实验四：模拟退火算法求解函数最小值

一、实验目的

1. 掌握模拟退火算法的基本原理与实现流程。
2. 应用模拟退火算法求解函数 (f(x) = 11sin(6x) + 7cos(5x)) 在区间 (x in [0, 2*pi]) 内的最小值。
3. 分析模拟退火算法的优缺点。

二、实验原理

模拟退火算法基于物理退火过程，通过引入随机因素允许算法以一定概率接受较差解，从而跳出局部最优，趋近全局最优。核心步骤包括：

1. **冷却参数表初始化**：设置初始温度 (T_0)、衰减因子 K、迭代次数 L、初始解 (xinit) 等。
2. **新解生成**：通过当前解生成邻域新解（如随机扰动）。
3. **Metropolis 接受准则**：根据目标函数差 Δf 决定是否接受新解：
   - 若 Δf < 0（新解更优），直接接受。
   - 若 Δf > 0（新解更差），以概率 (exp(-Δf / T)) 接受。
4. **温度衰减**：逐步降低温度 T，直至满足终止条件（如温度趋近于 0）。

代码如下：

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

# 目标函数
def f(x):
    return 11 * np.sin(6 * x) + 7 * np.cos(5 * x)

# 模拟退火算法
def simulated_annealing(func, bounds, max_iter=1000, T0=100, K=0.95, T_end=1e-8):
    low, high = bounds
    x_current = np.random.uniform(low, high)  # 初始解
    x_best = x_current
    f_current = func(x_current)
    f_best = f_current

    best_history = []  # 记录最优解历史
    temp_history = []  # 记录温度历史

    for iter in range(max_iter):
        if T0 < T_end:
            break
        temp_history.append(T0)

        for _ in range(100):  # 每个温度下的迭代次数
            # 生成新解（随机扰动）
            step = 0.1 * T0  # 步长随温度衰减
            x_new = x_current + np.random.uniform(-step, step)
```

```python
            x_new = np.clip(x_new, low, high)  # 确保在区间内

            f_new = func(x_new)
            delta_f = f_new - f_current

            # Metropolis准则
            if delta_f < 0 or np.random.rand() < np.exp(-delta_f / T0):
                x_current = x_new
                f_current = f_new

                # 更新最优解
                if f_current < f_best:
                    x_best = x_current
                    f_best = f_current

        best_history.append(f_best)
        T0 *= K  # 温度衰减

    return x_best, f_best, best_history, temp_history

# 运行算法
bounds = (0, 2 * np.pi)
x_opt, f_opt, best_history, temp_history = simulated_annealing(f, bounds)

# 结果可视化
x = np.linspace(bounds[0], bounds[1], 1000)
plt.figure(figsize=(10, 6))

# 绘制函数曲线
plt.subplot(2, 1, 1)
plt.plot(x, f(x), label='目标函数')
plt.scatter(x_opt, f_opt, color='red', s=100, label=f'最优解: x={x_opt:.4f}, f(x)={f_opt:.4f}')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('目标函数曲线与最优解')
plt.legend()

# 绘制收敛过程
plt.subplot(2, 1, 2)
plt.plot(temp_history, best_history, label='温度-最优解收敛曲线')
plt.xlabel('温度 T')
plt.ylabel('最优解 f(x)')
plt.title('模拟退火收敛过程')
plt.legend()

plt.tight_layout()
plt.show()

print(f"最优解位置: x = {x_opt:.4f}")
print(f"最小值: f(x) = {f_opt:.4f}")
```

模拟退火算法优缺点

- 优点:

  - 避免局部最优：通过概率接受机制，允许逃离局部极值。
  - 初始解无关性：结果不依赖初始解，鲁棒性强。
  - 并行性：适合大规模优化问题。
- 缺点：
  - 参数敏感：温度衰减因子、迭代次数等需反复调优。
  - 计算效率低：需大量迭代以保证收敛，尤其在高维问题中耗时显著。
  - 随机性：结果存在波动，需多次运行取平均以提高可靠性。