强度计算.数值计算方法：有限元法（FEM）：弹性力学与塑性力学基础

1 绪论

1.1 有限元法的历史与发展

有限元法（Finite Element Method, FEM）起源于 20 世纪 40 年代末，最初由工程师们在解决结构工程问题时提出。1943 年，R. Courant 在解决弹性力学问题时首次使用了类似于有限元法的离散化技术。然而，直到 1956 年，O.C. Zienkiewicz 和 Y.K. Cheung 在《工程计算中的有限元法》一文中详细阐述了有限元法的基本原理，这一方法才开始被广泛接受和应用。自那时起，FEM 迅速发展，成为解决复杂工程问题的强有力工具，其应用领域从最初的结构工程扩展到流体力学、热传导、电磁学等多个领域。

1.2 FEM 在工程中的应用

有限元法在工程中的应用极为广泛，它能够处理各种复杂的几何形状、材料性质和边界条件。在结构工程中，FEM 用于预测结构在不同载荷下的响应，如应力、应变和位移，帮助工程师设计更安全、更经济的结构。在汽车工业中，FEM 用于碰撞测试，模拟车辆在不同碰撞情况下的行为，以提高车辆的安全性。在航空航天领域，FEM 用于分析飞行器在极端条件下的性能，确保其结构的稳定性和可靠性。此外，FEM 还应用于生物医学工程、土木工程、电子工程等多个领域，成为现代工程分析不可或缺的一部分。

1.3 弹性力学与塑性力学概述

1.3.1 弹性力学

弹性力学研究的是物体在外力作用下发生弹性变形的规律。当外力去除后，物体能够恢复到原来的形状和尺寸。弹性力学的基本方程包括平衡方程、几何方程和物理方程。平衡方程描述了物体内部应力的分布，几何方程描述了应变与位移的关系，物理方程则描述了应力与应变之间的关系，即材料的本构关系。在弹性力学中，材料的本构关系通常由胡克定律描述，即应力与应变成正比。

1.3.2 塑性力学

塑性力学研究的是物体在外力作用下发生塑性变形的规律。与弹性变形不同，塑性变形是不可逆的，即使外力去除，物体也无法完全恢复到原来的形状

和尺寸。塑性力学的基本方程也包括平衡方程、几何方程和物理方程，但物理方程更为复杂，需要考虑材料的塑性流动和硬化行为。在塑性力学中，材料的本构关系通常由塑性流动理论和硬化理论描述。

1.3.3 示例：使用 Python 进行弹性力学分析

下面是一个使用 Python 进行弹性力学分析的简单示例，我们使用了 numpy 和 scipy 库来解决一个简单的平面应力问题。

```
import numpy as np
from scipy.sparse import lil_matrix
from scipy.sparse.linalg import spsolve

# 定义材料属性
E = 200e9 # 弹性模量，单位：Pa
nu = 0.3  # 泊松比
t = 0.001 # 板厚，单位：m

# 定义节点和单元
nodes = np.array([[0, 0], [1, 0], [1, 1], [0, 1]]) # 节点坐标
elements = np.array([[0, 1, 2], [0, 2, 3]]) # 单元节点

# 定义边界条件
boundary_nodes = [0, 3] # 固定节点
boundary_dofs = np.concatenate([2*boundary_nodes, 2*boundary_nodes+1]) # 固定自由度

# 定义载荷
loads = np.array([0, 0, 0, 0, 0, -1e3]) # 节点载荷，单位：N

# 计算刚度矩阵
D = E/(1-nu**2) * np.array([[1, nu, 0], [nu, 1, 0], [0, 0, (1-nu)/2]])
K = lil_matrix((2*len(nodes), 2*len(nodes)), dtype=float)
for e in elements:
    x = nodes[e, 0]
    y = nodes[e, 1]
    B = np.array([[1, 0, 0, 0, 0, 0],
                  [0, 1, 0, 0, 0, 0],
                  [0, 0, 1, 0, 0, 0],
                  [0, 0, 0, 1, 0, 0],
                  [0, 0, 0, 0, 1, 0],
                  [0, 0, 0, 0, 0, 1]])
    detJ = 0.5 * np.abs((x[1]-x[0])*(y[2]-y[0]) - (y[1]-y[0])*(x[2]-x[0]))
    B = B / detJ
    Ke = D @ B.T @ B * detJ * t
    for i in range(6):
```