数值天气预报课程作业

吴浩杰 20221170212

1 实习七

一维线性平流方程的两时间层积分方案 3

1.1 实习目的

通过编程实现用时间中央差分格式积分一维线性平流方程，使学生掌握时间中央差分格式的计算方法以及时间中央差分格式稳定性分析的方法。

1.2 实习内容

(1) 采用如下初始条件：

$$u(x,0) = \sin(\pi x), \quad 0 \le x \le 1 \tag{1}$$

参数设置：

$$c = 1.5$$

$$\Delta x = 0.05$$

$$\Delta t = 0.004$$

用中央差分格式积分一维线性平流方程。

(2) 分析差分格式的稳定性。

Listing 1: 实习内容

```
import numpy as np
import matplotlib.pyplot as plt

# ———————————————— 支持中文显示 ————————————————
plt.rcParams['font.sans-serif'] = ['SimHei']        # 用于显示中文标签
```

```
    plt.rcParams['axes.unicode_minus'] = False          # 用于正常显示负号

def time_integration_central(u, c, dx, dt):
    """
    中心差分格式积分一维线性平流方程:
      (u_i^{n+1} - u_i^{n-1})/(2dt) + c*(u_{i+1}^n - u_{i-1}^n)/(2dx) = 0

    首步用显式半步中心差分，后续用中心时差 + 中心空差。

    参数
    ----
    u  : ndarray, shape (nlon, ntime)
         储存积分结果，u[:,0] 已给定初值
    c  : 浮点数，流速
    dx : 浮点数，空间步长
    dt : 浮点数，时间步长

    返回
    ----
    u  : ndarray, shape (nlon, ntime)，积分后的解
    uk : ndarray, shape (ntime,)       ，各步动能
    """
    nlon, ntime = u.shape
    uk = np.zeros(ntime)
    # 初始动能
    uk[0] = np.sum(u[:,0]**2 / 2)

    beta = c * dt / dx

    # ——— 第一步（MATLAB 中 step_id=2） ———
    for i in range(1, nlon-1):
        u[i,1] = u[i,0] - 0.5 * beta * (u[i+1,0] - u[i-1,0])
    # 两端固定边界
    u[0,1]   = u[0,0]
    u[-1,1]  = u[-1,0]
    uk[1]    = np.sum(u[:,1]**2 / 2)

    # ——— 后续步（MATLAB 中 step_id=3:ntime） ———
    for n in range(2, ntime):
        for i in range(1, nlon-1):
            u[i,n] = u[i,n-2] - beta * (u[i+1,n-1] - u[i-1,n
```

```
                    -1])
47              # 边界固定
48              u[0,n]    = u[0,n-1]
49              u[-1,n]   = u[-1,n-1]
50              uk[n]     = np.sum(u[:,n]**2 / 2)
51
52          return u, uk
53
54      def main():
55          # ——— 参数设置 ———
56          nlon  = 20
57          ntime = 3000
58          dx    = 0.05
59          dt    = 0.004
60          c     = 1.5
61
62          # 初始化 u，满足 u(i,0) = sin(pi * x_i)，其中 x_i = i *
                dx
63          u = np.zeros((nlon, ntime))
64          x = np.arange(1, nlon+1) * dx
65          u[:,0] = np.sin(np.pi * x)
66
67          # 调用积分函数
68          u, uk = time_integration_central(u, c, dx, dt)
69
70          # ——— 绘图 ———
71          plt.figure(figsize=(8,4))
72          plt.plot(np.arange(1, ntime+1), uk, '--k', linewidth
                =1.5)
73          plt.ylim(0, 20)
74          plt.xlabel('积分步数', fontsize=14)
75          plt.ylabel('动能', fontsize=14)
76          plt.title('纬向总动能随积分次数的变化', fontsize=14)
77          plt.grid(True)
78          plt.tight_layout()
79          plt.show()
80
81      if __name__ == "__main__":
82          main()
```

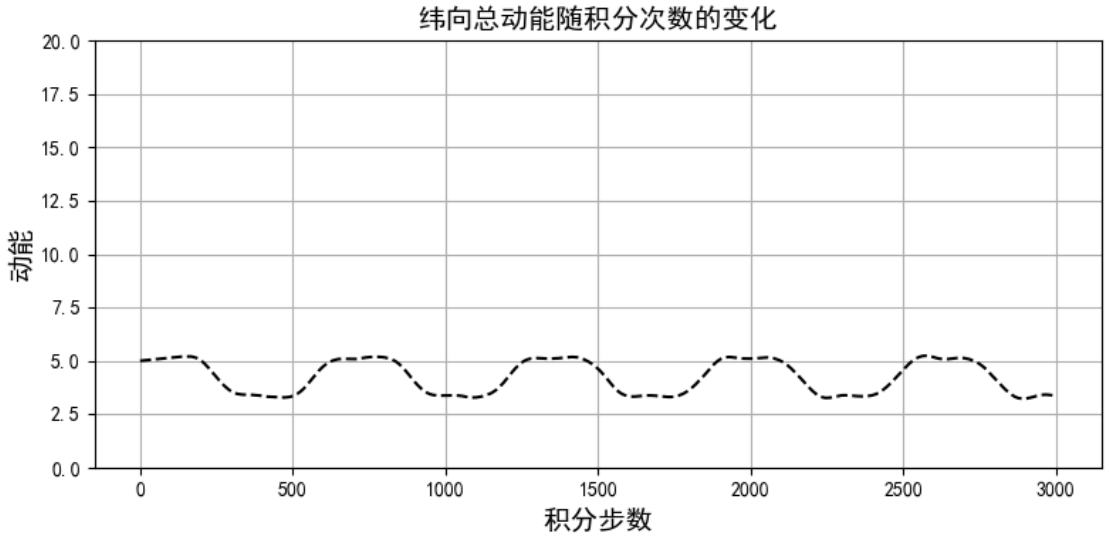

Figure 1: 实习内容

理论分析表明，在给定的初始条件下：

- $c = 1.5$
- $\Delta x = 0.05$
- $\Delta t = 0.004$

计算得到 $\beta = \frac{c\Delta t}{\Delta x} = 0.12$。由于 $|\beta| < 1$，根据稳定性分析可知该格式是稳定的。从上述数值积分结果来看，计算结果与理论分析相符合。

1.3 习题一

改变一维线性平流方程的初始条件为如下的周期边界条件，并对其稳定性进行分析。

初始条件：

$$u(x, 0) = \sin(\pi x) + 1.5, \quad 0 \leq x \leq 1 \tag{2}$$

参数设置：

$$c = 2.5$$

$$\Delta x = 0.05$$

$$\Delta t = 0.004$$

分析：改变初始条件后不影响差分格式的稳定性，只是动能积分结果有些许差异下面编写 python 代码进行分析：

Listing 2: 习题一

```
import numpy as np
import matplotlib.pyplot as plt

# ———————— 中文与负号支持 ————————
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

def central_dirichlet(u, c, dx, dt):
    """
    Dirichlet 边界下的 central (leap-frog) 格式积分一维线性平流方程。
    u: ndarray, shape=(nlon, ntime), u[:,0] 已给初值
    返回: u 完整场, uk 每步总动能
    对应 time_integration.m 中 case 'central' 的实现。:contentReference[oaicite:0]{index=0}:contentReference[oaicite:1]{index=1}
    """
    nlon, ntime = u.shape
      = c * dt / dx

    uk = np.zeros(ntime)
    uk[0] = np.sum(u[:,0]**2 / 2)

    # 启动步（step_id=2）
    for i in range(1, nlon-1):
        u[i,1] = u[i,0] - 0.5 *    * (u[i+1,0] - u[i-1,0])
    # 边界固定
    u[0,1]    = u[0,0]
    u[-1,1]   = u[-1,0]
    uk[1]     = np.sum(u[:,1]**2 / 2)

    # 正式 leap-frog 步（step_id=3:ntime）
    for n in range(2, ntime):
        # interior
        u[1:-1,n] = u[1:-1,n-2] -    * (u[2:,n-1] - u[:-2,n-1])
        # 边界固定
        u[0,n]    = u[0,n-1]
        u[-1,n]   = u[-1,n-1]
        uk[n]     = np.sum(u[:,n]**2 / 2)

    return u, uk
```

```
39
40      if __name__ == "__main__":
41          # 一 参数设置，与 xiti1.m 中相同，只改 c 和初始场 :
                contentReference[oaicite:2]{index=2}:contentReference
                [oaicite:3]{index=3}
42          nlon, ntime = 20, 3000
43          dx, dt, c    = 0.05, 0.004, 2.5
44
45            = c * dt / dx
46          print(f"Courant 数    = { :.3f} （| | 1 → 稳定）")
47
48          # 初始化
49          x = np.arange(nlon) * dx
50          u = np.zeros((nlon, ntime))
51          u[:,0] = np.sin(np.pi * x) + 1.5
52
53          # 积分 & 计算动能
54          u, uk = central_dirichlet(u, c, dx, dt)
55
56          # 绘图
57          plt.figure(figsize=(8,4))
58          plt.plot(np.arange(ntime), uk, '--k', linewidth=1.5)
59          plt.xlim(0, ntime)
60          plt.ylim(0, 100)
61          plt.xlabel('积分步数', fontsize=14)
62          plt.ylabel('动能', fontsize=14)
63          plt.title('纬向总动能随积分次数的变化', fontsize=14)
64          plt.grid(True)
65          plt.tight_layout()
66          plt.show()
```

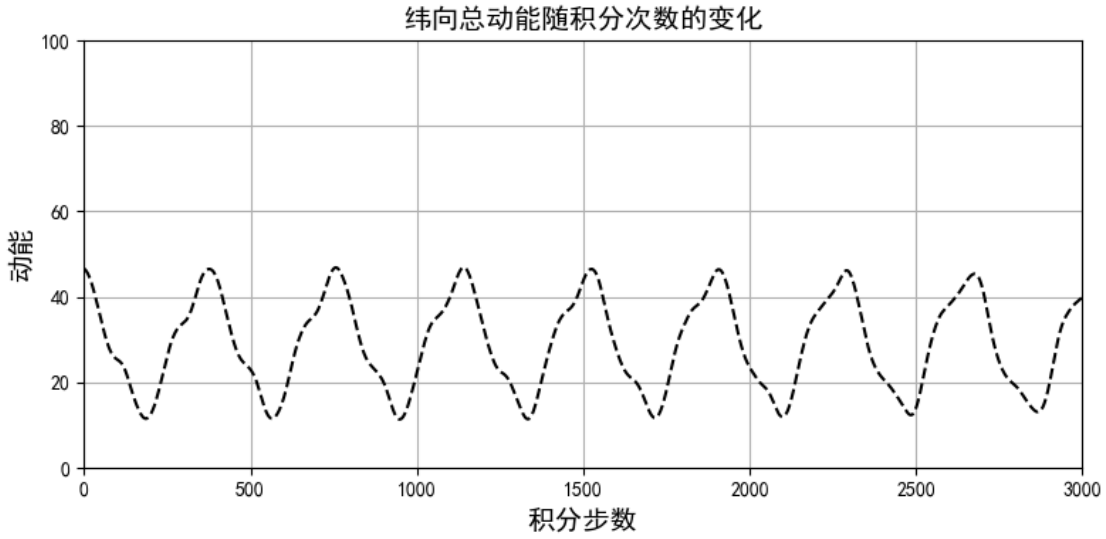

Figure 2: 习题一结果

1.4 习题二

采用周期性边界条件 $u(t,0)=u(t,1)$，对一维非线性平流方程进行积分。非线性平流方程可写为 $\frac{\partial u}{\partial t}+\frac{1}{2}\frac{\partial u^2}{\partial x}=0$，可采用以下两种差分格式：

$$u_i^{n+1}=u_i^{n-1}-\frac{\Delta t}{4\Delta x}[(u_{i+1}^n+u_i^n)^2-(u_i^n+u_{i-1}^n)^2] \tag{3}$$

$$u_i^{n+1}=u_i^{n-1}-\frac{\Delta t}{6\Delta x}(\bar{u}_{i+1}+\bar{u}_i+\bar{u}_{i-1})(\bar{u}_{i+1}-\bar{u}_{i-1}) \tag{4}$$

其中 $\bar{u}_i=\frac{u_i^{n+1}+u_i^n}{2}$。为满足线性稳定条件，选取 $\Delta x=0.1,\Delta t=0.004$。

Listing 3: 习题二

```python
import numpy as np
import matplotlib.pyplot as plt

# ———————— 支持中文和负号 ————————
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# ——— 通用参数 ———
nlon = 20
dx   = 0.1
dt   = 0.004
c    = 1.5

# 初始场: u(i,1) = sin(i• •dx/2)
x  = np.arange(1, nlon+1) * dx/2 * np.pi
```

```
16      u0 = np.sin(x)
17
18      # ---------------------------------
19      # 格式一：5500 步，日志坐标
20      # ---------------------------------
21      ntime1 = 5500
22      u1 = np.zeros((nlon, ntime1))
23      u1[:, 0] = u0
24      u1[:, 1] = u0              # MATLAB: u(:,2)=u(:,1)
25      uk1 = np.zeros(ntime1)
26      uk1[0] = uk1[1] = np.sum(u0**2/2)
27
28      coef1 = dt/(4*dx)          # =0.25*dt/dx
29
30      for n in range(2, ntime1):
31          # 内点
32          for i in range(1, nlon-1):
33              up = u1[i+1, n-1]
34              um = u1[i-1, n-1]
35              ui = u1[i  , n-1]
36              u1[i, n] = (
37                  u1[i, n-2]
38                  - coef1 * ((up + ui)**2 - (ui + um)**2)
39              )
40          # 周期边界
41          u1[0   , n] = u1[0   , n-1]
42          u1[-1  , n] = u1[-1  , n-1]
43          uk1[n] = np.sum(u1[:, n]**2/2)
44
45      # ---------------------------------
46      # 格式二：仅 3 步，线性放大
47      # ---------------------------------
48      ntime2 = 3
49      u2 = np.zeros((nlon, ntime2))
50      u2[:, 0] = u0
51      u2[:, 1] = u0
52      uk2 = np.zeros(ntime2)
53      uk2[0] = uk2[1] = np.sum(u0**2/2)
54
55      for n in range(2, ntime2):
56          # 周期边界前先算平均场
57          ubar = 0.5 * (u2[:, n-1] + u2[:, n-2])
58          for i in range(nlon):
```

```
            ip = (i+1) % nlon
            im = (i-1) % nlon
            u2[i, n] = (
                u2[i, n-2]
                - dt/(6*dx)*(ubar[ip] + ubar[i] + ubar[im])*(
                    ubar[ip] - ubar[im])
            )
        uk2[n] = np.sum(u2[:, n]**2/2)

    # ——— 绘图 ———
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12,4))

    # 左：格式一（对数纵轴）
    ax1.plot(np.arange(1, ntime1+1), uk1, '--k', linewidth=1.5)
    ax1.set_yscale('log')
    ax1.set_xlim(1, ntime1)
    ax1.set_ylim(1e-1, 1e5)
    ax1.set_xlabel('积分步数')
    ax1.set_ylabel('动能')
    ax1.set_title('格式一：总动能（对数刻度）')
    ax1.grid(True)

    # 右：格式二（放大前 3 步）
    ax2.plot(np.arange(1, ntime2+1), uk2, '--k', linewidth=1.5)
    ax2.set_xlim(1, 3)
    ax2.set_ylim(5.00, 5.00012)
    ax2.set_xlabel('积分步数')
    ax2.set_title('格式二：总动能（放大前三步）')
    ax2.grid(True)

    plt.tight_layout()
    plt.show()
```

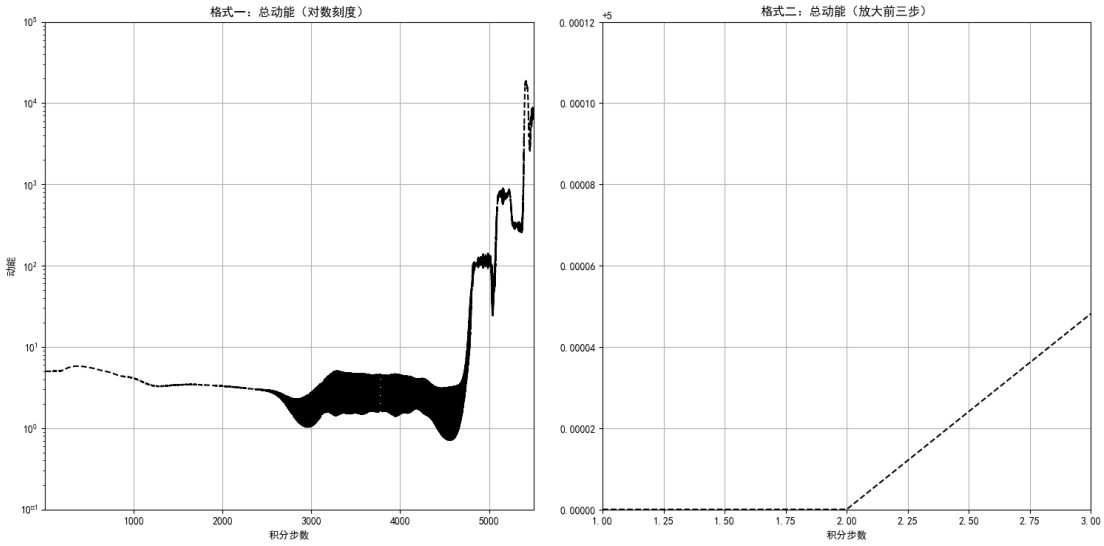

Figure 3: 习题二结果

经过理论分析可知，在该初始条件下 $|\beta| \leq |u_{max}|\frac{\Delta t}{\Delta x} = 0.1 < 1$，满足线性稳定，但格式一积分结果仍然溢出，说明非线性方程在满足线性稳定条件下仍可发生计算不稳定。格式二为隐式格式，需要解方程组，但计算量过大，故只计算出第 3 个时刻的值。