Московский Авиационный Институт (Национальный исследовательский университет)

Лабораторная работа №8

По курсу «Численные методы»

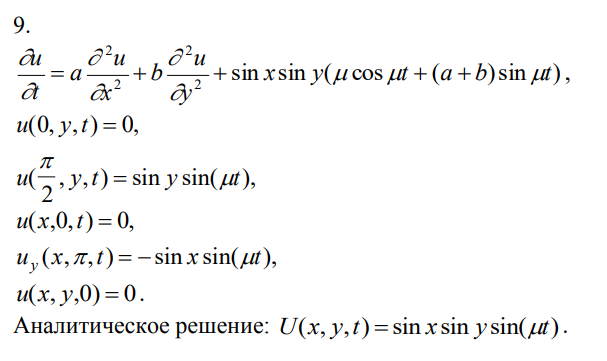
|  |  |
| --- | --- |
| Студент: | Сайгакова А.А. |
| Группа: | М8О-409Б-19 |
| Преподаватель: | Пивоваров Д. Е. |

Москва, 2022

**Задание:**

Используя схемы переменных направлений и дробных шагов, решить двумерную начально-краевую задачу для дифференциального уравнения параболического типа. 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, hx , hy

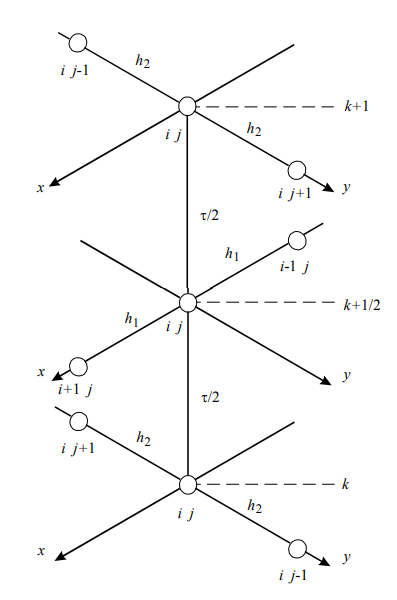
**Вариант:**

****

**Теория:**

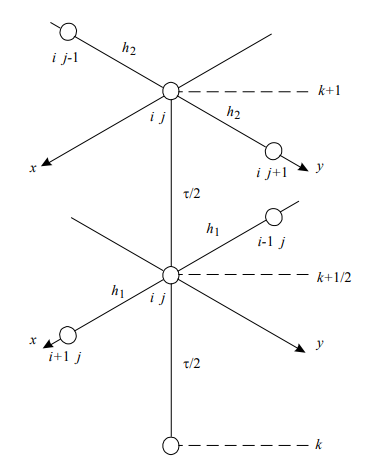
В схеме метода переменных направлений (МПН), как и во всех методах расщепления, шаг по времени τ разбивается на число независимых пространственных переменных (в двумерном случае - на два ). На каждом дробном временном слое один из пространственных дифференциальных операторов аппроксимируется неявно (по соответствующему координатному направлению осуществляются скалярные прогонки ), а остальные явно. На следующем дробном шаге следующий по порядку дифференциальный оператор аппроксимируется неявно, а остальные – явно и т.д

Схема МНП имеет вид:



В отличие от МПН метод дробных шагов (МДШ) использует только неявные конечноразностные операторы, что делает его абсолютно устойчивым в задачах, не содержащих смешанные производные. Он обладает довольно значительным запасом устойчивости и в задачах со смешанными производными

МДШ схема имеет вид:

****

**Код программы:**

def mpn(n):

u = np.zeros((n,n,n))

for i in range(n):

for j in range(n):

u[i][j][0] = 0

for k in range(1,n):

u1 = np.zeros((n,n))

u2 = np.zeros((n,n))

tau2 = t[k-1] + tau/2

for i in range(n):

u1[i][0] = 0

u1[0][i] = 0

u1[-1][i] = np.sin(y[i])\*np.sin(mu\*tau2)

u2[i][0] = 0

for j in range(n-1):

a = np.zeros(n)

b = np.zeros(n)

c = np.zeros(n)

d = np.zeros(n)

a[0] = 0

b[0] = 1

c[0] = 0

d[0] = 0

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

a[i] = acoef/h1/h1\*tau/2

b[i] = -(2/tau + 2\*acoef/h1/h1)\*tau/2

c[i] = acoef/h1/h1\*tau/2

d[i] = -(2\*u[i][j][k-1]/tau + bcoef/h2/h2\*(u[i][j+1][k-1]-2\*u[i][j][k-1]+u[i][j-1][k-1])-f(x[i],y[j],tau2))\*tau/2

a[-1] = 0

b[-1] = 1

c[-1] = 0

d[-1] = np.sin(y[j])\*np.sin(mu\*tau2)

result = slau([a,b,c,d])

#print(result)

for i in range(n):

u1[i][j] = result[i]

u1[i][-1] = -np.sin(x[i])\*np.sin(mu\*tau2)\*h1 + u1[i][-2]

#############

for i in range(n-1):

a = np.zeros(n)

b = np.zeros(n)

c = np.zeros(n)

d = np.zeros(n)

a[0] = 0

b[0] = 1

c[0] = 0

d[0] = 0

for j in range(1,n-1):

a[j] = bcoef/h2/h2\*tau/2

b[j] = -(2/tau + 2\*bcoef/h2/h2)\*tau/2

c[j] = bcoef/h2/h2\*tau/2

d[j] = -(u1[i][j]/tau/2 + acoef/h1/h1\*(u1[i+1][j]-2\*u1[i][j]+u1[i-1][j]) + f(x[i],y[j],tau2))\*tau/2

a[-1] = -1/h2

b[-1] = 1/h2

c[-1] = 0

d[-1] = -np.sin(x[i])\*np.sin(mu\*t[k])

result2 = slau([a,b,c,d])

#print(result2)

for j in range(n):

u2[i][j] = result2[j]

u2[0][j] = 0

u2[-1][j] = np.sin(y[j])\*np.sin(t[k])

for i in range(n):

u2[i][-1] = -np.sin(x[i])\*np.sin(mu\*t[k])\*h2 + u2[i][-2]

for i in range(n):

for j in range(n):

u[i][j][k] = u2[i][j]

return u

def mdsh(n):

u = np.zeros((n,n,n))

for i in range(n):

for j in range(n):

u[i][j][0] = 0

for k in range(1,n):

u1 = np.zeros((n,n))

u2 = np.zeros((n,n))

tau2 = t[k-1] + tau/2

for i in range(n):

u1[i][0] = 0

u1[0][i] = 0

u1[-1][i] = np.sin(y[i])\*np.sin(mu\*tau2)

u2[i][0] = 0

for j in range(n-1):

a = np.zeros(n)

b = np.zeros(n)

c = np.zeros(n)

d = np.zeros(n)

a[0] = 0

b[0] = 1

c[0] = 0

d[0] = 0

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

a[i] = acoef/h1/h1

b[i] = -(1/tau + 2\*acoef/h1/h1)

c[i] = acoef/h1/h1

d[i] = -(f(x[i],y[j],t[k-1])/2+u[i][j][k-1]/tau)

a[-1] = 0

b[-1] = 1

c[-1] = 0

d[-1] = np.sin(y[j])\*np.sin(mu\*tau2)

result = slau([a,b,c,d])

#print(result)

for i in range(n):

u1[i][j] = result[i]

u1[i][-1] = -np.sin(x[i])\*np.sin(mu\*tau2)\*h1 + u1[i][-2]

#############

for i in range(n-1):

a = np.zeros(n)

b = np.zeros(n)

c = np.zeros(n)

d = np.zeros(n)

a[0] = 0

b[0] = 1

c[0] = 0

d[0] = 0

for j in range(1,n-1):

a[j] = bcoef/h2/h2

b[j] = -(1/tau + 2\*bcoef/h2/h2)

c[j] = bcoef/h2/h2

d[j] = -(f(x[i],y[j],t[k])/2+u1[i][j]/tau)

a[-1] = -1/h2

b[-1] = 1/h2

c[-1] = 0

d[-1] = -np.sin(x[i])\*np.sin(mu\*t[k])

result2 = slau([a,b,c,d])

#print(result2)

for j in range(n):

u2[i][j] = result2[j]

u2[0][j] = 0

u2[-1][j] = np.sin(y[j])\*np.sin(t[k])

for i in range(n):

u2[i][-1] = -np.sin(x[i])\*np.sin(mu\*t[k])\*h2 + u2[i][-2]

for i in range(n):

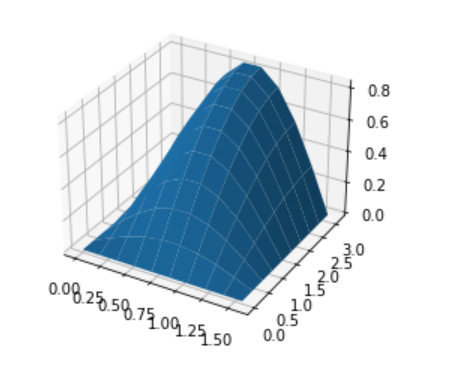
for j in range(n):

u[i][j][k] = u2[i][j]

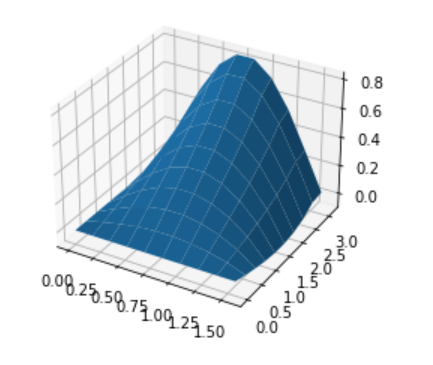
return u

**Результат:**

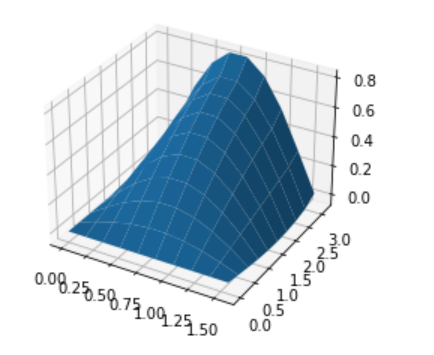
Аналитическое решение:



МПН:



МДШ:



**Вывод:**

Я реализовала схемы переменных направлений и дробных шагов для решения двумерной начально-краевой задачи для дифференциального уравнения параболического типа. Обе схемы дали не очень точное решение