|  | ax = fig.add\_subplot(111, projection='3d') |
|  |  |
|  | Q, W = np.meshgrid(X, T) |
|  | ax.plot\_surface(W, Q, np.array(u)) |
|  |  |
|  | ax.set\_xlabel('t Label') |
|  | ax.set\_ylabel('x Label') |
|  | ax.set\_zlabel('u Label') |
|  |  |
|  |  |
|  | # Неявная схема |
|  |  |
|  | # In[19]: |
|  |  |
|  |  |
|  | u = [[0 for j in range(len(T))] for i in range(len(X))] |
|  |  |
|  | for k in range(len(T) - 1): |
|  | u[0][k+1] = phi\_0(T[k+1]) |
|  |  |
|  | for k in range(len(T) - 1): |
|  | u[-1][k+1] = phi\_l(T[k+1]) |
|  |  |
|  | for j in range(len(X)): |
|  | u[j][0] = xi(X[j]) |
|  |  |
|  | for k in range(len(T) - 1): |
|  | A = [sigma if i != 0 else 0 for i in range(N-2)] |
|  | B = [-(1 + 2\*sigma) for i in range(N-2)] |
|  | C = [sigma if i != N-3 else 0 for i in range(N-2)] |
|  | D = [-u[j][k] for j in range(2, N-2)] |
|  | D.insert(0, -(u[1][k] + sigma\*phi\_0(T[k+1]))) |
|  | D.append(-(u[N-1][k] + sigma\*phi\_l(T[k+1]))) |
|  |  |
|  | P, Q = [-C[0] / B[0]], [D[0] / B[0]] |
|  | n = len(A) |
|  | # print(len(A), len(B), len(C), len(D)) |
|  |  |
|  | for i in range(1, n): |
|  | P.append(-C[i] / (B[i] + A[i] \* P[i - 1])) |
|  | Q.append((D[i] - A[i] \* Q[i - 1]) / (B[i] + A[i] \* P[i - 1])) |
|  |  |
|  | uk = [Q[n - 1]] |
|  |  |
|  | for i in range(1, n): |
|  | uk.append(P[n - 1 - i] \* uk[i - 1] + Q[n - 1 - i]) |
|  |  |
|  | uk = list(reversed(uk)) |
|  | # print(uk) |
|  |  |
|  | for j in range(1, N-1): |
|  | u[j][k+1] = uk[j-1] |
|  |  |
|  | # np.array(u) |
|  | fig = plt.figure() |
|  | ax = fig.add\_subplot(111, projection='3d') |
|  |  |
|  | Q, W = np.meshgrid(X, T) |
|  | ax.plot\_surface(W, Q, np.array(u)) |
|  |  |
|  | ax.set\_xlabel('x Label') |
|  | ax.set\_ylabel('t Label') |
|  | ax.set\_zlabel('u Label') |
|  |  |
|  |  |
|  | # Явно-неявная схема |
|  |  |
|  | # In[20]: |
|  |  |
|  |  |
|  | teta = 1/2 |
|  |  |
|  |  |
|  | # In[21]: |
|  |  |
|  |  |
|  | u = [[0 for j in range(len(T))] for i in range(len(X))] |
|  |  |
|  | for k in range(len(T) - 1): |
|  | u[0][k+1] = phi\_0(T[k+1]) |
|  |  |
|  | for k in range(len(T) - 1): |
|  | u[-1][k+1] = phi\_l(T[k+1]) |
|  |  |
|  | for j in range(len(X)): |
|  | u[j][0] = xi(X[j]) |
|  |  |
|  | for k in range(len(T) - 1): |
|  | A = [teta\*sigma if i != 0 else 0 for i in range(N-2)] |
|  | B = [-(1 + 2\*teta\*sigma) for i in range(N-2)] |
|  | C = [teta\*sigma if i != N-3 else 0 for i in range(N-2)] |
|  | D = [- (1-teta)\*sigma\*u[j+1][k] - (1 - 2\*(1-teta)\*sigma)\*u[j][k] - (1-teta)\*sigma\*u[j-1][k] for j in range(2, N-2)] |
|  | D.insert(0, -(teta\*sigma\*phi\_0(T[k+1]) + (1-teta)\*sigma\*u[2][k] + (1 - 2\*(1-teta)\*sigma)\*u[1][k] + (1-teta)\*sigma\*phi\_0(T[k]))) |
|  | D.append(-(teta\*sigma\*phi\_l(T[k+1]) + (1-teta)\*sigma\*phi\_l(T[k]) + (1 - 2\*(1-teta)\*sigma)\*u[N-1][k] + (1-teta)\*sigma\*u[N-2][k])) |
|  |  |
|  | P, Q = [-C[0] / B[0]], [D[0] / B[0]] |
|  | n = len(A) |
|  | # print(len(A), len(B), len(C), len(D)) |
|  |  |
|  | for i in range(1, n): |
|  | P.append(-C[i] / (B[i] + A[i] \* P[i - 1])) |
|  | Q.append((D[i] - A[i] \* Q[i - 1]) / (B[i] + A[i] \* P[i - 1])) |
|  |  |
|  | uk = [Q[n - 1]] |
|  |  |
|  | for i in range(1, n): |
|  | uk.append(P[n - 1 - i] \* uk[i - 1] + Q[n - 1 - i]) |
|  |  |
|  | uk = list(reversed(uk)) |
|  | # print(uk) |
|  |  |
|  | for j in range(1, N-1): |
|  | u[j][k+1] = uk[j-1] |
|  |  |
|  | # np.array(u) |
|  | fig = plt.figure() |
|  | ax = fig.add\_subplot(111, projection='3d') |
|  |  |
|  | Q, W = np.meshgrid(X, T) |
|  | ax.plot\_surface(W, Q, np.array(u)) |
|  |  |
|  | ax.set\_xlabel('x Label') |
|  | ax.set\_ylabel('t Label') |
|  | ax.set\_zlabel('u Label') |
|  |  |
|  |  |
|  | # Граничные условия третьего рода: |
|  |  |
|  | # In[22]: |
|  |  |
|  |  |
|  | alpha = 0 |
|  | beta = 1 |
|  | gamma = 0 |
|  | delta = 1 |
|  |  |
|  |  |
|  | # In[23]: |
|  |  |
|  |  |
|  | u = [[0 for j in range(len(T))] for i in range(len(X))] |
|  |  |
|  | for j in range(len(X)): |
|  | u[j][0] = xi(X[j]) |
|  |  |
|  | for j in range(1, len(X) - 1): |
|  | for k in range(len(T) - 1): |
|  | u[j][k+1] = sigma\*u[j+1][k] + (1 - 2\*sigma)\*u[j][k] + sigma\*u[j-1][k] |
|  |  |
|  | for k in range(len(T) - 1): |
|  | u[0][k+1] = -(alpha/h) / (beta - alpha/h) \* u[1][k+1] + phi\_0(T[k+1]) / (beta - alpha/h) |
|  |  |
|  | for k in range(len(T) - 1): |
|  | u[-1][k+1] = -(gamma/h) / (delta - gamma/h) \* u[-2][k+1] + phi\_l(T[k+1]) / (delta - gamma/h) |
|  |  |
|  |  |
|  | fig = plt.figure() |
|  | ax = fig.add\_subplot(111, projection='3d') |
|  |  |
|  | Q, W = np.meshgrid(X, T) |
|  | ax.plot\_surface(W, Q, np.array(u)) |
|  |  |
|  | ax.set\_xlabel('t Label') |
|  | ax.set\_ylabel('x Label') |
|  | ax.set\_zlabel('u Label') |