Санкт-Петербургский политехнический университет

Высшая школа теоретической механики, ФизМех

Направление подготовки

«01.03.03 Механика и математическое моделирование»

Отчет по индивидуальной работе №**03**

тема "Метод конечных разностей. Уравнение Лапласа"

дисциплина "Вычислительная механика"

Выполнил студент гр. 90301 **М. А.Бенюх**

Преподаватель: Е.Ю. Витохин

Санкт-Петербург

2021

Оглавление.

1. **Формулировка задания:**
   Методом конечных разностей, используя итерационную схему интегрирования, решить уравнение Лапласа.
2. **Постановка задачи:**

Объект моделирования: Среда с однородными граничными условиями.

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{d y^2} = 0$$

при заданных начальных условиях $u(x,0) = f(x),\ u(x,0) = \Phi(x),$ где $x \in [0,1]$. Решение выполнить при $h = 0.2$ для $t \in [0,1]$.

$$u(x,0) = 20x^2, u(x,1) = \cos\left(\frac{\pi x}{2}\right),$$
$$u(0,y) = \cos\left(\frac{\pi x}{2}\right), u(1,y) = 20y$$

Метод решения:

Разложим $U(x,t)$ в окрестности точки x_0 в ряд:

$$U(x_0 + h) = U(x_0) + U'(x_0)\frac{h}{1!} + U''(x_0)\frac{h^2}{2!} + o(h^2)$$
$$U(x_0 - h) = U(x_0) - U'(x_0)\frac{h}{1!} + U''(x_0)\frac{h^2}{2!} + o(h^2)$$
$$U(x_0 + h) + U(x_0 - h) = 2U(x_0) + U''(x_0)h^2 + o(h^2)$$
$$U''(x_0) = \frac{U(x_0 + h) - 2U(x_0) + U(x_0 - h)}{h^2} + o(h^2)$$

Разложим $U(x,t)$ в окрестности точки t_0 в ряд:

$$U(y_0 + \Delta y) = U(y_0) + U'(y_0)\frac{\Delta t}{1!} + U''(y_0)\frac{\Delta y^2}{2!} + o(\Delta y^2)$$
$$U(y_0 - \Delta \mathrm{y}) = U(y_0) - U'(y_0)\frac{\Delta y}{1!} + U''(y_0)\frac{\Delta y^2}{2!} + o(\Delta y^2)$$
$$U(y_0 + h) + U(y_0 - h) = 2U(y_0) + U''(y_0)h^2 + o(h^2)$$
$$U''(y_0) = \frac{U(y_0 + h) - 2U(y_0) + U(y_0 - h)}{h^2} + o(h^2)$$

Введем сетки для времени $y = k \cdot \Delta y$ и для пространства $x = j \cdot h$. Тогда:

$$U''_{xx}(x,y) = \frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{h^2}$$
$$U''_{yy}(x,y) = \frac{U_{i,j-1} - 2U_{i,j} + U_{i,j+1}}{h^2}$$

Конечно-разностное уравнение примет вид:

$$U_{i,j} = \frac{1}{4}\left(U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1}\right)$$

Итерационный метод:

$$\widetilde{U_{\iota,J}} = \frac{1}{4}\left(U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1}\right)$$

$$U_{i,j}^{k+1} = U_{i,j} + w * (\widetilde{U_{\iota,J}} - U_{i,j})$$

Остановка

$$\left\| U_{i,j}^{k+1} - U_{i,j}^{k} \right\| = \max_{i,j} \left| U_{i,j}^{k+1} - U_{i,j}^{k} \right| \le \varepsilon$$

3. Решение

Оптимальная константа метода: w= 1.8 1.9

w	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
k	10	11	12	12	12	12	12	12	11	11	11	11	11	10	10	10	10	9	9

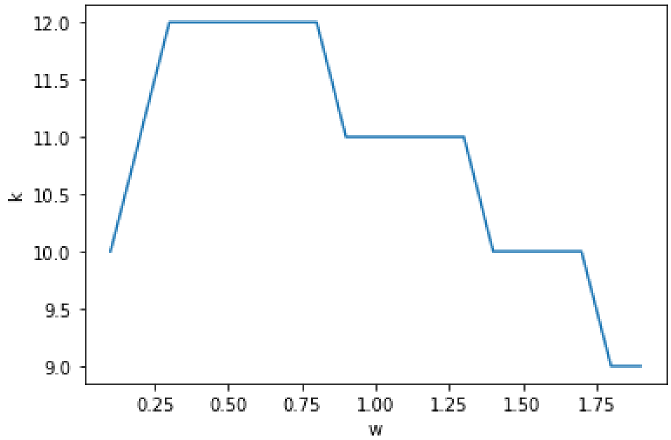


Рис. 1
зависимость числа итераций от w

Далее будем рассматривать разрезы и поверхность для w=1.9

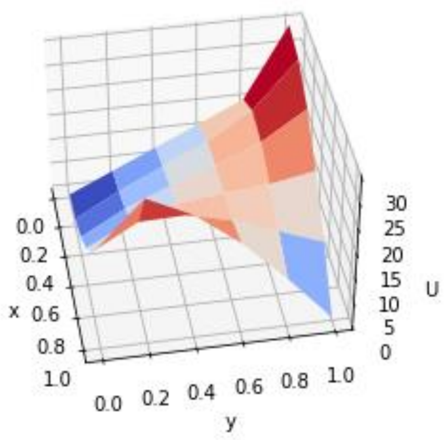


Рис. 2
Поверхность решения U (x, y) при $\omega = 1.9$

Таблица значений матрицы решений

U =

y

x

0	0.8	3.2	7.2	12.8	20
4	5.69663	8.1342	11.8401	17.805	28.5317
8	9.87965	11.7877	14.2356	18.048	24.2705
12	14.0219	14.9265	15.2692	15.8909	17.6336
16	19.2959	18.6298	16.0363	12.6155	9.27051
30	28.5317	24.2705	17.6336	9.27051	1.83697e-15

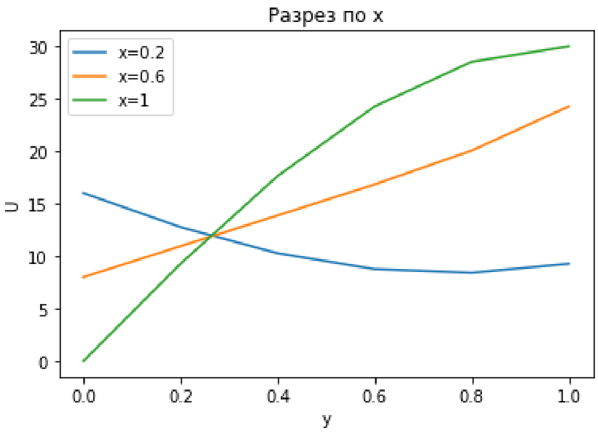


Рис. 3
Разрезы в разные моменты x

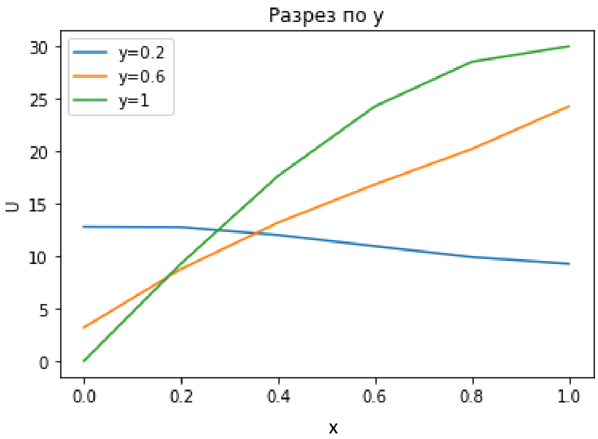
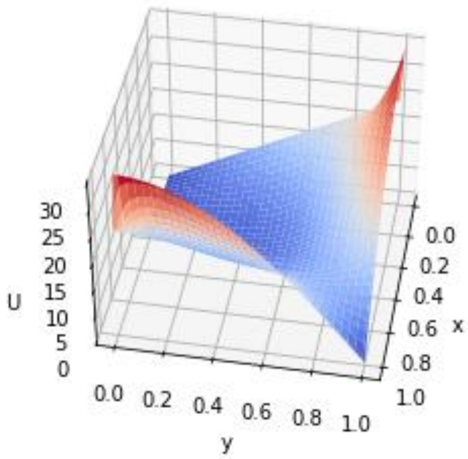


Рис. 4
Разрезы в разные моменты y

4. Проверка решения

Уменьшим шаг разбиения с 0,2 до 0,01 , тогда оптимальная w=0,1

w	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
k	43	79	110	137	161	182	201	218	233	246	259	270	280	290	300	310	320	331	345



w=1.9

5. Заключение

Было получено решение уравнения Лапласа при помощи итерационного метода. Исходя из поставленных условий и h=0.2 оптимальной константой метода будет w=1,9. Проверка сходимости метода при измельчении шага прошла успешно.

6. Код

```
x,h,y,eps=1,0.2,1,0.01

func_psi_0 = lambda y: 20*y
func_psi_l = lambda y: [30*math.cos((math.pi*i)/2) for i in y]
func_fi_0  = lambda x: 20*(x**2)
func_fi_l  = lambda x: [30*math.cos((math.pi*i)/2) for i in x] Test=Struna(x, t, h, dt, func_x_0, d_func_x_0 ,func2_x_0 , func_0_t ,
func_1_t)
Test=Laplas(x, y, h, func_fi_0, func_fi_l ,func_psi_0 , func_psi_l,eps) res2=Test.implicit_schema()
_return=Test.iterative_procedures(m)
  res1=_return[0]
class Laplas():
  def __init__(self,x, y, h, func_fi_0, func_fi_l ,func_psi_0 , func_psi_l,eps):
    self.eps=eps
    self.x= x
    self.x_0=0
    self.y=y
    self.y_0=0
    self.h=h
    self.func_fi_0=func_fi_0
    self.func_fi_l=func_fi_l
    self.func_psi_0=func_psi_0
    self.func_psi_l=func_psi_l
    self.X=np.arange(self.x_0,self.x+self.h, self.h)
    self.Y=np.arange(self.y_0,self.y+self.h,  self.h)
    self.len_X=len(self.X)
    self.len_Y=len(self.Y)
    print("init")

  def StartFillMatrix(self):
    T=np.zeros((self.len_X, self.len_Y))
    T[:,0]=self.func_psi_0(self.Y)
    T[:,-1]=self.func_psi_l(self.Y)
    T[-1,:]=self.func_fi_l(self.X)
    T[0,:]=self.func_fi_0(self.X)

    return T

  def iterative_procedures(self,w):
    T=self.StartFillMatrix()
    T_new=np.zeros((self.len_X, self.len_Y))
    T2=self.StartFillMatrix()
    k=0
    while (True) :
      k=k+1
      delta=[]
      for i in range(1,self.len_X-1):
        for j in range(1,self.len_Y-1):
          T[i,j]=(1/4)*(T[i-1,j]+T[i+1,j]+T[i,j-1]+T[i,j+1])

      for i in range(1,self.len_X-1):
        for j in range(1,self.len_Y-1):
          T2[i,j]=(1/4)*(T[i-1,j]+T[i+1,j]+T[i,j-1]+T[i,j+1])

      for i in range(0,self.len_X):
        for j in range(0,self.len_Y):
          T_new[i,j]=T[i,j]+w*(T2[i,j]-T[i,j])
```

```
        for i in range(0,self.len_X):
            for j in range(0,self.len_Y):
                delta.append(abs(T_new[i,j]-T[i,j]))

        if(max(delta)<self.eps):
            print("break")
            break
        else:
            for i in range(0,self.len_X):
                for j in range(0,self.len_Y):
                    T[i,j]=T_new[i,j]

    return [T_new, k]
```