初值条件（两种）

- 条件1 $u_0^i = \sin(2\pi i\triangle x)$
- 条件2 $u_0^i = 1.5 + \sin(2\pi\triangle x)$

差分格式

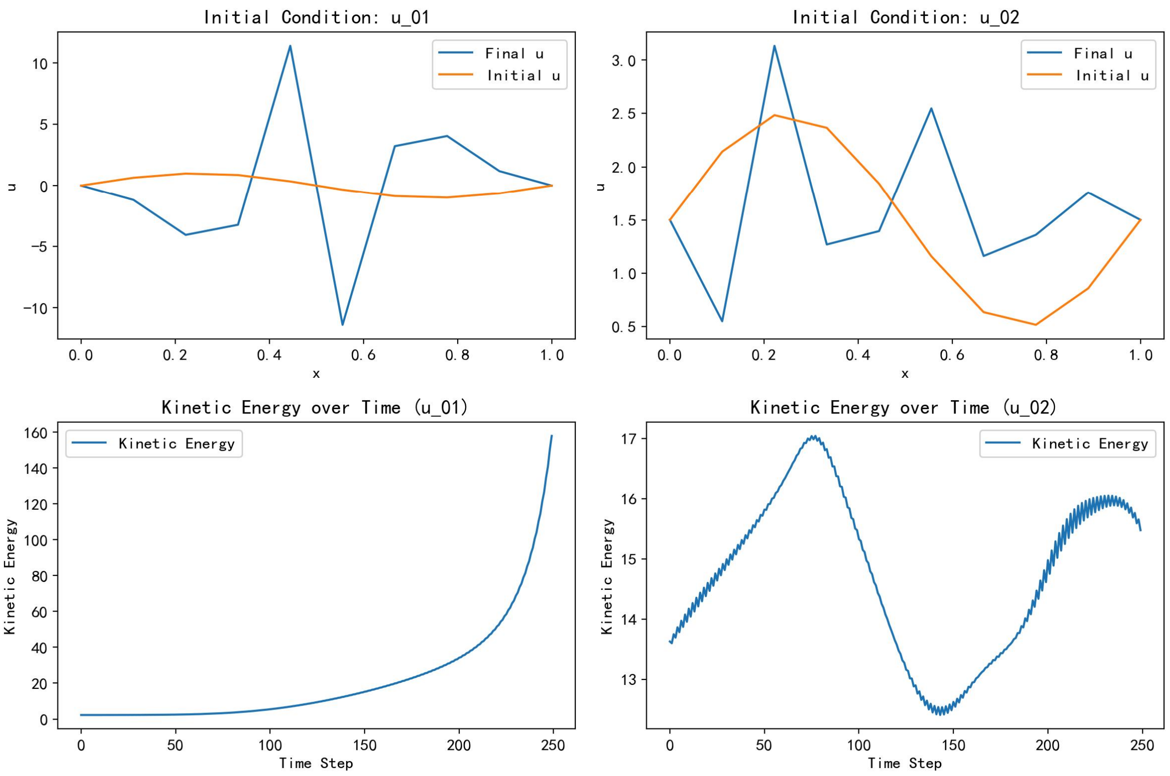
- $u_i^{n+1} = u_i^{n-1} - \frac{1}{4}\frac{\Delta t}{\Delta x}\left[(u_{i+1}^n + u_i^n)^2 - (u_i^n + u_{i-1}^n)^2\right]$

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
mpl.rcParams['font.size'] = 12
mpl.famaly = 'Times New Roman'
# dpi
mpl.rcParams['figure.dpi'] = 200
```

In [2]:

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

# 参数设置
nx = 10  # 空间网格点数
nt = 250  # 时间步数
dx = 1 / nx  # 空间步长
dt = 0.004  # 时间步长
x = np.linspace(0, 1, nx)  # 空间网格

# 初始条件
u_01 = np.sin(2 * np.pi * x)  # 初值 u_i^p
u_02 = 1.5 + np.sin(2 * np.pi * x)  # 初值 u_i^q

# 定义一个函数来计算时间积分
def time_integration(u, u_prev, nt, dt, dx):
    kinetic_energy = []
    for n in range(nt):
        u_new = u.copy()
        for i in range(1, nx - 1):
            # 计算差分方程
            term = (u[i + 1] + u[i])**2 - (u[i] + u[i - 1])**2
            u_new[i] = u_prev[i] - 0.25 * (dt / dx) * term

        # 更新上一时刻和当前时刻的值
        u_prev = u.copy()
        u = u_new.copy()
        kinetic_energy.append(0.5 * np.sum(u**2))
    return u, kinetic_energy

# 对第一个初始条件 u_01 进行计算
u_01_final, kinetic_energy_01 = time_integration(u_01.copy(), u_01.copy(), nt, d

# 对第二个初始条件 u_02 进行计算
u_02_final, kinetic_energy_02 = time_integration(u_02.copy(), u_02.copy(), nt, d
# 绘制结果
plt.figure(figsize=(12, 8))

# 第一列：初始条件 u_01 的结果
plt.subplot(2, 2, 1)
plt.plot(x, u_01_final, label='Final u')
plt.plot(x, u_01, label='Initial u')
plt.title('Initial Condition: u_01')
plt.xlabel('x')
plt.ylabel('u')
plt.legend()

plt.subplot(2, 2, 3)
plt.plot(range(nt), kinetic_energy_01, label='Kinetic Energy')
plt.title('Kinetic Energy over Time (u_01)')
plt.xlabel('Time Step')
plt.ylabel('Kinetic Energy')
plt.legend()

# 第二列：初始条件 u_02 的结果
plt.subplot(2, 2, 2)
plt.plot(x, u_02_final, label='Final u')
plt.plot(x, u_02, label='Initial u')
plt.title('Initial Condition: u_02')
plt.xlabel('x')
plt.ylabel('u')
plt.legend()

plt.subplot(2, 2, 4)
plt.plot(range(nt), kinetic_energy_02, label='Kinetic Energy')
plt.title('Kinetic Energy over Time (u_02)')
plt.xlabel('Time Step')
plt.ylabel('Kinetic Energy')
plt.legend()

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



初值条件

- 条件1 $u_0^i = \sin(2\pi i \triangle x)$
- 条件2 $u_0^i = 1.5 + \sin(2\pi \triangle x)$

差分格式

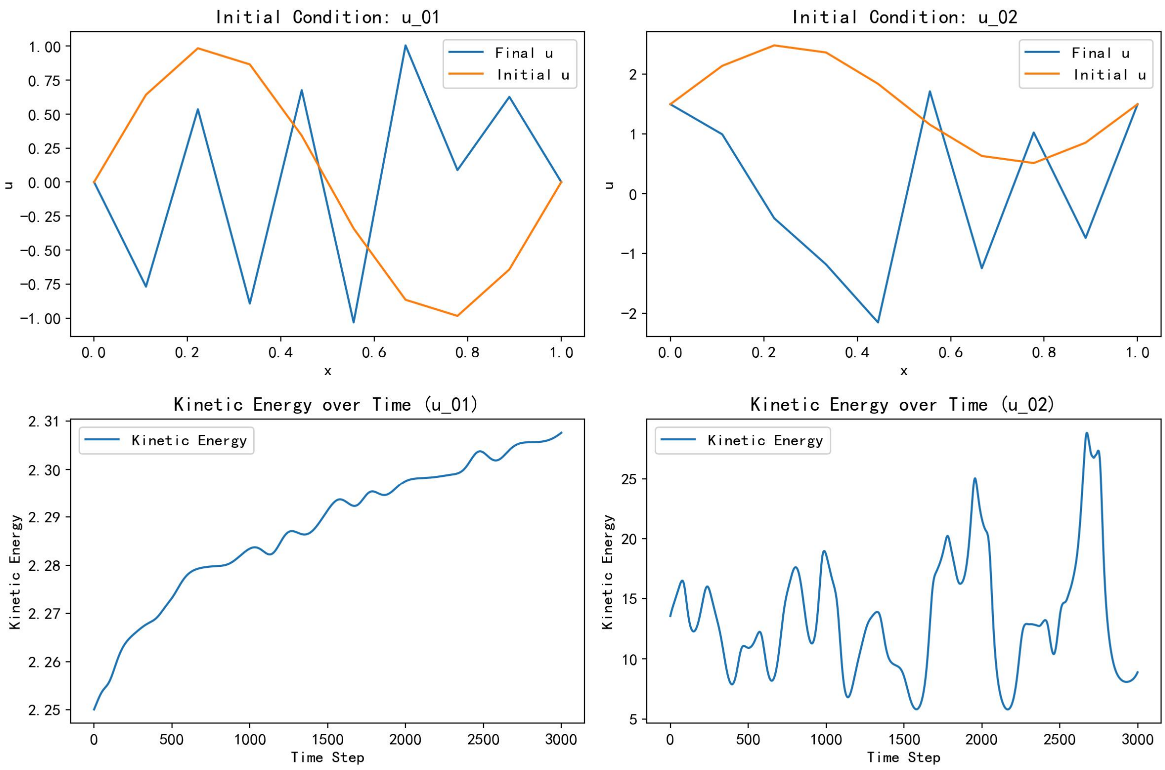
$$u_i^{n+1} = u_i^n - \frac{\Delta t}{6\Delta x}\left(\bar{u}_{i+1} + \bar{u}_i + \bar{u}_{i-1}\right)\left(\bar{u}_{i+1} - \bar{u}_{i-1}\right)$$

式中

$$\bar{u}_i = \frac{1}{2}\left(u_i^n + u_i^{n+1}\right)$$

In [ ]:

```
nx = 10  # 空间网格点数
nt = 3000  # 时间步数
dx = 0.1  # 空间步长
dt = 0.004  # 时间步长
x = np.linspace(0, 1, nx)  # 空间网格

# 初始条件
u_01 = np.sin(2 * np.pi * x)  # 初值 u_i^p
u_02 = 1.5 + np.sin(2 * np.pi * x)  # 初值 u_i^q

# 定义一个函数来计算时间积分
def time_integration(u, nt, dt, dx):
    energy = []
    for n in range(nt):
        u_new = u.copy()
        for i in range(1, nx - 1):
            # 计算中间值 \overline{u}_i
            u_bar_i = 0.5 * (u[i] + u_new[i])
            u_bar_i_plus_1 = 0.5 * (u[i + 1] + u_new[i + 1])
            u_bar_i_minus_1 = 0.5 * (u[i - 1] + u_new[i - 1])
            # 更新 u_new[i]
            u_new[i] = u[i] - (dt / (6 * dx)) * (u_bar_i_plus_1 + u_bar_i + u_ba
        u = u_new.copy()
        # 计算动能
        energy.append(0.5 * np.sum(u**2))
    return u, energy

# 对第一个初始条件 u_01 进行计算
u_01_final, energy_01 = time_integration(u_01.copy(), nt, dt, dx)

# 对第二个初始条件 u_02 进行计算
u_02_final, energy_02 = time_integration(u_02.copy(), nt, dt, dx)

# 绘制结果
plt.figure(figsize=(12, 8))

# 第一列: 初始条件 u_01 的结果
plt.subplot(2, 2, 1)
plt.plot(x, u_01_final, label='Final u')
plt.plot(x, u_01, label='Initial u')
plt.title('Initial Condition: u_01')
plt.xlabel('x')
plt.ylabel('u')
plt.legend()

plt.subplot(2, 2, 3)
plt.plot(range(nt), energy_01, label='Kinetic Energy')
plt.title('Kinetic Energy over Time (u_01)')
plt.xlabel('Time Step')
plt.ylabel('Kinetic Energy')
plt.legend()

# 第二列: 初始条件 u_02 的结果
plt.subplot(2, 2, 2)
plt.plot(x, u_02_final, label='Final u')
plt.plot(x, u_02, label='Initial u')
plt.title('Initial Condition: u_02')
plt.xlabel('x')
plt.ylabel('u')
plt.legend()

plt.subplot(2, 2, 4)
plt.plot(range(nt), energy_02, label='Kinetic Energy')
plt.title('Kinetic Energy over Time (u_02)')
plt.xlabel('Time Step')
plt.ylabel('Kinetic Energy')
plt.legend()

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


Initial Condition: u_01
Final u
Initial u
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
u
0.0
0.2
0.4
0.6
0.8
1.0
x
Initial Condition: u_02
Final u
Initial u
2
1
0
−1
−2
u
0.0
0.2
0.4
0.6
0.8
1.0
x
Kinetic Energy over Time (u_01)
Kinetic Energy
2.31
2.30
2.29
2.28
2.27
2.26
2.25
Kinetic Energy
0
500
1000
1500
2000
2500
3000
Time Step
Kinetic Energy over Time (u_02)
Kinetic Energy
25
20
15
10
5
Kinetic Energy
0
500
1000
1500
2000
2500
3000
Time Step