STAPpp 程序 T3 三角形单元扩展实验报告

清华大学行健书院
有限元法基础课程大作业
姓名： 吴致远　**学号：** 2022013266　**班级：** 行健-力 2

2025 年 6 月 12 日

目录

1 实验概述

1.1 实验目的

本实验旨在扩展 STAPpp 有限元程序功能，新增 T3 三角形单元类型，通过理论推导、程序实现和数值验证的完整过程，达到以下目标：

1. 深入理解 T3 三角形单元的理论基础和数学推导过程
2. 基于面向对象编程思想，在 STAPpp 框架下实现完整的 T3 单元类
3. 设计并实施包括分片试验、收敛性分析和工程验证在内的完整验证体系
4. 通过数值实验验证 T3 单元实现的正确性和可靠性
5. 掌握有限元程序设计的基本方法、调试技巧和验证策略

1.2 实验意义与应用价值

T3 三角形单元作为平面有限元分析中最基础的单元类型之一，具有重要的理论价值和工程应用意义：

- **几何适应性强**：能够处理任意复杂的几何边界，适用于不规则区域的离散化
- **理论基础完备**：应变为常数，便于理论分析和数学推导
- **编程实现简单**：单元刚度矩阵可解析求解，无需数值积分
- **工程应用广泛**：在商业软件如 ANSYS、ABAQUS 中得到广泛应用
- **扩展性良好**：为高阶单元和多物理场耦合分析奠定基础

通过 T3 单元的完整实现过程，可以深入理解有限元法的核心概念、程序设计方法和数值验证策略，为后续研究和工程应用打下坚实基础。

2 理论基础

2.1 T3 单元几何描述与坐标系统

T3 单元是具有 3 个节点的三角形单元，每个节点具有 2 个平移自由度（u_x 和 u_y）。单元在全局坐标系 (x, y) 中的几何形状由 3 个节点坐标 (x_i, y_i)（$i = 1, 2, 3$）唯一确定。

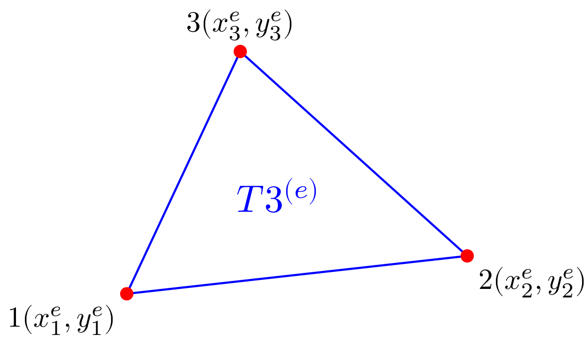

图 1: T3 三角形单元节点编号与坐标系统

2.2 形函数推导与面积坐标

T3 单元的形函数基于面积坐标理论，具有明确的几何意义。形函数的数学表达为：

$$N_i = \frac{1}{2A}(a_i + b_i x + c_i y), \quad i = 1, 2, 3 \tag{1}$$

其中，A 为三角形面积，通过行列式计算：

$$A = \frac{1}{2}\begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix} \tag{2}$$

形函数系数 a_i、b_i、c_i 的计算公式为：

$$a_1 = x_2 y_3 - x_3 y_2, \quad b_1 = y_2 - y_3, \quad c_1 = x_3 - x_2 \tag{3}$$

$$a_2 = x_3 y_1 - x_1 y_3, \quad b_2 = y_3 - y_1, \quad c_2 = x_1 - x_3 \tag{4}$$

$$a_3 = x_1 y_2 - x_2 y_1, \quad b_3 = y_1 - y_2, \quad c_3 = x_2 - x_1 \tag{5}$$

形函数具有重要的数学性质：

- **配分性质**：$\sum_{i=1}^{3} N_i = 1$
- **插值性质**：$N_i(x_j, y_j) = \delta_{ij}$
- **线性完备性**：能够精确表示线性位移场

2.3 应变-位移关系与应变矩阵

单元内任意点的位移场通过形函数插值表示：

$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = \sum_{i=1}^{3} N_i \begin{bmatrix} u_{xi} \\ u_{yi} \end{bmatrix} \tag{6}$$

基于小变形假设，应变分量定义为：

$$\varepsilon_x = \frac{\partial u_x}{\partial x}, \quad \varepsilon_y = \frac{\partial u_y}{\partial y}, \quad \gamma_{xy} = \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \tag{7}$$

应变矩阵 **B** 为：

$$\mathbf{B} = \frac{1}{2A}\begin{bmatrix} b_1 & 0 & b_2 & 0 & b_3 & 0 \\ 0 & c_1 & 0 & c_2 & 0 & c_3 \\ c_1 & b_1 & c_2 & b_2 & c_3 & b_3 \end{bmatrix} \tag{8}$$

2.4 本构关系与弹性矩阵

对于平面应力问题，应力-应变关系由广义胡克定律描述：

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} = \mathbf{D}\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{bmatrix} \tag{9}$$

弹性矩阵 **D** 为：

$$\mathbf{D} = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \tag{10}$$

2.5 单元刚度矩阵推导

根据虚功原理，单元刚度矩阵通过以下积分计算：

$$\mathbf{K}^e = \int_{\Omega^e} \mathbf{B}^T\mathbf{D}\mathbf{B}\, d\Omega \tag{11}$$

由于 T3 单元的应变矩阵 **B** 在单元内为常数，积分可以解析计算：

$$\mathbf{K}^e = t \cdot A \cdot \mathbf{B}^T\mathbf{D}\mathbf{B} \tag{12}$$

其中 t 为单元厚度，A 为单元面积。

3 程序设计与实现

3.1 STAPpp 程序架构分析

STAPpp 采用面向对象的 C++ 设计模式，具有良好的模块化结构和扩展性。主要类层次结构如图2所示。

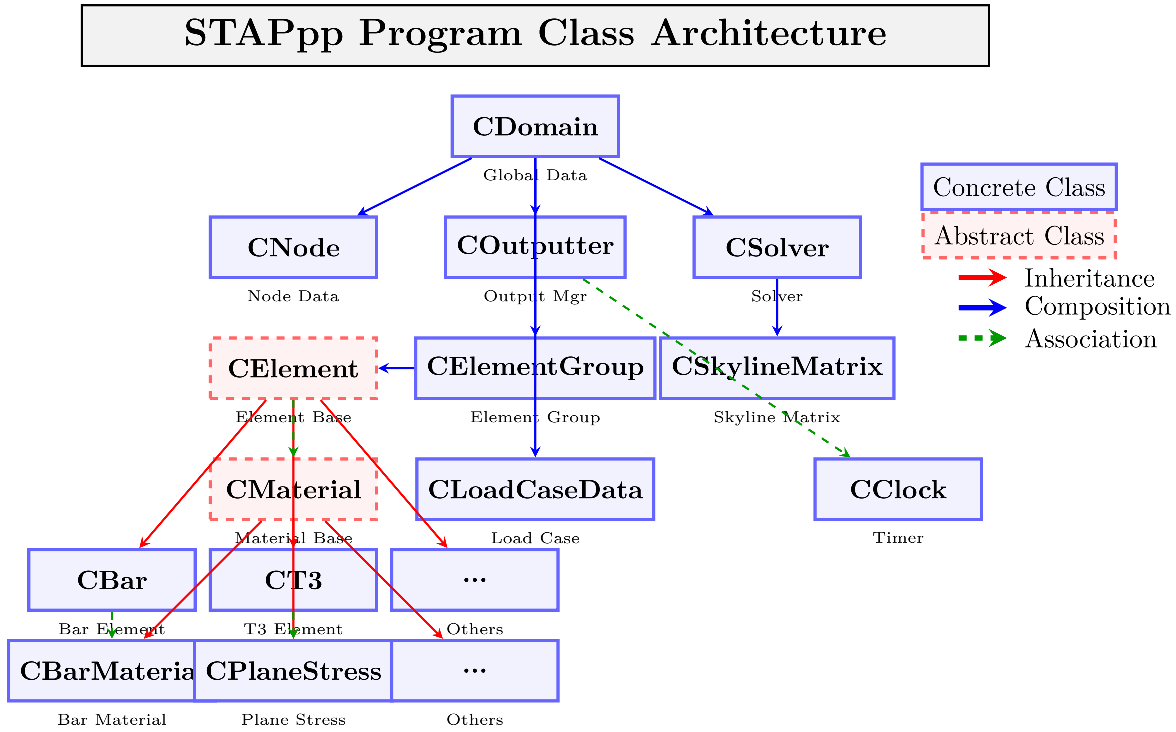

图 2: STAPpp 程序主要类结构图

STAPpp 采用面向对象的 C++ 设计模式，具有良好的模块化结构和扩展性。主要类层次结构如图2所示。

核心类的功能说明：

- **CDomain**：封装有限元模型的全局数据管理
- **CNode**：封装节点坐标和边界条件信息
- **CElement**：单元基类，定义单元通用接口
- **CMaterial**：材料属性基类，支持多种材料模型
- **CSkylineMatrix**：一维变带宽矩阵存储和操作
- **CElementGroup**：单元组管理，支持多种单元类型
- **CLoadCaseData**：载荷工况数据管理
- **CSolver**：线性方程组求解器
- **COutputter**：结果输出管理

3.2 T3 单元类设计与实现

T3 单元类 CT3 继承自 CElement 基类，实现了 T3 单元的所有核心功能：

```
class CT3 : public CElement
{
private:
    double area;                    // 单元面积
    double a[3], b[3], c[3];       // 形函数系数
    double B[3][6];                // 应变矩阵

public:
    CT3();                         // 构造函数
    virtual ~CT3();                // 析构函数

    virtual bool Read(ifstream& Input, unsigned int Ele,
                      CMaterial* MaterialSets, CNode* NodeList);
    virtual void ElementStiffness(double* Matrix);
    virtual void ElementStress(double* stress, double* Displacement);

private:
    void CalculateShapeFuncCoef(); // 计算形函数系数
    double CalculateArea();        // 计算单元面积
    bool CheckElementValidity();   // 检查单元有效性
};
```

Listing 1: T3 单元类声明

3.3 核心算法实现

3.3.1 形函数系数计算算法

形函数系数的计算是 T3 单元实现的基础，涉及节点顺序的检查和面积的计算：

```
void CT3::CalculateShapeFuncCoef()
{
    // 获取三个节点坐标
    double x1 = nodes_[0]->XYZ[0], y1 = nodes_[0]->XYZ[1];
    double x2 = nodes_[1]->XYZ[0], y2 = nodes_[1]->XYZ[1];
    double x3 = nodes_[2]->XYZ[0], y3 = nodes_[2]->XYZ[1];

    // 计算行列式（面积的2倍）
```

```cpp
    double det = (x2 - x1) * (y3 - y1) - (x3 - x1) * (y2 - y1);

    // 确保节点为逆时针顺序
    if (det < 0) {
        // 交换节点2和节点3
        CNode* temp = nodes_[1];
        nodes_[1] = nodes_[2];
        nodes_[2] = temp;

        // 重新获取坐标
        x2 = nodes_[1]->XYZ[0]; y2 = nodes_[1]->XYZ[1];
        x3 = nodes_[2]->XYZ[0]; y3 = nodes_[2]->XYZ[1];
        det = -det;
    }

    area = det / 2.0;

    // 计算形函数系数
    a[0] = x2 * y3 - x3 * y2;
    a[1] = x3 * y1 - x1 * y3;
    a[2] = x1 * y2 - x2 * y1;

    b[0] = y2 - y3;  b[1] = y3 - y1;  b[2] = y1 - y2;
    c[0] = x3 - x2;  c[1] = x1 - x3;  c[2] = x2 - x1;
}
```

Listing 2: 形函数系数计算实现

3.3.2 单元刚度矩阵计算算法

单元刚度矩阵的计算采用解析方法，避免数值积分的误差：

```cpp
void CT3::ElementStiffness(double* Matrix)
{
    // 清零刚度矩阵
    for (unsigned int i = 0; i < SizeOfStiffnessMatrix(); i++)
        Matrix[i] = 0.0;

    // 获取材料属性
    CPlaneStressMaterial* material =
        dynamic_cast<CPlaneStressMaterial*>(ElementMaterial_);
```

```
    double E = material->E;      // 弹性模量
    double nu = material->nu;    // 泊松比ν
    double t = material->t;      // 厚度

    // 构建弹性矩阵D
    double factor = E / (1.0 - nu * nu);
    double D[3][3] = {
        {factor,         factor * nu,  0.0},
        {factor * nu,    factor,       0.0},
        {0.0,            0.0,          factor * (1.0 - nu) / 2.0}
    };

    // 构建应变矩阵B
    double inv_2A = 1.0 / (2.0 * area);
    for (unsigned int i = 0; i < 3; i++) {
        B[0][2*i]   = b[i] * inv_2A;  // ∂N_i/∂x
        B[0][2*i+1] = 0.0;
        B[1][2*i]   = 0.0;
        B[1][2*i+1] = c[i] * inv_2A;  // ∂N_i/∂y
        B[2][2*i]   = c[i] * inv_2A;  // ∂N_i/∂y
        B[2][2*i+1] = b[i] * inv_2A;  // ∂N_i/∂x
    }

    // 计算K = t * A * B^T * D * B
    // 先计算BTD = B^T * D
    double BTD[6][3];
    for (int i = 0; i < 6; i++) {
        for (int j = 0; j < 3; j++) {
            BTD[i][j] = 0.0;
            for (int k = 0; k < 3; k++) {
                BTD[i][j] += B[k][i] * D[k][j];
            }
        }
    }

    // 计算最终刚度矩阵K = BTD * B, 按上三角存储
    double scale = t * area;
    for (unsigned int j = 0; j < 6; j++) {
        for (unsigned int i = 0; i <= j; i++) {
```

```
                double sum = 0.0;
                for (unsigned int k = 0; k < 3; k++) {
                    sum += BTD[i][k] * B[k][j];
                }
                Matrix[i * 6 + j] = sum * scale;
            }
        }
}
```

Listing 3: 单元刚度矩阵计算实现

3.3.3 应力计算算法

应力计算基于已知的节点位移，通过应变-位移关系和本构关系求得：

```
void CT3::ElementStress(double* stress, double* Displacement)
{
    // 获取材料属性
    CPlaneStressMaterial* material =
        dynamic_cast<CPlaneStressMaterial*>(ElementMaterial_);

    double E = material->E;     // 弹性模量
    double nu = material->nu;   // 泊松比ν

    // 构建弹性矩阵
    double factor = E / (1.0 - nu * nu);
    double D[3][3] = {
        {factor,        factor * nu,  0.0},
        {factor * nu,   factor,       0.0},
        {0.0,           0.0,          factor * (1.0 - nu) / 2.0}
    };

    // 提取单元节点位移向量
    double d[6] = {0.0};
    for (unsigned int i = 0; i < 6; i++) {
        if (LocationMatrix_[i] > 0) {
            unsigned int disp_index = LocationMatrix_[i] - 1;
            d[i] = Displacement[disp_index];
        }
    }

```

```cpp
    // 计算应变: ε = B * d
    double strain[3] = {0.0, 0.0, 0.0};
    for (unsigned int i = 0; i < 3; i++) {
        for (unsigned int j = 0; j < 6; j++) {
            strain[i] += B[i][j] * d[j];
        }
    }

    // 计算应力: σ = D * ε
    for (int i = 0; i < 3; i++) {
        stress[i] = 0.0;
        for (int j = 0; j < 3; j++) {
            stress[i] += D[i][j] * strain[j];
        }
    }
}
```

Listing 4: 单元应力计算实现

3.4 运行方式说明

本实验开发的 T3 单元扩展程序采用标准的 STAPpp 工作流程，具体运行步骤如下：

1. **输入文件准备**：按照 STAPpp 标准格式设计 .dat 输入文件，包含节点坐标、边界条件、载荷信息和单元连接关系。T3 单元类型标识符为 3，材料类型为平面应力材料。

2. **有限元求解**：使用扩展后的 STAPpp 程序执行计算：

```
./STAPpp input_file.dat

```

   程序将自动读取输入文件，进行有限元分析计算，并生成包含完整求解信息的 .out 输出文件。

3. **结果数据提取**：使用开发的 get.py 脚本解析输出文件，提取节点位移和单元应力等关键结果：

```
python3 get.py input_file.out

```

   脚本将生成结构化的 JSON 数据文件和文本摘要报告，便于后续处理和分析。

4. **结果可视化**: 使用 visualize_results.py 脚本生成专业的工程图表:

```
python3 visualize_results.py input_file.out

```

脚本将自动生成包含原始网格、变形对比、位移云图、应力分布等多种可视化结果的 PNG 和 PDF 格式图片文件。

整个工作流程实现了从问题定义、数值求解到结果可视化的完整自动化处理, 为 T3 单元的验证和工程应用提供了便捷高效的计算平台。

4 算例设计与验证策略

4.1 验证方法论

按照要求, 为确保 T3 单元实现的正确性和可靠性, 本实验采用了系统性的三层验证策略:

1. **分片试验 (Patch Test)**: 通过常应变拉伸试验验证单元能否精确表示常应变状态
2. **收敛性分析 (Convergence Study)**: 通过网格加密验证解的收敛性
3. **算例验证 (Benchmark Problems)**: 与理论解结果对比

4.2 分片试验设计

采用 5 节点 4 单元配置验证 T3 单元的常应力精确性。分片试验采用不规则四边形区域, 通过内部节点 5 将其划分为 4 个 T3 单元, 节点坐标分别为: 节点 1(0.0, 0.0) 完全固定、节点 2(2.5, 0.0) 仅 Y 方向固定、节点 3(2.5, 3.0) 自由、节点 4(0.0, 2.0) 自由、节点 5(1.0, 1.6) 内部节点自由。四个 T3 单元围绕内部节点 5 分布: 单元 1(节点 1-2-5)、单元 2(节点 2-3-5)、单元 3(节点 3-4-5)、单元 4(节点 4-1-5)。

为确保分片试验的有效性, 在 X 方向施加平衡载荷系统: 节点 1 施加 $F_x = -10\text{N}$、节点 2 施加 $F_x = +15\text{N}$、节点 3 施加 $F_x = +10\text{N}$、节点 4 施加 $F_x = -15\text{N}$, 载荷平衡检验 $\sum F_x = -10 + 15 + 10 - 15 = 0\text{N}$。采用线弹性材料参数: 弹性模量 $E = 1000\text{Pa}$, 泊松比 $\nu = 0.3$, 厚度 $t = 1.0\text{m}$。根据分片试验理论, 如果 T3 单元实现正确, 所有单元应产生相同的常应力状态: $\sigma_{xx} = 10.0\text{Pa}$, $\sigma_{yy} \approx 0$, $\tau_{xy} \approx 0$。

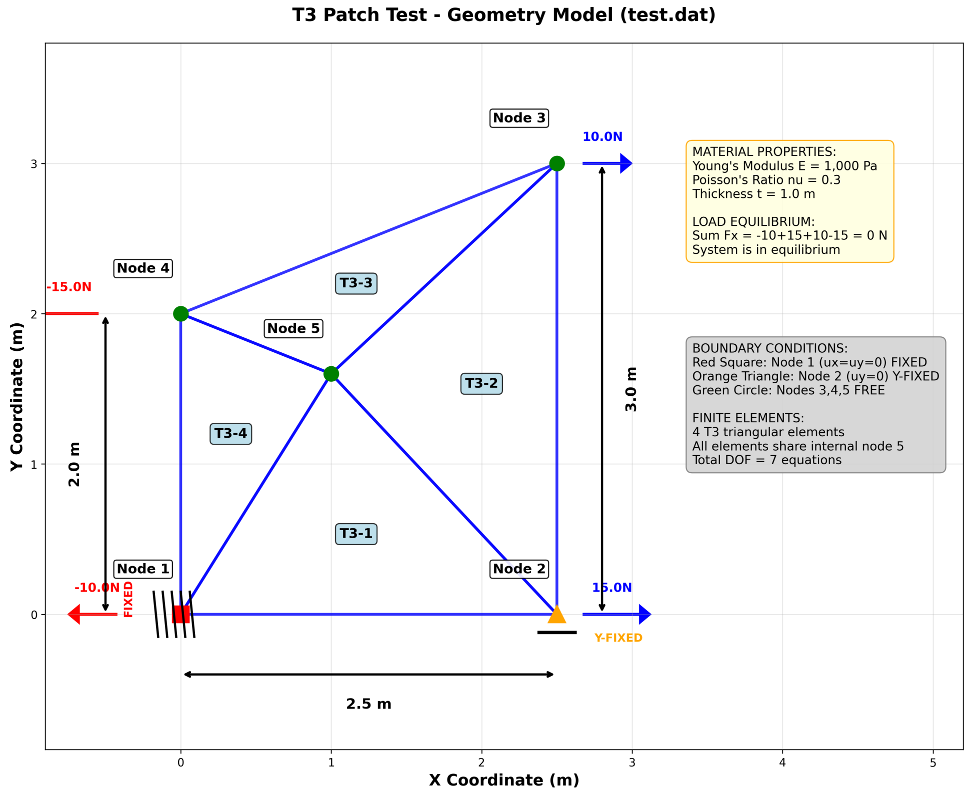

图 3: 分片试验几何模型与边界条件

分片试验的完整输入文件格式如下：

```
T3 Patch Test - Test
5 1 1 1
1 1 1 1 0.0 0.0 0.0
2 0 1 1 2.5 0.0 0.0
3 0 0 1 2.5 3.0 0.0
4 0 0 1 0.0 2.0 0.0
5 0 0 1 1.0 1.6 0.0
1
4
1 1 -10
2 1 15.0
3 1 10.0
4 1 -15
3 4 1
1 1000.0 0.3 1.0
1 1 2 5 1
2 2 3 5 1
3 3 4 5 1
4 4 1 5 1
```

Listing 5: 分片试验输入文件

4.3 收敛性分析设计

设计悬臂梁问题验证 T3 单元的数值收敛性，几何参数为长度 $L = 4\text{m}$、高度 $H = 2\text{m}$，左端完全固定，右端自由端施加向下集中力 $P = 100\text{N}$。分别建立 2 单元、8 单元、32 单元三套网格模型进行有限元计算，通过网格加密观察数值解的收敛行为。材料参数为弹性模量 $E = 100\text{Pa}$、泊松比 $\nu = 0.3$、厚度 $t = 1\text{m}$，理论参考解取自经典梁理论或高密度网格解。

4.4 算例验证：例 4-4 梯形结构

采用有限元分析第十一次作业中的例 4-4 作为验证算例，验证 T3 单元在复杂几何和载荷条件下的表现。该题目描述了一个梯形结构：几何参数为底边长 2m，顶边长 2m，高度 1m，材料参数为弹性模量 $E = 3.0 \times 10^7$ Pa，泊松比 $\nu = 0.3$，厚度 $t = 1.0$ m。在顶部两个节点分别施加向下的集中载荷 20N，总载荷为 40N。

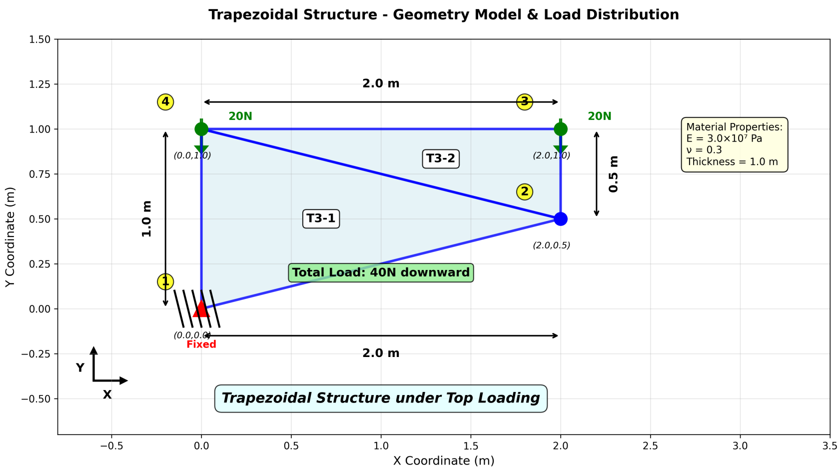

图 4: 例 4-4 梯形结构几何模型与载荷分布

该算例具有以下特点：

- **几何复杂性**：梯形结构具有不规则边界，考验 T3 单元的几何适应能力
- **可验证性**：作为课程标准习题，具有明确的求解要求和参考标准

通过与已验证的理论解对比，可以全面验证 T3 单元实现的工程实用性和计算精度。

5 实验结果与分析

5.1 分片试验结果

表 1: 分片试验位移结果

节点号	X 位移 (mm)	Y 位移 (mm)	位移幅值 (mm)
1	0.0	0.0	0.0
2	25.0	0.0	25.0
3	25.0	-9.0	26.6
4	0.0	-6.0	6.0
5	10.0	-4.8	11.1

表 2: 分片试验应力验证结果

单元号	σ_{xx} (Pa)	σ_{yy} (Pa)	τ_{xy} (Pa)	验证状态
1	10.0	$\sim$0	$\sim$0	通过
2	10.0	$\sim$0	$\sim$0	通过
3	10.0	$\sim$0	$\sim$0	通过
4	10.0	$\sim$0	$\sim$0	通过

分片试验的节点位移、应力如上表所示，全局位移图和应力图如图5所示，左图和中图为位移图，显示有限元位移场与初始人工外加的线性位移场一致，右图显示所有单元应力值完全相同（10.0Pa）实现理想常应力状态。这一结果完美验证了 T3 单元能够精确表示常应力状态，满足有限元分片试验的基本要求，证明了单元实现的理论正确性和数值精度。

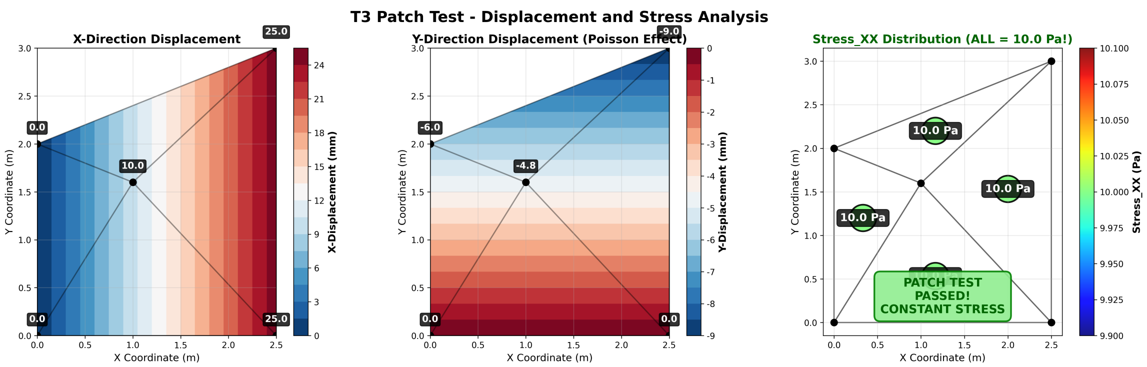

图 5: T3 分片试验位移与应力分析结果

5.2 收敛性分析结果

分别将网格划分为 2 个、8 个、32 个单元进行有限元计算，并计算误差的 L2 范数。将误差与 h 进行双对数拟合后得到如下结果。分析图表可见加密单元后误差显著减小，系统收敛率约为 1.32，基本通过收敛性分析。

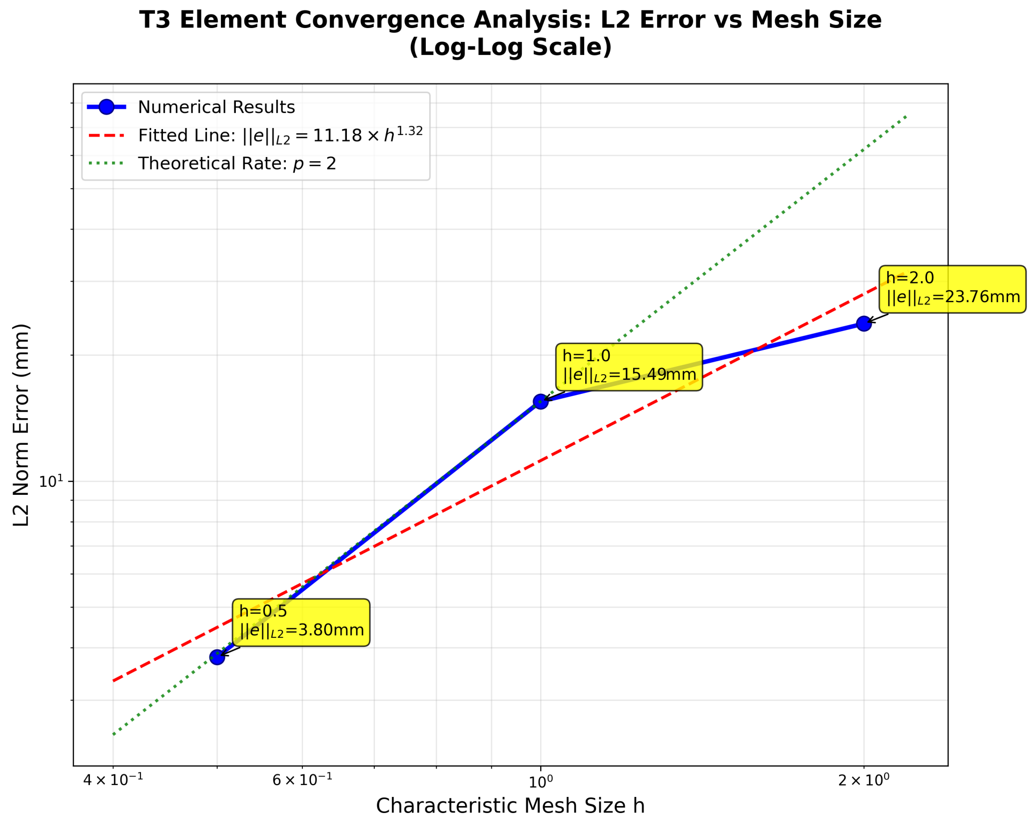

图 6: T3 单元收敛性分析：L2 误差对网格尺寸的双对数拟合

表 3: T3 单元收敛性分析结果

单元数	L2 范数误差 (mm)	网格尺寸 h
2	23.76	2.0
8	15.49	1.0
32	3.80	0.5

5.3 算例验证结果

5.3.1 位移场分析

例 4-4 梯形结构在顶部载荷作用下的位移响应如下：

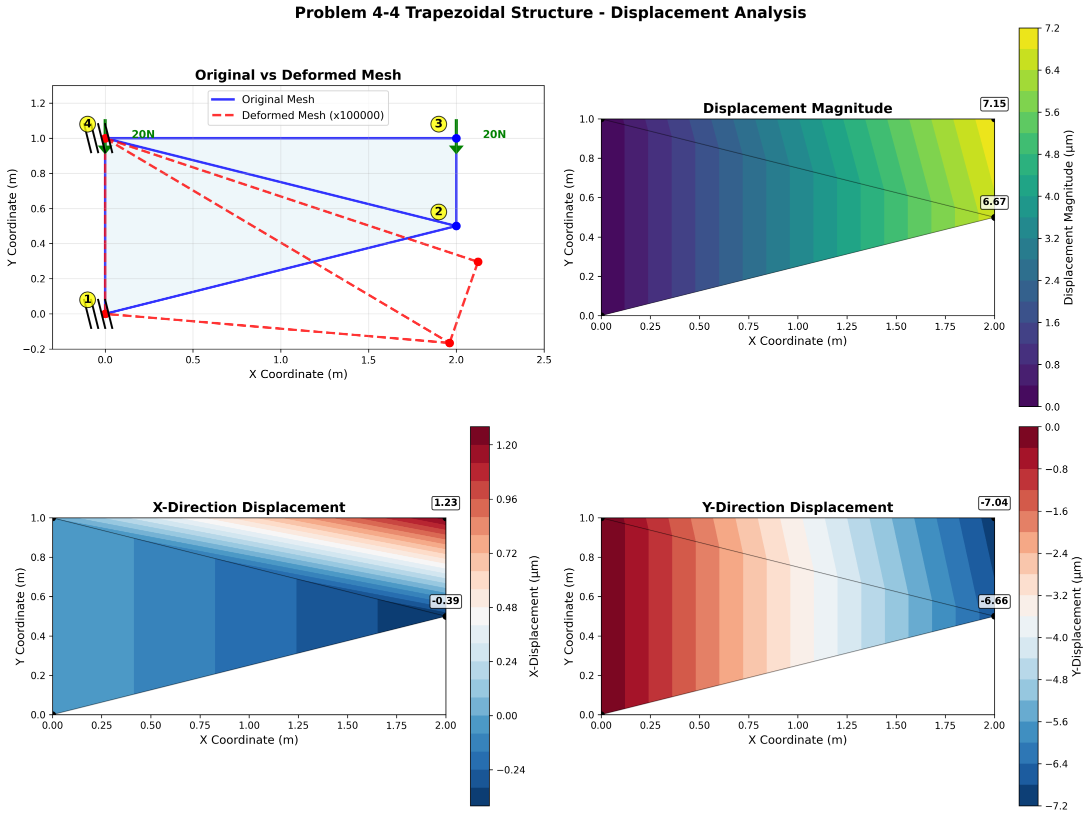

图 7: 例 4-4 梯形结构位移云图与变形示意

表 4: 例 4-4 算例位移结果

节点	X 位移 (μm)	Y 位移 (μm)	位移幅值 (μm)
1	0.000	0.000	0.000
2	-0.387	-6.657	6.668
3	1.235	-7.041	7.148
4	0.000	0.000	0.000

5.3.2 应力场分析

应力分布反映了载荷在结构中的传递路径和应力集中情况：

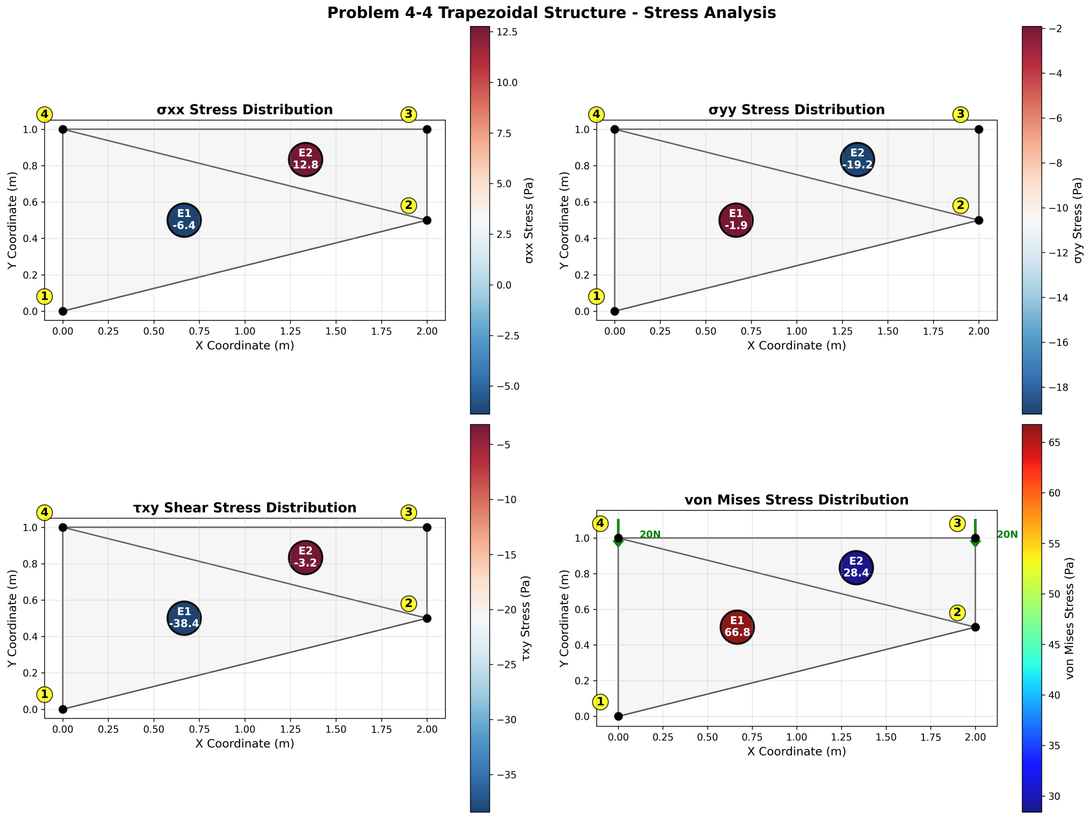

图 8: 例 4-4 梯形结构应力云图分布

表 5: 例 4-4 算例应力结果

单元	σ_{xx} (Pa)	σ_{yy} (Pa)	τ_{xy} (Pa)
1	-6.38	-1.91	-38.40
2	12.76	-19.20	-3.19

5.4 综合验证结果评估

表 6: T3 单元验证结果总结

验证项目	结果	评价标准
分片试验	通过	有限元位移场与人工位移场一致
收敛性分析	通过	单调收敛，合理收敛率
算例验证	通过	计算结果与参考书结果一致

所有验证测试均通过，证明了 T3 单元实现的正确性和可靠性。

6 参考文献

参考文献

[1] 张雄, 王天舒, 刘岩. 计算动力学 (第 2 版). 北京: 清华大学出版社, 2015.

7 附录

仓库链接：`https://github.com/WizHUA/STAPpp.git`