МИНЕСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ

Курсовая работа по дисциплине

«Численные методы»

Тема: «Расчет колебаний тонкой пластины без учета потерь на трение»

Выполнил: студент группы А-13а-19
Чуворкин Михаил
Преподаватель: Амосова О. А.

Москва, 2021

1 Оглавление

1. Постановка задачи

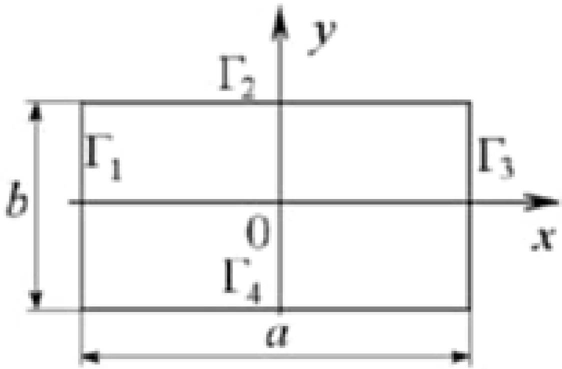
Колебания тонкой пластины без учета потерь на трение описывается нормированным волновым уравнением вида

$$\frac{\partial^2 u}{\partial t^2} - \Delta u = 0$$

Где $u(x, y, t)$ – деформация пластины, x, y – координаты, t – время.

Задание: рассчитать колебания при заданных размерах a, b, граничных $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4$ и начальных $u(x, y, 0)$ и $\frac{\partial u}{\partial t}(x, y, 0)$ условиях.

Вариант 2:



$$a = 2$$
$$b = 1$$
$$\Gamma_1, \Gamma_3 : u = 0$$
$$\Gamma_2, \Gamma_4 : \frac{\partial u}{\partial n} = 0$$

$$\begin{cases} \mathrm{u(x, y, 0)} = \arctan\left[\cos\left(\frac{\pi x}{a}\right)\right] \\ \frac{\partial u}{\partial t}(x, y, 0) = sin\left(\frac{2\pi x}{a}\right) sin\left(\frac{\pi y}{b}\right) \end{cases}$$

Запишем граничные условия с учетом размеров:

$$\Gamma_1, \Gamma_3 : \mathrm{u}\big|_{\mathrm{x} = \pm\frac{\mathrm{a}}{2}} = 0$$

$$\Gamma_2, \Gamma_4 : \left.\frac{\partial u}{\partial n}\right|_{\mathrm{y} = \pm\frac{\mathrm{b}}{2}} = 0$$

2. Необходимый теоретический материал

Колебания плоской однородной мембраны описываются уравнением колебаний:

$$u_{tt} = a^2 \Delta u$$

В нашем случае $a = 1$, поэтому уравнение принимает вид

$$u_{tt} = \Delta u$$
$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \qquad (1)$$

Уравнение решается при заданных начальных условиях

$$\begin{cases} u(x, y, 0) = \varphi(x, y) \\ \frac{\partial u}{\partial t}(x, y, 0) = \psi(x, y) \end{cases} \qquad (2)$$

И граничных условиях (если закреплены все 4 края пластины и пластина лежит в прямоугольнике $[0, a] \times [0, b]$):

$$\begin{cases}\begin{cases}u(0,y,t)=0\\u(a,y,t)=0\end{cases}\\\begin{cases}u(x,0,t)=0\\u(x,b,t)=0\end{cases}\end{cases}\qquad(3)$$

Если же пластина лежит в прямоугольнике $\left[-\frac{a}{2},\frac{a}{2}\right]\times\left[-\frac{b}{2},\frac{b}{2}\right]$ и закреплены только 2 противоположных края, как в нашей задаче, то граничные условия принимают вид:

$$\begin{cases}\begin{cases}u\left(\frac{a}{2},y,t\right)=0\\u\left(-\frac{a}{2},y,t\right)=0\end{cases}\\\begin{cases}\frac{\partial u}{\partial n}\left(x,\frac{b}{2},t\right)=0\\\frac{\partial u}{\partial n}\left(x,-\frac{b}{2},t\right)=0\end{cases}\end{cases}\qquad(4)$$

Решение уравнения (1) при условиях (2) и (3) имеет вид:

$$u(x,y,t)=\sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\left(\bar{B}_{n,m}\cos\sqrt{\lambda_{n,m}}\,t+\bar{\bar{B}}_{n,m}\sin\sqrt{\lambda_{n,m}}\,t\right)v_{n,m}(x,y)\qquad(5)$$

Где

$v_{n,m}(x,y)=A_{n,m}\sin\left(\frac{n\pi x}{a}\right)\sin\left(\frac{m\pi y}{b}\right)$ – собственные функции,

$A_{n,m}$ – некоторый постоянный множитель, берется как $\sqrt{\frac{4}{ab}}$.

$$\bar{B}_{n,m}=A_{n,m}\int_0^a\int_0^b\varphi(x,y)\sin\left(\frac{n\pi x}{a}\right)\sin\left(\frac{m\pi y}{b}\right)\mathrm{d}x\,\mathrm{d}y\qquad(6)$$

$$\bar{\bar{B}}_{n,m}=\frac{1}{\sqrt{\lambda_{n,m}}}A_{n,m}\int_0^a\int_0^b\psi(x,y)\sin\left(\frac{n\pi x}{a}\right)\sin\left(\frac{m\pi y}{b}\right)dxdy\qquad(7)$$

В случае двух свободных концов (4) и смещения задача для Y принимает вид:

$$\begin{cases}Y''+\mu Y=0\\Y'\left(-\frac{b}{2}\right)=0;\quad Y'\left(\frac{b}{2}\right)=0\end{cases}$$

Решая ее, получим $Y(y)=C_m\cos\left(\frac{2m\pi y}{b}\right)$ (одно из решений – случай, когда в размер $\left[-\frac{b}{2},\frac{b}{2}\right]$ укладывается полный период косинуса)

Аналогично $\mathrm{X}(x)=C_n\sin\left(\frac{2n\pi x}{a}\right)$

3. Построение тестовых примеров

3.1. Первый тестовый пример

Возьмем:

$$\mathrm{u}(x,y,t)=\frac{1}{\sqrt{\lambda_{1,1}}}sin\left(\sqrt{\lambda_{1,1}}t\right)sin\left(\frac{2\pi x}{a}\right)\cos\left(\frac{2\pi y}{b}\right)$$

$$\mathrm{u}(x,y,t)=\frac{1}{\pi\sqrt{5}}sin\left(\pi\sqrt{5}t\right)sin(\pi x)\cos(2\pi y)$$

Проверим выполнение граничных условий и получим начальные условия:

$\mathrm{u}(x,y,0)=0=\varphi(x,y)$ – так как аргумент синуса равен 0

$$\frac{\partial u}{\partial t}(x,y,0)=\frac{\pi\sqrt{5}}{\pi\sqrt{5}}\cos\left(\pi\sqrt{5}t\right)sin(\pi x)\cos(2\pi y)\ =\ sin(\pi x)\cos(2\pi y)\ =\psi(x,y)$$

$$\begin{cases}\begin{cases}u(1,y,t)=\dfrac{1}{\pi\sqrt{5}}sin\left(\pi\sqrt{5}t\right)sin(\pi)\cos(2\pi y)\ =\ 0\\ u(-1,y,t)=\dfrac{1}{\pi\sqrt{5}}sin\left(\pi\sqrt{5}t\right)sin(-\pi)\cos(2\pi y)=0\end{cases}\\ \begin{cases}\dfrac{\partial u}{\partial y}\left(x,\dfrac{1}{2},t\right)=-\dfrac{2\pi}{\pi\sqrt{5}}sin\left(\pi\sqrt{5}t\right)sin(\pi x)\,sin(\pi)=0\\ \dfrac{\partial u}{\partial y}\left(x,-\dfrac{1}{2},t\right)=-\dfrac{2\pi}{\pi\sqrt{5}}sin\left(\pi\sqrt{5}t\right)sin(\pi x)\,sin(-\pi)=0\end{cases}\end{cases}$$

3.2. Второй тестовый пример

Добавим к примеру 1 гармонику:

Значение для $\lambda_{2,2}=4\pi^2+16\pi^2=20\pi^2\Rightarrow\ \sqrt{\lambda_{2,2}}=2\pi\sqrt{5}$

$$\mathrm{u}(x,y,t)=\frac{1}{\pi\sqrt{5}}sin\left(\pi\sqrt{5}t\right)sin(\pi x)\cos(2\pi y)+\frac{1}{2\pi\sqrt{5}}sin\left(2\pi\sqrt{5}t\right)sin(2\pi x)\cos(4\pi y)$$

Тогда

$\mathrm{u}(x,y,0)=0=\varphi(x,y)$ – аналогично 1

$\psi(x,y)=sin(\pi x)\cos(2\pi y)+sin(2\pi x)\cos(4\pi y)$

Граничные условия также выполняются, так как в первой гармонике аргументы синуса по x и косинуса по y кратны тем, что были в предыдущем примере.

4. Построение разностной схемы

Запишем уравнение (1), заменив частные производные на вторые разностные производные.

$$\frac{u_{j,i}^{n+1}-2u_{j,i}^{n}+u_{j,i}^{n-1}}{\Delta t^2}=\frac{u_{j+1,i}^{n}-2u_{j,i}^{n}+u_{j-1,i}^{n}}{\Delta x^2}+\frac{u_{j,i+1}^{n}-2u_{j,i}^{n}+u_{j,i-1}^{n}}{\Delta y^2}\qquad(8)$$

Обозначим для удобства:

$$\frac{u_{j+1,i}^{n}-2u_{j,i}^{n}+u_{j-1,i}^{n}}{\Delta x^2}=u_{xx}\qquad(9)$$

$$\frac{u_{j,i+1}^{n} - 2u_{j,i}^{n} + u_{j,i-1}^{n}}{\Delta y^2} = u_{yy} \qquad (10)$$

Из (8) выразим $u_{j,i}^{n+1}$ (новый временной слой):

$$u_{j,i}^{n+1} = 2u_{j,i}^{n} - u_{j,i}^{n-1} + \Delta t^2(u_{xx} + u_{yy}) \qquad (11)$$

Первое начальное условие дает $u_{j,i}^{0} = \varphi(\mathrm{x_j}, \mathrm{y_i}) \qquad (12)$

Для второго начального условия возьмем аппроксимацию первой производной центральной разностной производной:

$$\frac{u_{j,i}^{n+1} - u_{j,i}^{n-1}}{2\Delta \mathrm{t}} = \psi(\mathrm{x_j}, \mathrm{y_i})$$

При $n = 0$:

$$\frac{u_{j,i}^{1} - u_{j,i}^{-1}}{2\Delta \mathrm{t}} = \psi(\mathrm{x_j}, \mathrm{y_i})$$

Выразим $u_{j,i}^{-1}$:

$$u_{j,i}^{-1} = u_{j,i}^{1} - 2\Delta \mathrm{t} \psi(\mathrm{x_j}, \mathrm{y_i})$$

Подставим в (9):

$$u_{j,i}^{1} = 2u_{j,i}^{0} - u_{j,i}^{1} + 2\Delta \mathrm{t} \psi(\mathrm{x_j}, \mathrm{y_i}) + \Delta t^2(u_{xx} + u_{yy})$$

$$u_{j,i}^{1} = u_{j,i}^{0} + \Delta \mathrm{t} \psi(\mathrm{x_j}, \mathrm{y_i}) + \frac{1}{2}\Delta t^2(u_{xx} + u_{yy}) \qquad (13)$$

Рассмотрим граничные условия:

$$\Gamma_1, \Gamma_3 \colon \mathrm{u}\big|_{\mathrm{x}=\pm\frac{\mathrm{a}}{2}} = 0$$

$$\Gamma_1 \colon u_{0,i}^{n} = 0 \quad (14)\,, \qquad \text{где } j = 0 \text{ соответствует } x = -\frac{a}{2}$$

$$\Gamma_3 \colon u_{k,i}^{n} = 0 \quad (15), \qquad \text{где } j = k \text{ соответствует } x = \frac{a}{2}$$

$$\Gamma_2, \Gamma_4 \colon \frac{\partial u}{\partial n}\bigg|_{\mathrm{y}=\pm\frac{\mathrm{b}}{2}} = 0$$

Нормаль на границе Γ_2 соответствует положительному направлению оси Oy, поэтому

$$\Gamma_2 \colon \frac{\partial u}{\partial n} = \frac{\partial u}{\partial y} = 0$$

Нормаль на границе Γ_4 соответствует отрицательному направлению оси Oy, поэтому

$$\Gamma_4 \colon \frac{\partial u}{\partial n} = -\frac{\partial u}{\partial y} = 0$$

Аппроксимируем их центральными разностными производными:

$$\Gamma_2\colon \frac{\partial u}{\partial y} = \frac{u_{j,1}^n - u_{j,-1}^n}{2\Delta y} = 0$$

$$\Gamma_2\colon u_{j,1}^n = u_{j,-1}^n$$

$$\Gamma_4\colon \frac{\partial u}{\partial y} = \frac{u_{j,m+1}^n - u_{j,m-1}^n}{2\Delta \mathrm{y}} = 0$$

$$\Gamma_4\colon u_{j,m+1}^n = u_{j,m-1}^n$$

где $i = 0$ соответствует $y = -\dfrac{b}{2}$

где $i = m$ соответствует $y = \dfrac{b}{2}$

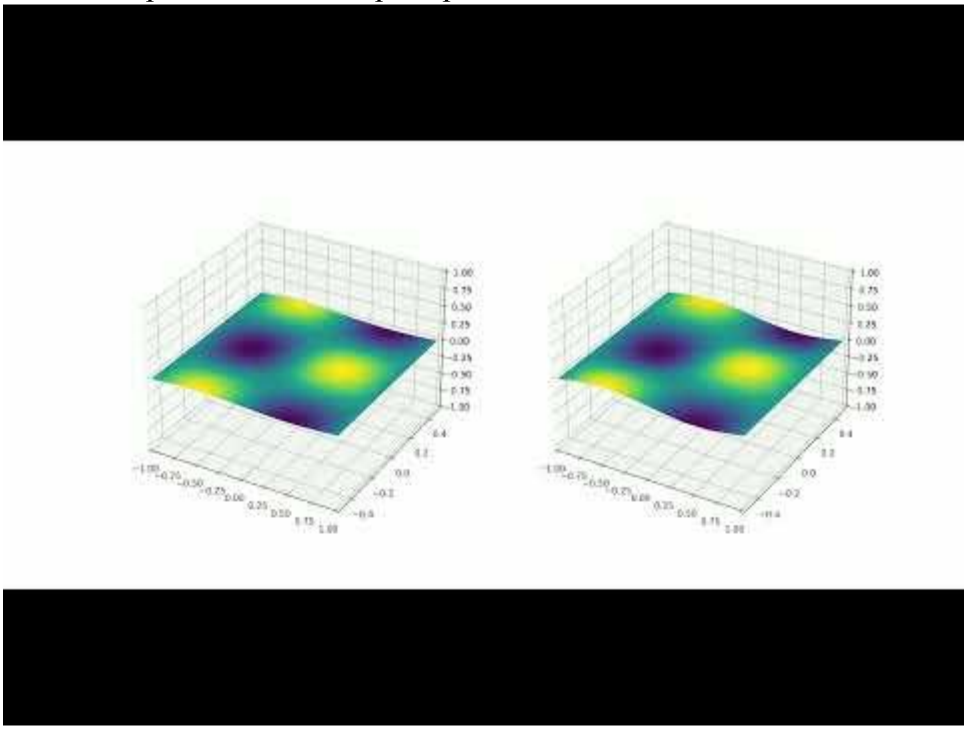
Подставим полученные выражения в схему (11):

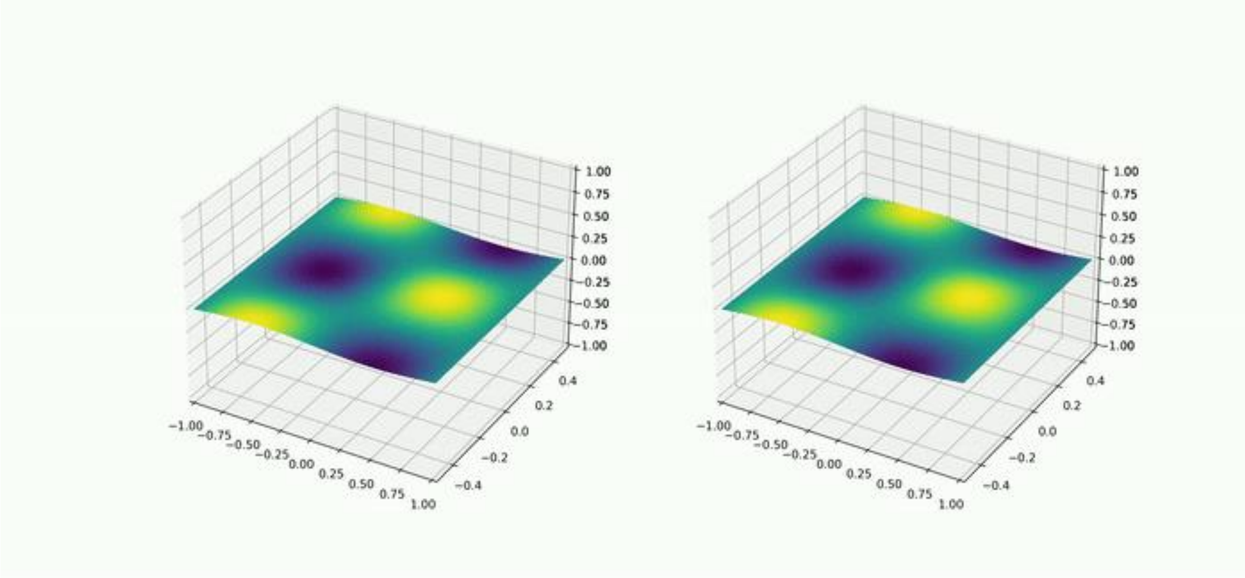
$$\Gamma_2\colon u_{j,0}^{n+1} = 2u_{j,0}^n - u_{j,0}^{n-1} + \Delta t^2 \left(\frac{u_{j+1,0}^n - 2u_{j,0}^n + u_{j-1,0}^n}{\Delta x^2} + \frac{2u_{j,1}^n - 2u_{j,0}^n}{\Delta y^2} \right) \qquad (16)$$

$$\Gamma_4\colon u_{j,m}^{n+1} = 2u_{j,m}^n - u_{j,m}^{n-1} + \Delta t^2 \left(\frac{u_{j+1,m}^n - 2u_{j,m}^n + u_{j-1,m}^n}{\Delta x^2} + \frac{2u_{j,m+1}^n - 2u_{j,m}^n}{\Delta y^2} \right) \qquad (17)$$

5. Результаты расчетов по тестовым примерам

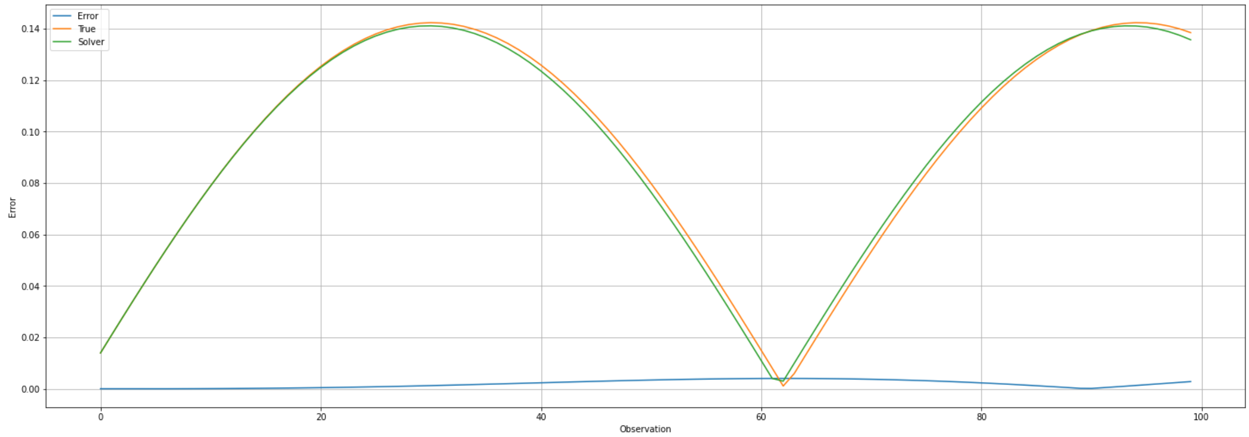
5.1. Первый тестовый пример



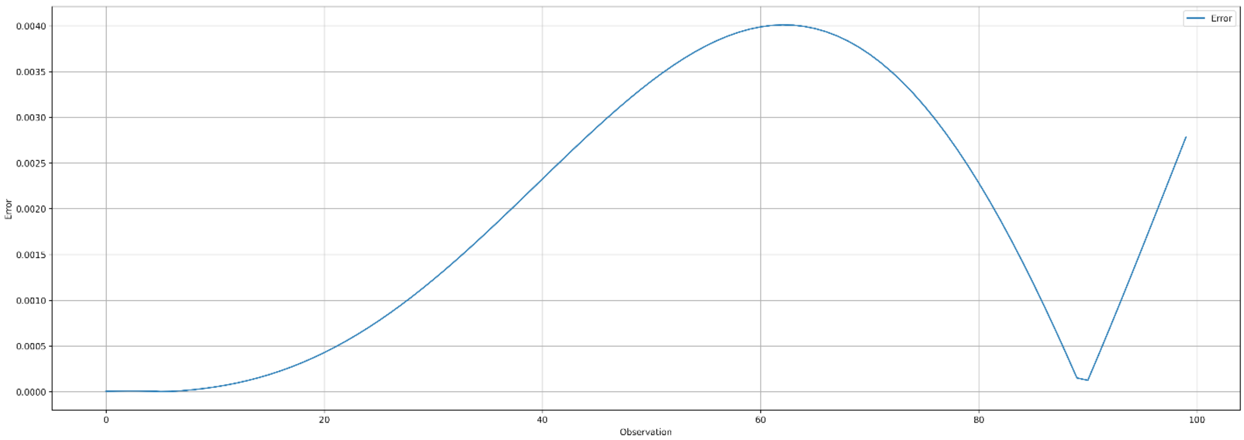


Графики бесконечных норм:

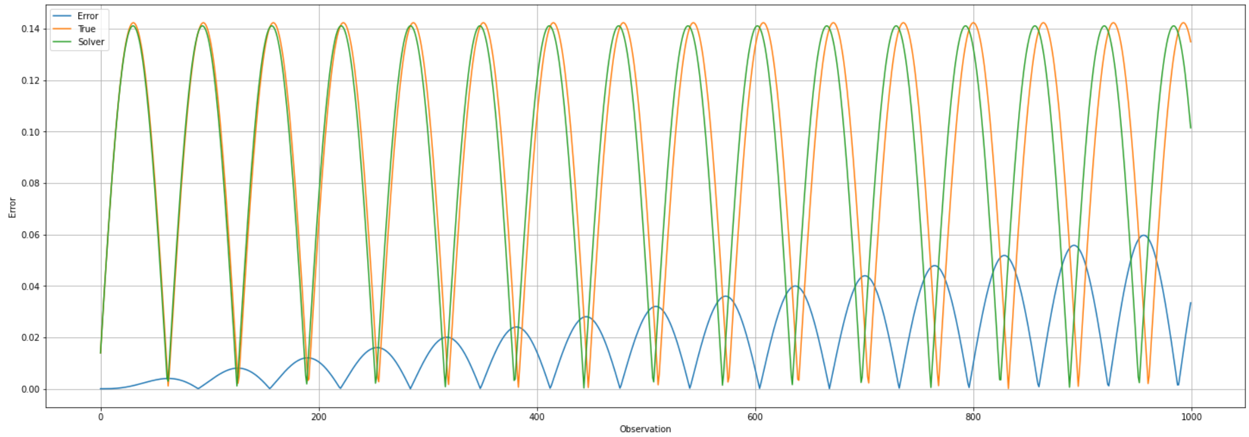
- точного решения(оранжевый)
- с использованием разностной схемы(зеленый)
- погрешности (синий)



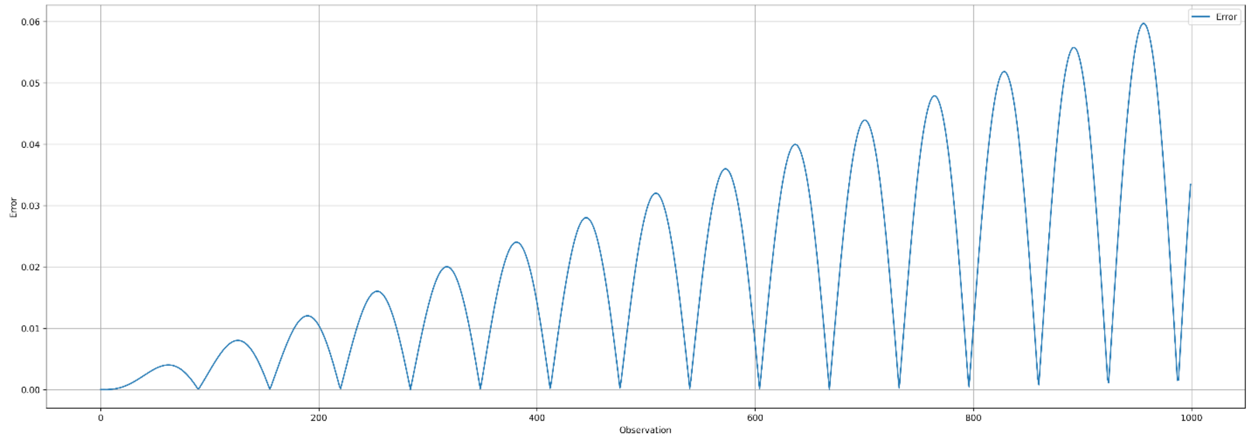
1. 100 временных слоев



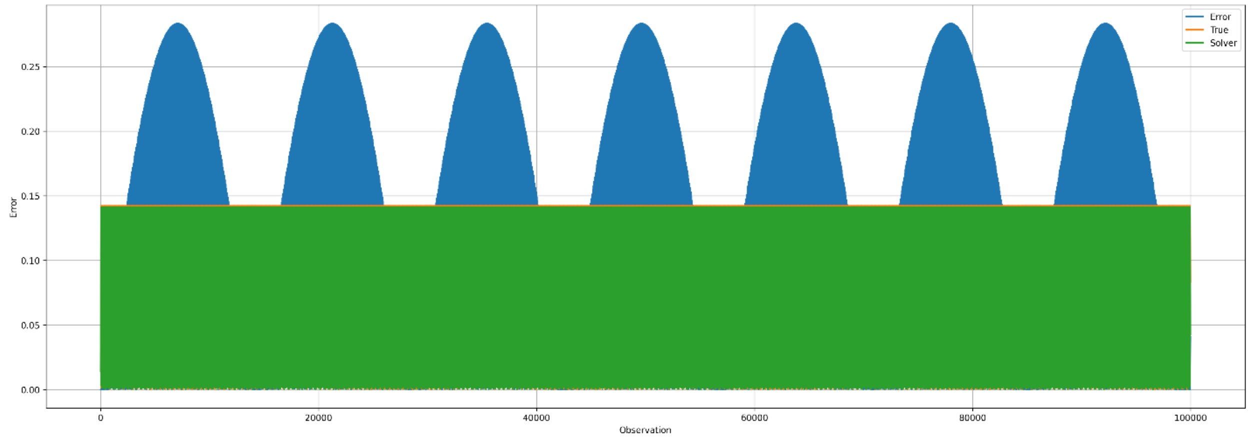
2.100 временных слоев (только погрешность)



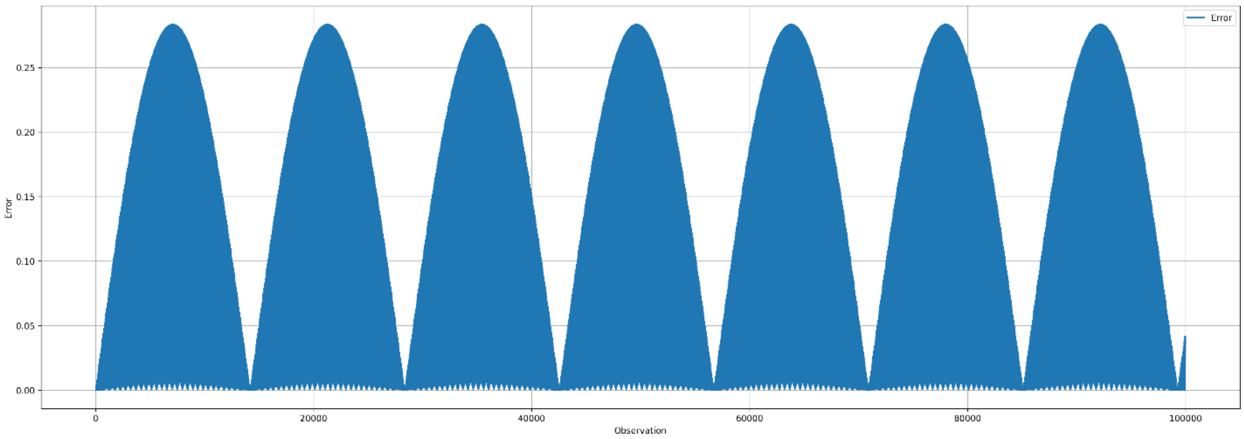
3. 1000 временных слоев



4. 1000 временных слоев (только погрешность)



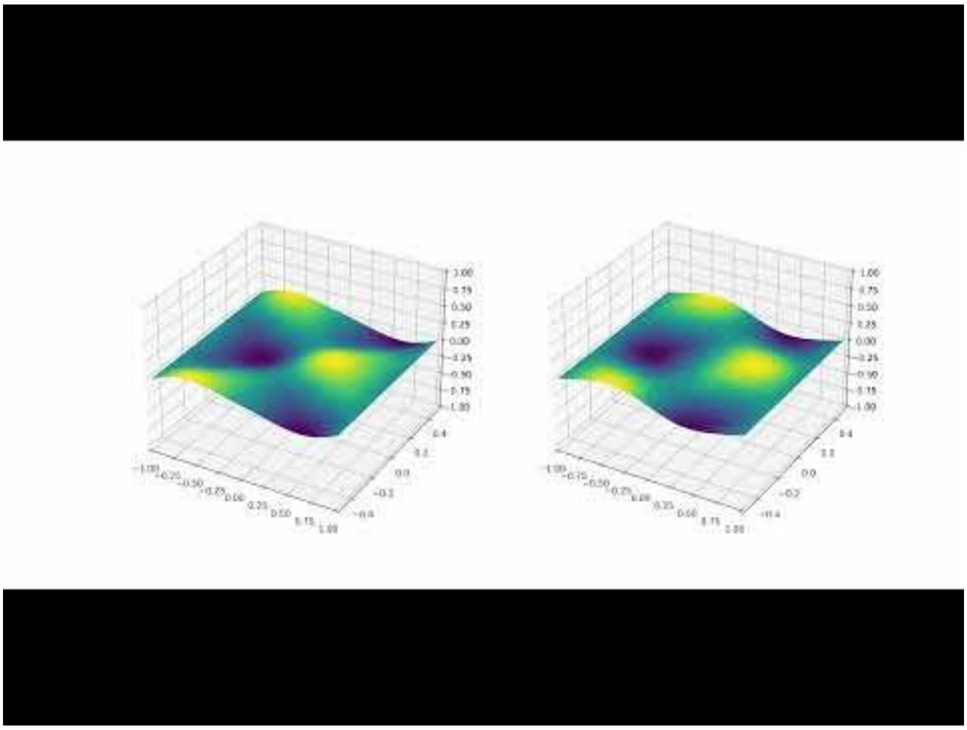
5. 100 тысяч временных слоев

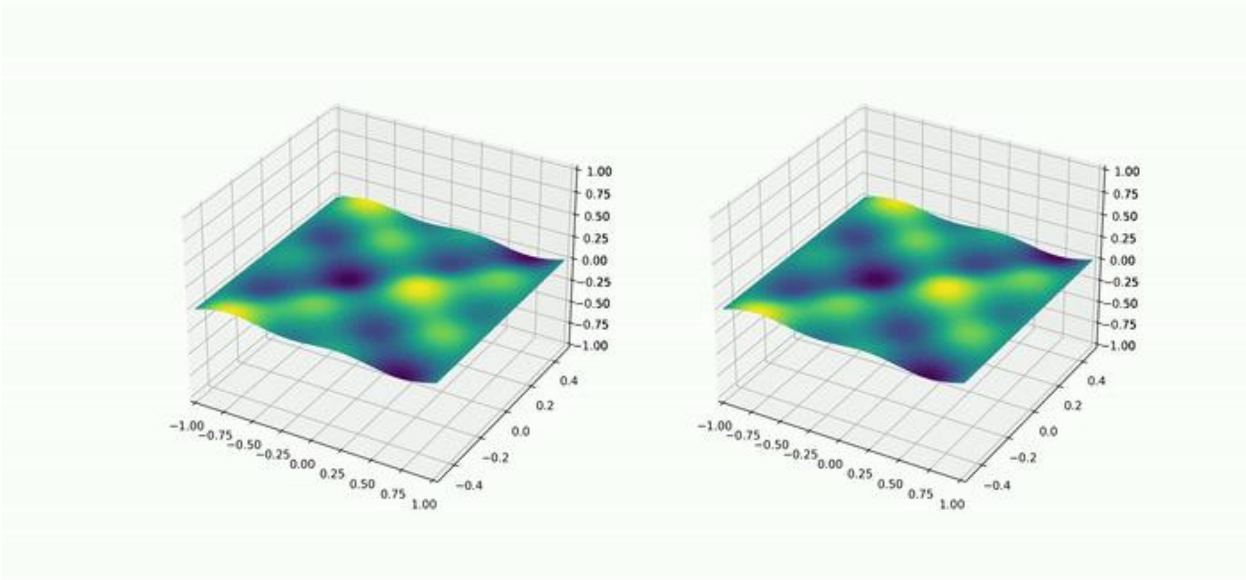


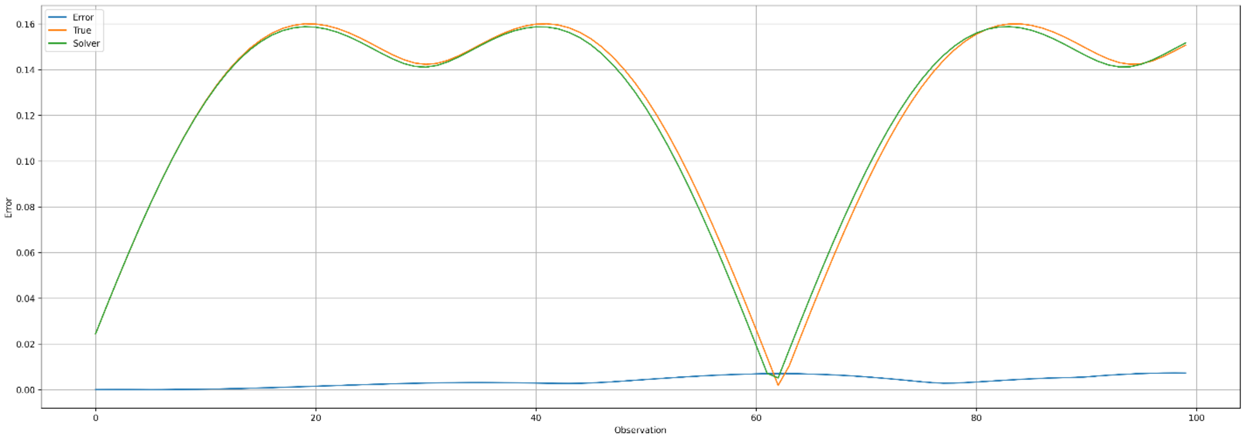
6. 100 тысяч временных слоев (только погрешность)

На графиках видно, что происходит накопление ошибки по времени.

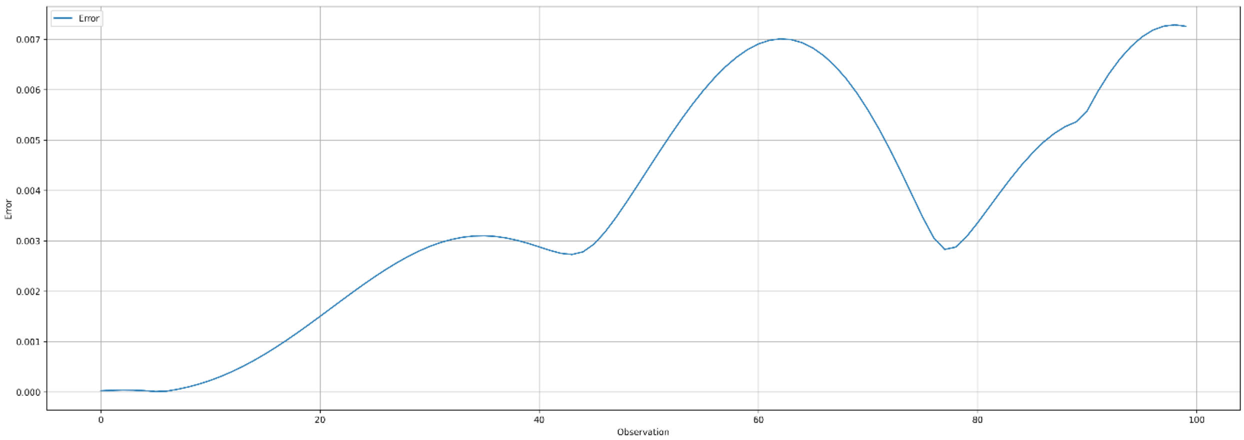
5.2. Второй тестовый пример



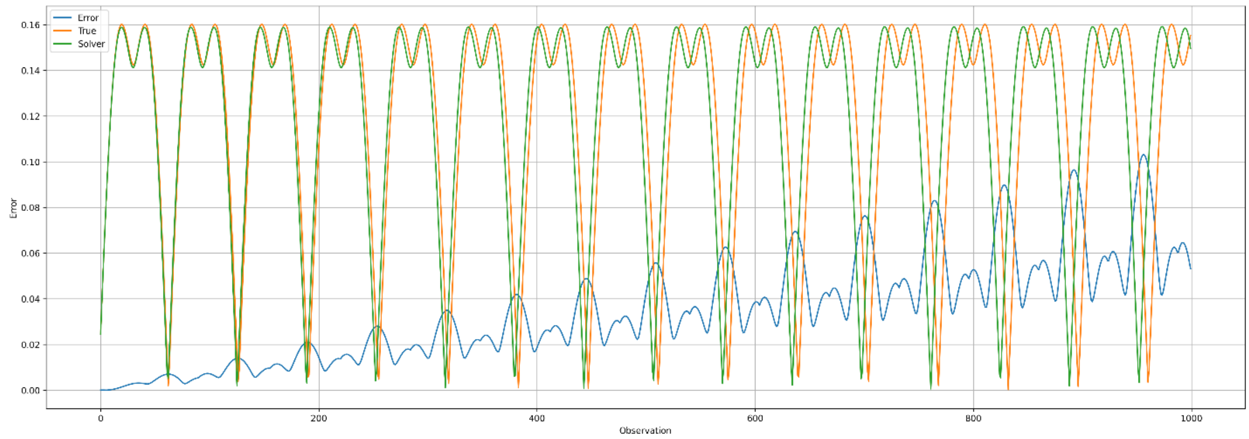




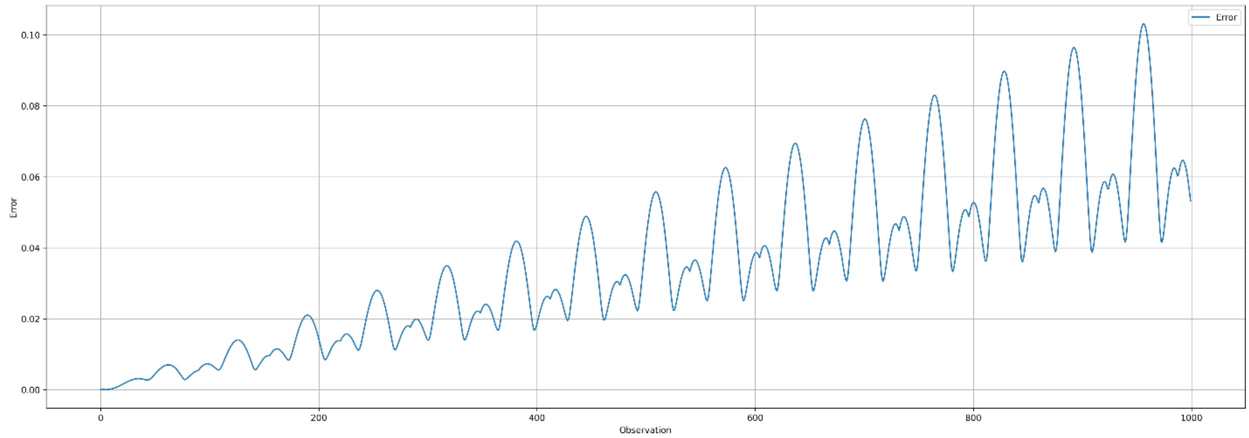
7. 100 временных слоев



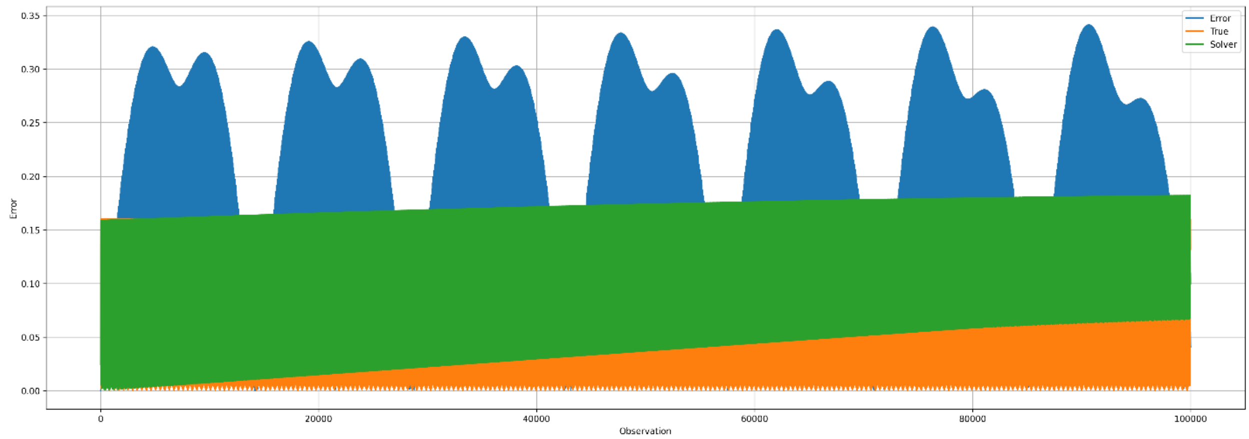
8. 100 временных слоев (только погрешность)



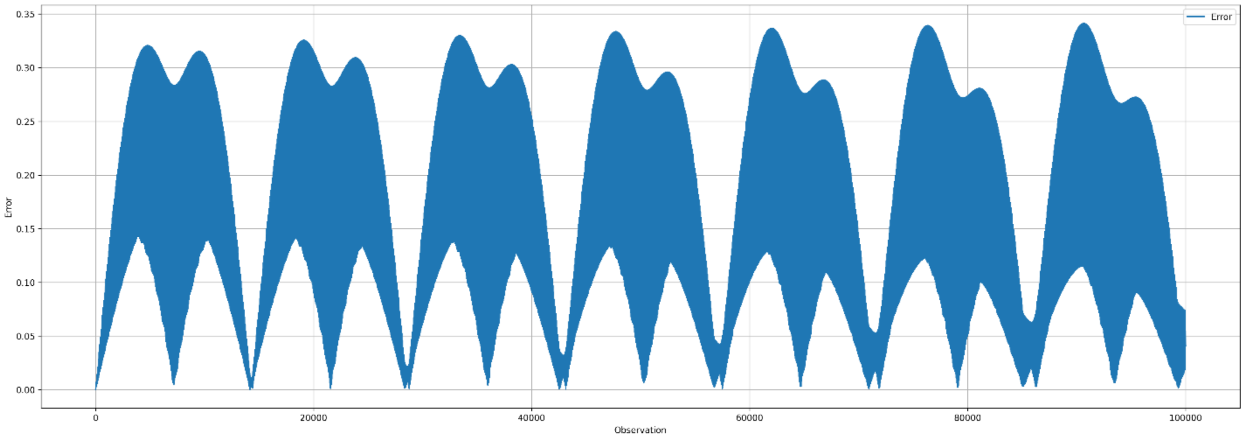
9. 1000 временных слоев



10. 1000 временных слоев (только погрешность)

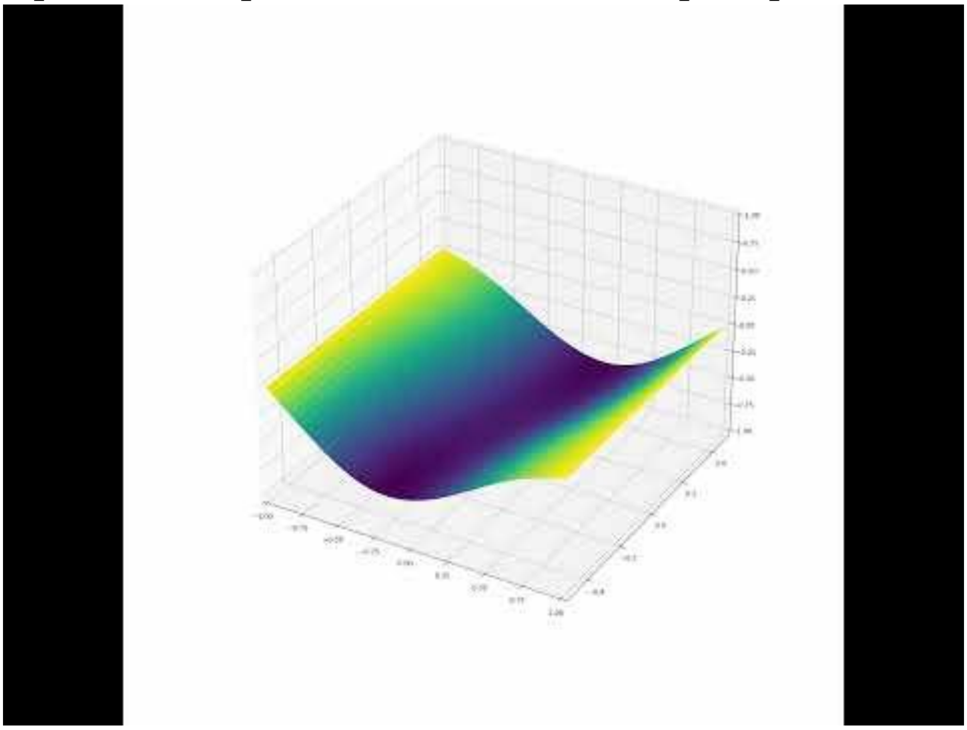


11. 100 тысяч временных слоев



12. 100 тысяч временных слоев (только погрешность)

6. Применение разностной схемы для примера задачи



7. Анализ полученных результатов

По графикам погрешности и построенным анимациям заметим, что наблюдается расхождение в решениях (точное решение имеет меньший период). Такое происходит, возможно, из-за накапливания вычислительной погрешности и погрешности аппроксимации. При этом заметим, что форма колебаний одинакова при решении тестового примера и на самом тестовом примере. Таким образом, решена задача о колебаниях пластины без учета потерь на трение с использованием метода конечных разностей.

8. Код с комментариями

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import time
import tqdm
from IPython import display
from moviepy.editor import VideoClip
from moviepy.video.io.bindings import mplfig_to_npimage
plt.rcParams['figure.dpi'] = 300
```

```
# первое начальное условие из примера задачи u(x, y, 0):
def phi(x, y, a, b):
    return np.arctan(np.cos(np.pi * x / a))

# второе начальное условие из примера задачи dudt(x, y, 0):
def psi(x, y, a, b):
    return np.sin(2 * np.pi * x / a) * np.sin(np.pi * y / b)

# тестовая функция (первое начальное условие - u(x, y, 0))
def testphi(x, y, a, b):
    return np.zeros((x.shape[0], y.shape[1]))

# тестовая функция (второе начальное условие - dudt(x, y, 0))
def testpsi(x, y, a, b):
    n = m = 1
    nu = 2 * np.pi * n / a
    mu = 2 * np.pi * m / b
    # lambda = nu + mu
    k = np.sqrt(nu ** 2 + mu ** 2)
    return np.sin(nu * x) * np.cos(mu * y) # тестовый пример 1
    # return np.sin(nu * x) * np.cos(mu * y) + np.sin(2 * nu * x) * np.cos(2 * mu
* y)

# тестовый пример - функция u(x, y, t)
def testU(x, y, t, a, b):
    n = m = 1
    nu = 2 * np.pi * n / a
    mu = 2 * np.pi * m / b
    # lambda = nu + mu
    k = np.sqrt(nu ** 2 + mu ** 2)
    return (1 / k) * np.sin(nu * x) * np.cos(mu * y) * np.sin(k * t) # тестовый
пример 1
    # return (1 / k) * np.sin(nu * x) * np.cos(mu * y) * np.sin(k * t) + (1 /
(2*k)) * np.sin(2 * nu * x) * np.cos(2 * mu * y) * np.sin(2 * k * t) # тестовый
пример 2
```

```python
# класс, решающий задачу
class SolverVec:
    u = None
    def __init__(self, a, b, step, ut0, dudt0):
        self.a = np.float64(a)
        self.b = np.float64(b)
        self.dx = np.float64(step)
        self.dy = np.float64(step)
        self.nodes_x = int(self.a / self.dx)
        self.nodes_y = int(self.b / self.dy)
        x = np.linspace(-self.a/2, self.a/2, self.nodes_x, dtype='float64')
        y = np.linspace(-self.b/2, self.b/2, self.nodes_y, dtype='float64')
        self.x, self.y = np.meshgrid(x, y)
        print(self.x.shape)
        print(self.y.shape)
        self.I = ut0
        self.V = dudt0
        self.invdxsq = 1 / (self.dx ** 2)
        self.invdysq = 1 / (self.dy ** 2)

    # вторая производная по x
    def uxx(self, first_step):
        u = self.u0 if first_step else self.u1
        uxx = np.zeros((u.shape[0], u.shape[1]))
        uxx[:, 1:-1] = (u[:, :-2] - 2 * u[:, 1:-1] + u[:, 2:]) * self.invdxsq
        return uxx

    # вторая производная по y
    def uyy(self, first_step):
        u = self.u0 if first_step else self.u1
        uyy = np.zeros((u.shape[0], u.shape[1]))
        uyy[0, :] = (2 * u[1, :] - 2 * u[0, :]) * self.invdysq
        uyy[-1, :] = (2 * u[-2, :] - 2 * u[-1, :]) * self.invdysq
        uyy[1:-1, :] = (u[:-2, :] - 2 * u[1:-1, :] + u[2:, :]) * self.invdysq
        return uyy

    # заполнение слоев t=0 и t=dt (используя начальные условия)
    def first_step(self, dt):
        self.u0 = self.I(self.x, self.y, self.a, self.b)
        uxx = self.uxx(True)
        uyy = self.uyy(True)
        psi = self.V(self.x, self.y, self.a, self.b)
        self.u1 = self.u0 + dt * psi + 0.5 * dt**2 * (uxx + uyy)

    # расчет нового временного слоя с шагом dt
    def advance(self, dt):
        if self.u is not None:
            self.u0 = self.u1
            self.u1 = self.u
```

```python
        uxx = self.uxx(False)
        uyy = self.uyy(False)
        self.u = 2 * self.u1.copy() - self.u0.copy() + dt**2 * (uxx + uyy)
        return self.u
```

```python
a = 2
b = 1
h = 0.01
dt = round(h*h/np.sqrt(h**2 + h**2), 7) - (1 * 1e-4)

print(f'Шаг по сетке: {h}')
print(f'Шаг по времени: {dt}')
```

```python
# инициализация класса и расчет первых двух слоев
solverVec = SolverVec(a, b, h, testphi, testpsi)
solverVec.first_step(dt)
```

```python
# определяем погрешность
errList = []
maxU = []
maxSolv = []
observations = 1000
t = 2 * dt
for i in tqdm.trange(2, observations + 2):
    u = solverVec.advance(dt)
    uTrue = testU(solverVec.x, solverVec.y, t, a, b)
    err = np.max(np.abs(u - uTrue))
    errList.append(err)
    maxU.append(np.max(np.abs(uTrue)))
    maxSolv.append(np.max(np.abs(u)))
    t += dt
```

```python
# отрисовка графика погрешности, максимума модуля точного и приближенного решения
fig, axs = plt.subplots(nrows=1, ncols=1, constrained_layout=True, figsize=(20,
7))
axs.plot([i for i in range(observations)], [errList[i] for i in
range(observations)], label='Error')
axs.plot([i for i in range(observations)], [maxU[i] for i in
range(observations)], label='True')
axs.plot([i for i in range(observations)], [maxSolv[i] for i in
range(observations)], label='Solver')
axs.set_xlabel('Observation')
axs.set_ylabel('Error')
axs.grid()
axs.legend()
plt.show()
```

```
# создание анимации: приближенное решение
solverVec = SolverVec(a, b, h, phi, psi)
solverVec.first_step(dt)

# определение количества кадров, длительности и прореживания
duration = 40
fps = 30
iterations = fps * duration + 1
drop = 4 # добавляем только каждый 4й слой

time_cube_solver = np.zeros((iterations, solverVec.x.shape[0],
solverVec.x.shape[1]))
start = time.time()
for i in tqdm.trange(iterations):
    curU = solverVec.advance(dt)
    if i % drop == 0: time_cube_solver[i//drop] = curU
print('Среднее время итерации:', (time.time() - start) / iterations)

# figsize - 1 == 72px -> 20 - 1440p / 15 - 1080p
fig = plt.figure(figsize=(15, 15))
ax = fig.add_subplot(projection='3d')


idx = 0
# каждая отрисовка ~ <=40мс
def make_frame(t):
    global idx
    ax.clear()
    ax.set_xlim(-a/2, a/2)
    ax.set_ylim(-b/2, b/2)
    ax.set_zlim(-1, 1)
    ax.plot_surface(solverVec.x, solverVec.y, time_cube_solver[idx],
cmap='viridis', edgecolor='none')
    npimg = mplfig_to_npimage(fig)
    idx += 1
    return npimg


animation = VideoClip(make_frame, duration=duration/drop)
animation.write_videofile(f'final{1}.mp4', fps=fps, codec='mpeg4', audio=False,
bitrate='6M', threads = 12, preset='ultrafast')
```

```
# создание анимации: точное и приближенное решение
solverVec = SolverVec(a, b, h, testphi, testpsi)
solverVec.first_step(dt)

# определение количества кадров, длительности и прореживания
duration = 40
fps = 30
iterations = fps * duration + 1
```

```
drop = 4 # добавляем только каждый 4й слой

time_cube = np.zeros((iterations, solverVec.x.shape[0], solverVec.x.shape[1]))
start = time.time()
t = dt*2
for i in tqdm.trange(iterations):
    if i % drop == 0: time_cube[i//drop] = testU(solverVec.x, solverVec.y, t, a, b)
    t += dt
print('Среднее время итерации:', (time.time() - start) / iterations)

time_cube_solver = np.zeros((iterations, solverVec.x.shape[0], solverVec.x.shape[1])) # len(solverVec.y), len(solverVec.x)
start = time.time()
for i in tqdm.trange(iterations):
    curU = solverVec.advance(dt)
    if i % drop == 0: time_cube_solver[i//drop] = curU
print('Среднее время итерации:', (time.time() - start) / iterations)

# figsize - 1 == 72px -> 20 - 1440p / 15 - 1080p
fig, (ax, ax2) = plt.subplots(ncols=2, figsize=(15, 7), subplot_kw={'projection': '3d'})

idx = 0
# функция отрисовки графика (~ <=40мс)
def make_frame(t):
    global idx
    ax.clear()
    ax.set_xlim(-a/2, a/2)
    ax.set_ylim(-b/2, b/2)
    ax.set_zlim(-1, 1)
    ax.plot_surface(solverVec.x, solverVec.y, time_cube[idx], cmap='viridis', edgecolor='none')
    ax2.clear()
    ax2.set_xlim(-a/2, a/2)
    ax2.set_ylim(-b/2, b/2)
    ax2.set_zlim(-1, 1)
    ax2.plot_surface(solverVec.x, solverVec.y, time_cube_solver[idx], cmap='viridis', edgecolor='none')
    npimg = mplfig_to_npimage(fig)
    idx += 1
    return npimg


animation = VideoClip(make_frame, duration=duration/drop)
animation.write_videofile(f'compare_final{3}.mp4', fps=fps, codec='mpeg4', audio=False, bitrate='6M', threads = 12, preset='ultrafast')
```

9. Источники:

А.Н.Тихонов, А.А.Самарский Уравнения математической физики. с. 426

И.Г.Араманович В.И.Левин. Уравнения математической физики Москва 1969. с.120

А.А.Амосов, Ю.А.Дубинский. Н.В.Копченова. Вычислительные методы для инженеров. Москва 1994, с. 366

Hans Petter Langtangen, Finite difference methods for wave motion, Oslo, 2016, c.81