Санкт-Петербургский политехнический университет

Высшая школа теоретической механики, ФизМех

Направление подготовки

«01.03.03 Механика и математическое моделирование»

Отчет по индивидуальной работе №**02**

тема "Метод конечных разностей. Уравнение колебаний струны"

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 явную и неявной схему интегрирования, решить уравнение колебаний струны.
2. **Постановка задачи:**

Объект моделирования: Среда с однородными граничными условиями.

$$\frac{\partial^2 u}{d^2 t} = \frac{\partial^2 u}{dx^2}$$

при заданных начальных условиях $u(x,0) = f(x),\ u_t(x,0) = \Phi(x),$ где $x \in [0,1];\ u(0,t) = \varphi(t),\ u(1,t) = \psi(t)$. Решение выполнить при $h = 0.1$ для $t \in [0, 0.5]$.

$$u(x,0) = \cos\left(\frac{\pi x}{2}\right), u'(x,0) = x^2,$$
$$u(0,t) = 1 + 2t, u(1,t) = 0$$

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:

$$U(t_0 + \Delta t) = U(t_0) + U'(t_0)\frac{\Delta t}{1!} + U''(t_0)\frac{\Delta t^2}{2!} + o(\Delta t^2)$$
$$U(t_0 - \Delta t) = U(t_0) - U'(t_0)\frac{\Delta t}{1!} + U''(t_0)\frac{\Delta t^2}{2!} + o(\Delta t^2)$$
$$U(t_0 + \Delta t) + U(t_0 - h) = 2U(t_0) + U''(t_0)\Delta t^2 + o(\Delta t^2)$$
$$U''(t_0) = \frac{U(t_0 + \Delta t) - 2U(t_0) + U(t_0 - \Delta t)}{\Delta t^2} + o(\Delta t^2)$$

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

$$\ddot{U}(\mathrm{x}, t) = \frac{U_j^{k+1} - 2U_j^k + U_j^{k-1}}{\Delta t^2}$$
$$U''(x,t) = \frac{U_{j+1}^k - 2U_j^k + U_{j-1}^k}{h^2}$$

Конечно-разностное уравнение примет вид:

$$\frac{U_{j+1}^k - 2U_j^k + U_{j-1}^k}{h^2} - \frac{1}{c^2}\frac{U_j^{k+1} - 2U_j^k + U_j^{k-1}}{\Delta t^2} = 0$$

Явная схема метода:

$$U_j^{k+1} = \frac{c^2 \Delta t^2}{h^2}\left(U_{j+1}^k - 2U_j^k + U_{j-1}^k\right) + 2U_j^k - U_j^{k-1}$$

Для преобразования начальных условий воспользуемся формулой 2-го порядка точности:

$U_j^1 = U_j^*(x) + U_j^{**}(x)\Delta t + \frac{c^2 \Delta t^2}{2h^2}(U_{i+1}^* - 2U_i^* + U_{i-1}^*)$

где $U^*(x) = U(x,0),\ \ U^{**}(x) = U_t(x,0)$

Для вывода неявной схемы интегрирования введем значение для аппроксимации частных производных:

$$\frac{\partial^2 U}{\partial t^2} = \frac{U_{j+1}^{k-1} - 2U_{j+1}^{k} - U_{j+1}^{k+1}}{\Delta t^2}$$
$$\frac{\partial^2 U}{\partial x^2} = \frac{U_{j-1}^{k+1} - 2U_{j}^{k+1} - U_{j+1}^{k+1}}{h^2}$$

Далее воспользуемся формулами:

$$-AU_{i-1}^{k+1} + BU_i^{k+1} - CU_{i+1}^{k+1} = F_i$$
$$A = \frac{1}{h^2}$$
$$B = \frac{2c^2\Delta t^2 + h^2}{c^2\Delta t^2 h^2}$$
$$C = \frac{1}{h^2}$$
$$F_i = \frac{2}{\Delta t^2 c^2} U_i^k - \frac{1}{\Delta t^2 c^2} U_i^{k-1}$$

Решение можно получить неявно, используя метод прогонки:

Прямой ход:

$P_i = \frac{c}{B - AP_i}$ $\quad Q_i = \frac{F_i + AQ_{i-1}}{B - AP_{i-1}}$

$P_1 = \frac{C}{B}$ $\quad Q_1 = \frac{F_i^k}{B}$

Обратный ход:

$$U_i^{k+1} = P_i U_{i+1}^{k+1} + Q_i$$

Отдельно отметим, что шаг по времени Δt необходимо выбирать из условия Куранта о сходимости явной схемы интегрирования: $\Delta t \ll \frac{h}{c}$

3. Явная схема интегрирования

$$U_j^{k+1} = \frac{c^2\Delta t^2}{h^2}\left(U_{j+1}^k - 2U_j^k + U_{j-1}^k\right) + 2U_j^k - U_j^{k-1}$$

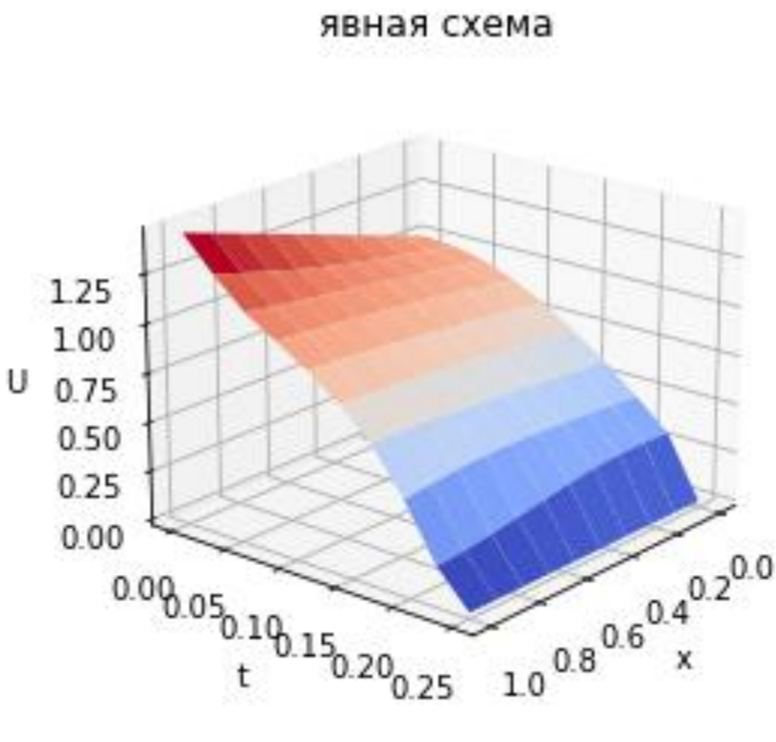

Рис. 1
Визуализация матрицы решений
(явной схемы интегрирования)

Таблица значений матрицы решений (неявной схемы интегрирования)

U1 =

t

x

0.987688	1.01238	1.03708	1.06321	1.09194	1.12402	1.15976	1.19899	1.24116	1.28544
0.951057	0.974833	0.997109	1.01795	1.03759	1.05651	1.07536	1.09498	1.11623	1.13988
0.891007	0.913282	0.934151	0.953583	0.971554	0.988064	1.00317	1.017	1.02983	1.04204
0.809017	0.829242	0.848192	0.865836	0.882147	0.897101	0.910678	0.922863	0.93366	0.943092
0.707107	0.724784	0.741347	0.756768	0.771025	0.784092	0.795938	0.8065	0.815665	0.823235
0.587785	0.60248	0.616247	0.629066	0.640878	0.651522	0.660669	0.667786	0.672153	0.672938
0.45399	0.46534	0.475974	0.485249	0.492039	0.495005	0.492932	0.485059	0.471334	0.452529
0.309017	0.316742	0.313959	0.301679	0.282017	0.257855	0.2324	0.208702	0.189233	0.175569
0.987688	1.01238	1.03708	1.06321	1.09194	1.12402	1.15976	1.19899	1.24116	1.28544
0.951057	0.974833	0.997109	1.01795	1.03759	1.05651	1.07536	1.09498	1.11623	1.13988

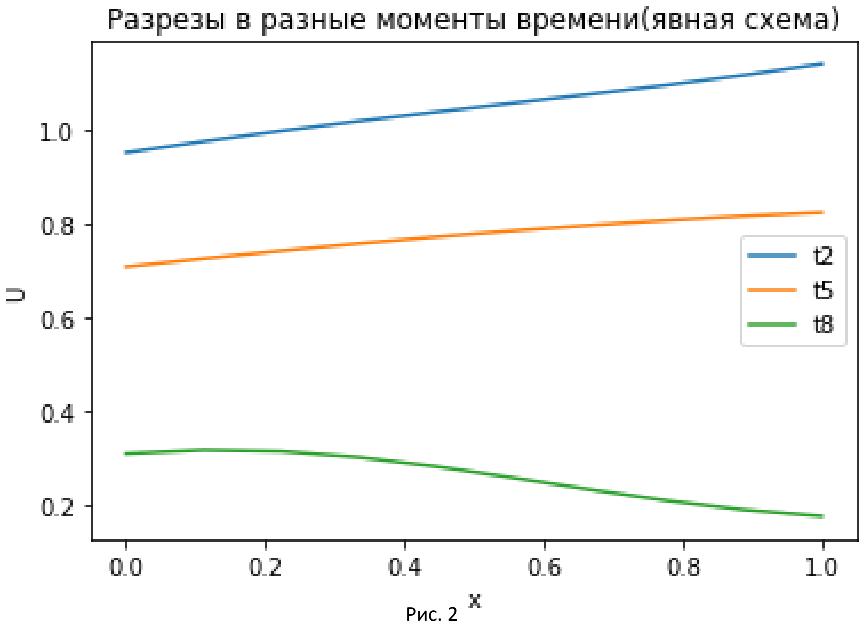

Рис. 2
Разрезы в разные моменты времени
(явной схемы интегрирования)

4. Неявная схема интегрирования

$$-AU_{i-1}^{k+1} + BU_i^{k+1} - CU_{i+1}^{k+1} = F_i$$

$$A = \frac{1}{h^2};\ B = \frac{2c^2\Delta t^2 + h^2}{c^2\Delta t^2 h^2};\ C = \frac{1}{h^2};\quad F_i = \frac{2}{\Delta t^2 c^2}U_i^k - \frac{1}{\Delta t^2 c^2}U_i^{k-1}$$

Метод матричной прогонки:

Прямой ход: $P_i = \frac{c}{B-AP_i}; Q_i = \frac{(F_i+AQ_{i-1})}{B-AP_{i-1}}; P_1 = \frac{C}{B}; Q_1 = \frac{F_i^k}{B}$

Обратный ход: $T_i^{k+1} = P_i T_{i+1}^{k+1} + Q_i$

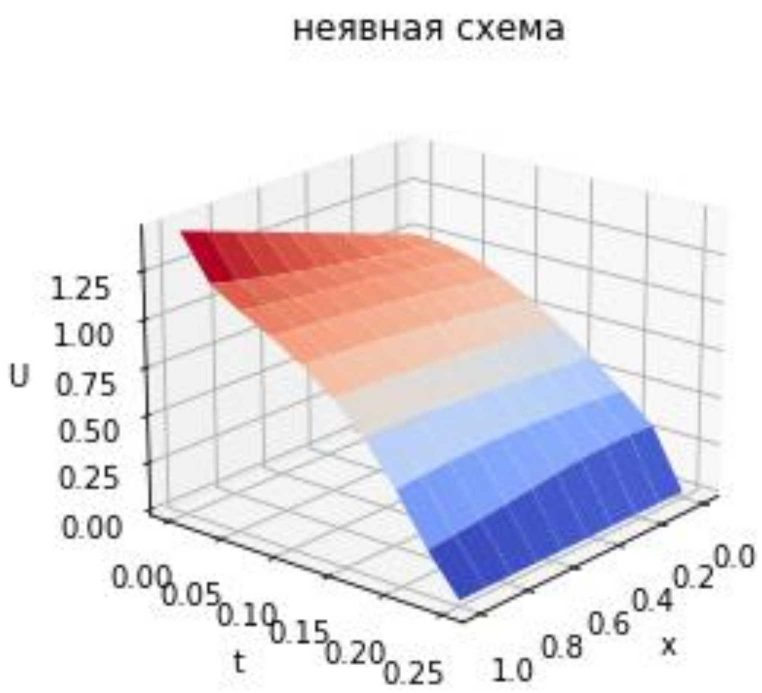

Рис. 3
Визуализация матрицы решений
(неявной схемы интегрирования)

Таблица значений матрицы решений (явной схемы интегрирования)

U2 =

t

x

0.987688	1.01238	1.03475	1.05624	1.07817	1.10159	1.12721	1.15544	1.18635	1.21978
0.951057	0.974833	0.997034	1.01764	1.03674	1.05461	1.07163	1.08833	1.1053	1.12312
0.891007	0.913282	0.934117	0.953479	0.971343	0.987707	1.00261	1.01615	1.0285	1.03991
0.809017	0.829242	0.848162	0.865749	0.881974	0.896808	0.910223	0.922185	0.932661	0.941623
0.707107	0.724784	0.74132	0.756681	0.770821	0.783668	0.795108	0.804977	0.813058	0.819094
0.587785	0.60248	0.616198	0.628823	0.640142	0.649831	0.65748	0.662632	0.664859	0.663829
0.45399	0.46534	0.475448	0.483485	0.488557	0.489928	0.487191	0.480354	0.469847	0.456457
0.309017	0.316742	0.31483	0.305009	0.289531	0.270821	0.251165	0.232472	0.216129	0.202946
0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
0.987688	1.01238	1.03475	1.05624	1.07817	1.10159	1.12721	1.15544	1.18635	1.21978

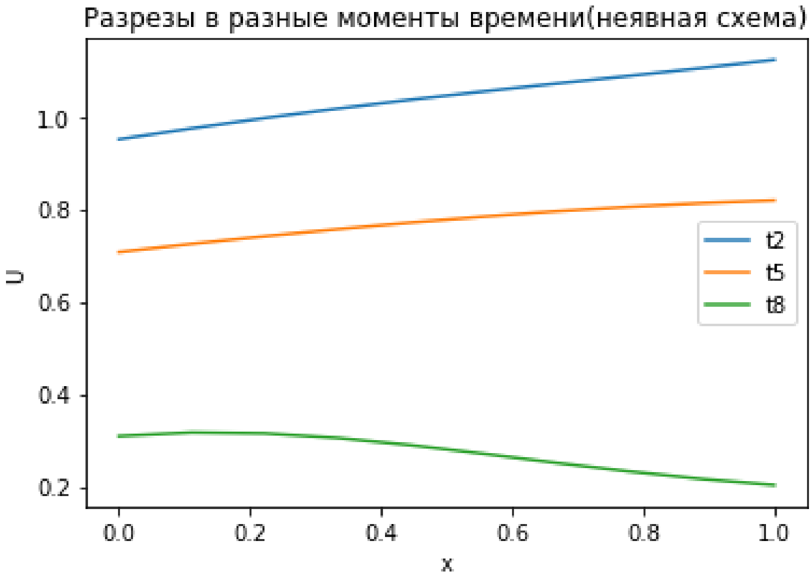

Рис. 4
Разрезы в разные моменты времени
(неявной схемы интегрирования)

5. Численный анализ решения задач

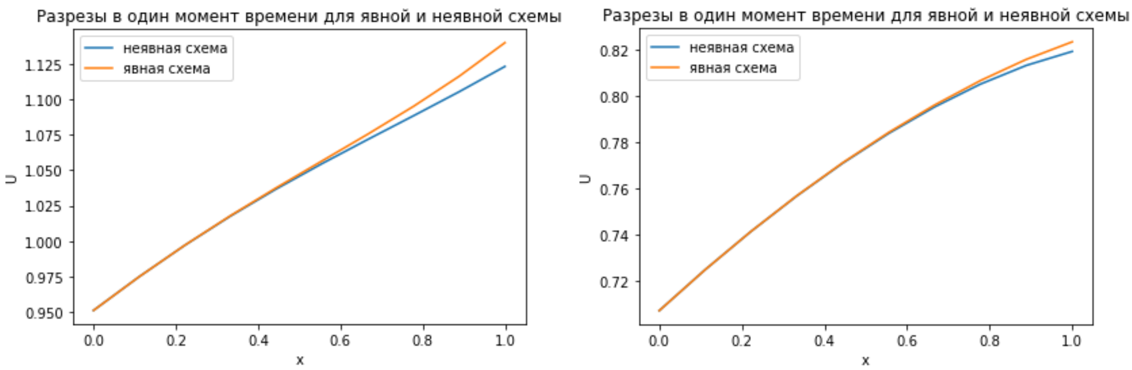

Рис. 5,6
Разрезы явной и неявной схемы
в момент времени 2(рис. 5),
в момент времени 5(рис. 6),

6. Заключение

Было получено решение тепловой задачи при помощи метода конечных разностей с применением явной и неявной схемы интегрирования. Неявная схема показала себя более гладкой по сравнению с явной.

7. Код

```
x,h,t,dt=1,0.1,0.25,0.05
func_x_0  = lambda x: [math.cos((math.pi*i)/2) for i in x]
func2_x_0 = lambda x: math.cos((math.pi*x)/2)
d_func_x_0= lambda x: x**2
func_0_t  = lambda t: 1+2*t
func_1_t  = lambda t: 0
Test=Struna(x, t, h, dt, func_x_0, d_func_x_0 ,func2_x_0 , func_0_t , func_1_t)
res1=Test.explicit_schema()
res2=Test.implicit_schema()
def __init__(self, x, t, h, dt, func_x_0, func2_x_0 , d_func_x_0 , func_0_t , func_1_t):
    self.x= x
    self.x_0=0
    self.t=t
    self.t_0=0
    self.h=h
    self.func_x_0=func_x_0
    self.func2_x_0=func2_x_0
    self.d_func_x_0=d_func_x_0
    self.func_0_t=func_0_t
    self.func_1_t=func_1_t
    self.X=np.arange(self.x_0,self.x, self.h)
    self.dt=self.t/(len(self.X))
    self.Time=np.arange(self.t_0,self.t,  self.dt)
    self.len_X=len(self.X)
    self.len_T=len(self.Time)
    print("init")
  def StartFillMatrix(self):
    T=np.zeros((self.len_X, self.len_T))
    T[:,0]=self.func_x_0(self.X)
    T[0,:]=self.func_0_t(self.Time)
    T[-1,:]=self.func_1_t(self.Time)

    #T[:,1]
    for i in range(1,self.len_X-1):
      print(i)
      print(self.X[i])
      T[i,1]=T[i,0]+self.d_func_x_0(self.X[i])*self.dt+((self.dt**2)/(2*(self.h**2)))*(T[i,0]-2*T[i,0]+T[i,0])

    return T
  def explicit_schema(self):
    T=self.StartFillMatrix()
    c=1
    for k in range(1,self.len_T-1):
      for i in range(1,self.len_X-1):
        T[i,k+1]=(((c**2)*(self.dt**2))/(self.h**2))*(T[i+1,k]-2*T[i,k]+T[i-1,k])+2*T[i,k]-T[i,k-1]

    return T
  def implicit_schema(self):
    T=self.StartFillMatrix()
    A=1/(self.h**2)
    C=1/(self.h**2)
    B=(2*(self.dt**2)+(self.h**2))/((self.h**2)*(self.dt**2))
    F=np.zeros((self.len_X))
    P=np.zeros((self.len_X))
    Q=np.zeros((self.len_X))
```

```
        for k in range(1,self.len_T-1):
            for i in range(0, self.len_X):
                F[i]=((2*T[i,k])/(self.dt**2))-(T[i,k-1]/(self.dt**2))
            P[0]=C/B
            Q[0]=F[0]/B
            for i in range(1,self.len_X):
                P[i]=C/(B-A*P[i-1])
                Q[i]=(F[i]+A*Q[i-1])/(B-A*P[i-1])
            for i in range(self.len_X-2,0,-1):
                T[i,k+1]=P[i]*T[i+1,k+1]+Q[i]

    return T
```