强度计算.结构分析：稳定性分析：8.有限元方法在结构分析中的应用

1 引言

1.1 有限元方法的历史

有限元方法（Finite Element Method, FEM）的起源可以追溯到 20 世纪初，但其真正的发展和广泛应用是在 20 世纪 50 年代末至 60 年代初。这一时期，随着计算机技术的迅速发展，有限元方法作为一种数值分析工具，开始在工程领域崭露头角。最初，它被用于解决航空工程中的复杂结构问题，如飞机机翼的应力分析。随着时间的推移，有限元方法的应用范围不断扩大，涵盖了土木工程、机械工程、生物医学工程、电子工程等多个领域。

有限元方法的理论基础主要来自于变分原理和加权残值法。1943 年，R. Courant 在解决弹性力学问题时，提出了使用分片线性函数逼近连续函数的方法，这被认为是有限元方法的早期雏形。到了 1956 年，O.C. Zienkiewicz 和 Y.K. Cheung 在解决热传导问题时，首次使用了“有限元”这一术语，标志着有限元方法的正式诞生。

1.2 有限元方法的基本原理

有限元方法是一种将连续体离散化为有限个单元的数值分析方法，通过在每个单元内建立近似解，然后将这些解组合起来，得到整个结构的近似解。这种方法的核心在于将复杂的连续问题转化为一系列简单的离散问题，从而使得计算机可以进行有效的数值求解。

1.2.1 基本步骤

1. **结构离散化**：将结构分解为有限个单元，每个单元可以是线性的、平面的或三维的，单元之间通过节点连接。
2. **选择位移模式**：在每个单元内，选择适当的函数来表示位移，通常是多项式函数。
3. **建立单元刚度矩阵**：根据选择的位移模式和材料属性，利用变分原理或加权残值法，建立每个单元的刚度矩阵。
4. **组装整体刚度矩阵**：将所有单元的刚度矩阵组装成整体结构的刚度矩阵。
5. **施加边界条件**：根据问题的物理特性，施加适当的边界条件，如固定端、自由端、载荷等。
6. **求解线性方程组**：将整体刚度矩阵和边界条件转化为线性方程组，使用数值方法求解得到结构的位移、应力和应变。

7. **后处理**：分析求解结果，如绘制位移图、应力图等，以直观地理解结构的响应。

1.2.2 示例：使用 Python 进行简单梁的有限元分析

假设我们有一根简支梁，长度为 4 米，承受均布载荷，使用 Python 和 NumPy 库进行有限元分析。

```
import numpy as np

# 定义材料属性和截面属性
E = 200e9 # 弹性模量，单位：帕斯卡
I = 0.05**4 / 12 # 惯性矩，单位：米^4
L = 4 # 梁的长度，单位：米
q = 10000 # 均布载荷，单位：牛/米

# 定义单元和节点
n_nodes = 5 # 节点数
n_elements = n_nodes - 1 # 单元数
nodes = np.linspace(0, L, n_nodes) # 节点位置
elements = np.array([(i, i+1) for i in range(n_nodes-1)]) # 单元连接

# 建立单元刚度矩阵
def element_stiffness_matrix(E, I, L):
    k = E * I / L**3 * np.array([[12, 6*L, -12, 6*L],
                                 [6*L, 4*L**2, -6*L, 2*L**2],
                                 [-12, -6*L, 12, -6*L],
                                 [6*L, 2*L**2, -6*L, 4*L**2]])
    return k

# 组装整体刚度矩阵
K = np.zeros((2*n_nodes, 2*n_nodes))
for i in range(n_elements):
    k = element_stiffness_matrix(E, I, L/n_elements)
    nodes_i = elements[i]
    K[2*nodes_i[0]:2*nodes_i[0]+2, 2*nodes_i[0]:2*nodes_i[0]+2] += k[:2, :2]
    K[2*nodes_i[0]:2*nodes_i[0]+2, 2*nodes_i[1]:2*nodes_i[1]+2] += k[:2, 2:]
    K[2*nodes_i[1]:2*nodes_i[1]+2, 2*nodes_i[0]:2*nodes_i[0]+2] += k[2:, :2]
    K[2*nodes_i[1]:2*nodes_i[1]+2, 2*nodes_i[1]:2*nodes_i[1]+2] += k[2:, 2:]

# 施加边界条件
K[0, :] = 0
K[:, 0] = 0
K[0, 0] = 1
K[-1, :] = 0
```