Journal of System Simulation

Volume 36 | Issue 5 Article 6

5-15-2024

Implementation and Numerical Simulation on Object-oriented Elastic-plastic Finite Element Method Based on Python

Henghui Li
School of Mechanical Engineering and Mechanics, Xiangtan University, Xiangtan 411105, China,
202021002587@smail.xtu.edu.cn

Yingxiong Xiao
School of Mechanical Engineering and Mechanics, Xiangtan University, Xiangtan 411105, China,
xyx610xyx@xtu.edu.cn

Follow this and additional works at: https://dc-china-simulation.researchcommons.org/journal

 Part of the Artificial Intelligence and Robotics Commons, Computer Engineering Commons, Numerical Analysis and Scientific Computing Commons, Operations Research, Systems Engineering and Industrial Engineering Commons, and the Systems Science Commons

This Paper is brought to you for free and open access by Journal of System Simulation. It has been accepted for inclusion in Journal of System Simulation by an authorized editor of Journal of System Simulation. For more information, please contact xtfzxb@126.com.

Implementation and Numerical Simulation on Object-oriented Elastic-plastic Finite Element Method Based on Python

Abstract

Abstract: With the continuous expansion of the application fields of finite element methods, higher requirements are put forward for the scalability of finite element methods. In order to overcome the defects of the traditional finite element methods, a simple and easily extensible object-oriented elasticplastic finite element program framework is proposed based on Python. Combined with the characteristics of Python, we design some finite element classes such as the pre-processing class, the post-processing class, the linear solution class, the stress integration class and the analysis class. By applying the resulting framework to several typical elastic-plastic mechanical problems and comparing the results with those calculated by ABAQUS, the correctness and effectiveness of object-oriented elasticplastic finite element program based on Python have been verified.

Keywords

object-oriented programming, elastic-plastic finite element, program framework, numerical simulation, Python

Recommended Citation

Li Henghui, Xiao Yingxiong. Implementation and Numerical Simulation on Object-oriented Elasticplastic Finite Element Method Based on Python[J]. Journal of System Simulation, 2024, 36(5): 1107-1117.

This paper is available in Journal of System Simulation: https://dc-china-simulation.researchcommons.org/journal/vol36/iss5/6

第 36 卷第 5 期 系统仿真学报© Vol. 36 No. 5
2024 年 5 月 Journal of System Simulation May 2024

基于Python的面向对象弹塑性有限元方法实现与数值模拟

李恒辉，肖映雄*
(湘潭大学 机械工程与力学学院，湖南 湘潭 411105)

摘要：随着有限元应用领域的不断扩大，对有限元的可扩充性提出了更高的要求。为克服传统有限元的缺陷，基于Python提出了一种简单、易扩充的面向对象弹塑性有限元程序框架，结合Python的特性，设计了前处理类、后处理类、线性求解类、应力积分类、分析类等有限元类。通过将程序框架应用于几类典型弹塑性问题的有限元分析与模拟，并与ABAQUS计算结果进行了对比，验证了基于Python的面向对象弹塑性有限元程序的准确性和有效性。

关键词：面向对象编程；弹塑性有限元；程序框架；数值模拟；Python
中图分类号：TP391.9；O344.3；O302　文献标志码：A　文章编号：1004-731X(2024)05-1107-11
DOI: 10.16182/j.issn1004731x.joss.23-0084
引用格式: 李恒辉, 肖映雄. 基于Python的面向对象弹塑性有限元方法实现与数值模拟[J]. 系统仿真学报, 2024, 36(5): 1107-1117.
Reference format: Li Henghui, Xiao Yingxiong. Implementation and Numerical Simulation on Object-oriented Elastic-plastic Finite Element Method Based on Python[J]. Journal of System Simulation, 2024, 36(5): 1107-1117.

Implementation and Numerical Simulation on Object-oriented Elastic-plastic Finite Element Method Based on Python

Li Henghui, *Xiao Yingxiong**
(School of Mechanical Engineering and Mechanics, Xiangtan University, Xiangtan 411105, China)

Abstract: With the continuous expansion of the application fields of finite element methods, higher requirements are put forward for the scalability of finite element methods. In order to overcome the defects of the traditional finite element methods, *a simple and easily extensible object-oriented elastic-plastic finite element program framework is proposed based on Python. Combined with the characteristics of Python, we design some finite element classes such as the pre-processing class, the post-processing class, the linear solution class, the stress integration class and the analysis class.* By applying the resulting framework to several typical elastic-plastic mechanical problems and comparing the results with those calculated by ABAQUS, the correctness and effectiveness of object-oriented elastic-plastic finite element program based on Python have been verified.

Keywords: object-oriented programming; elastic-plastic finite element; program framework; numerical simulation; Python

0 引言

由于有限元程序需要适用不同问题，因而产生了复杂多样的单元类型和本构模型，对程序的可扩充性和可维护性提出了更高的要求。传统的面向过程式的程序设计由于本身的局限性不再适用有限元的发展。这种局限性主要表现在[1]：① 即

收稿日期：2023-02-02　　修回日期：2023-04-30
基金项目：国家自然科学基金(10972191)；湖南省自然科学基金(2023JJ30569)
第一作者：李恒辉(1997-)，男，硕士生，研究方向为工程力学与计算。E-mail：202021002587@smail.xtu.edu.cn
通讯作者：肖映雄(1970-)，男，苗族，教授，博士，研究方向为计算固体力学。E-mail：xyx610xyx@xtu.edu.cn

Journal of System Simulation, Vol. 36 [2024], Iss. 5, Art. 6

使是修改代码的一小部分，也需要对整个程序有高度的了解；② 代码可重用性差；③ 数据结构的微小变化可能会对整个系统产生连锁反应；④ 设计组件之间的众多相互依赖关系是隐藏的，难以确定；⑤ 不能保证数据结构的完整性。

20 世纪 80 年代以来，面向对象程序设计逐渐出现在人们的视野中。这种编程思想将数据结构和算法封装成对象，将客观世界看成一系列相互作用、相互关联的对象，每个对象都拥有自己的属性和方法，并且具有封装性、继承性、多态性等优点。目前，已经有许多学者进行了面向对象有限元程序的研究。Forde 等[2]尝试使用面向对象的编程思想来设计有限元程序。此后，国内外许多学者基于 C++语言将面向对象编程应用于非线性有限元分析中[3-9]，如 Menétrey 等[6]介绍了面向对象有限元在 J2 塑性中的应用；Lages 等[7]提出了基于面向对象程序有限元法求解几何非线性梁模型和弹塑性 Cosserat 连续体；李会平等[8]在已有线性面向对象类的基础上进行了扩充，实现了弹塑性分析；陈飙松等[9]基于有限元分析软件系统 SiPESC.FEM，采用经典软件设计模式，提出了一种基于面向对象设计方法的通用小变形率无关弹塑性分析软件框架。

面向对象编程语言从 Smalltalk 到后来逐渐发展的 C++、Python 等。C++凭借着优异的计算性能以及与 C 语言良好的共性，长期被运用于面向对象的有限元程序设计。然而，C++语言本身的复杂性、无法内存自动管理，以及缺乏庞大的标准库和第三方库，使得程序的开发周期较长、需要构建矩阵类，并且需要开发人员有着较强的编程能力，这使得有限元程序难以扩充和维护。Python 语言是一种解释性、面向对象及动态数据类型的高级程序设计语言。它具有许多优秀的特性，如简洁易读、内存的自动管理、拥有庞大的标准库及第三方库、与其他语言(C、Fortran)良好互操作性和互用性，以及庞大的科学计算社区，还有优秀的可视化能力，已成为很受欢迎的程序设计语言之一。已经有许多有限元软件包使用 Python 语言，其中一些使用 Python 作为主要语言，例如，Fenics[10]、Firedrake[11]、Sfepy[12]等，这表明 Python 在有限元领域已经得到了广泛应用。

也有学者基于 Python 采用面向对象编程思想编制了相应的软件，如 Murat[13]基于 Python 的描述符协议提出了一种以领域特定建模语言(DSML)形式的面向对象应用程序编程接口；Miroslav 等[14]基于动态刚度单元，使用 Python 编制了 FREEVIB 面向对象软件，提出了一种板状结构自由振动分析框架。

基于 Python 的面向对象有限元程序大多是基于特别问题编制的有限元程序，对于使用 Python 的非线性有限元分析的有限元框架目前较少。本文针对弹塑性力学问题，采用面向对象编程思想对其有限元方法及实现重新进行了梳理，设计出了前处理类、后处理类、线性求解类、应力积分类、分析类等相应的有限元类，提出了一种简单、易扩充的自主可控有限元程序框架，并基于 Python 进行了实现。该程序框架由于避免了基于 C++的面向对象有限元程序中矩阵类的设计，相应的程序开发难度得到了大幅度降低。通过将该有限元程序框架应用于厚壁圆筒、土石混合体及混凝土骨料模型等几类典型问题的弹塑性有限元分析与模拟，并分别与理论解或 ABAQUS 解进行了对比分析，验证了程序框架的有效性和适应性。面向对象弹塑性有限元程序框架实现了从几何建模、划分网格、加载到提交计算、后处理等功能的一体化处理，显著提高了工作效率，其中的核心模块更加结构化，程序代码完全自主可控，非常方便验证、改正及扩充相关算法，Python 代码编写更简便，代码的可重用成分也更大，更能适应实际问题的多样性和多变性。

1 弹塑性有限元方法

弹塑性问题是固体力学中的一类非线性问题，属于材料非线性或物理非线性，其特点是应力与

https://dc-china-simulation.researchcommons.org/journal/vol36/iss5/6
DOI: 10.16182/j.issn1004731x.joss.23-0084

Li and Xiao: Implementation and Numerical Simulation on Object-oriented Elasti

应变之间不再是一一对应的单值关系，应力依赖于变形状态，且与加载历史有关，其本构关系的建立需采用增量理论。土、岩石、混凝土等具有典型的材料非线性性质。在弹塑性状态下，材料的应变应力关系是非线性的，有限元方法是求解弹塑性问题最为有效的数值方法之一。

考虑小变形弹塑性力学问题，其中位移和应变是微小量，几何方程也是线性的，相应的数学模型为

$$\begin{cases}\mu\Delta\boldsymbol{u}+(\lambda+\mu)\nabla\operatorname{div}\boldsymbol{u}=\boldsymbol{f},\ \text{in}\ \Omega\\ \boldsymbol{u}=\hat{\boldsymbol{u}},\ \text{on}\ \Gamma_{\mathrm{u}}\\ \boldsymbol{\sigma}(\boldsymbol{u})\cdot\boldsymbol{n}=\bar{\boldsymbol{T}},\ \text{on}\ \Gamma_{\mathrm{S}}\end{cases}\tag{1}$$

式中：$\Omega\subset\boldsymbol{R}^2$，$\Gamma_{\mathrm{u}}\cap\Gamma_S=\phi$，$\boldsymbol{u}=(u,v)^{\mathrm{T}}$为位移；$\boldsymbol{\sigma}=(\sigma_{ij})(i,j=1,2)$为Cauchy应力张量；$\boldsymbol{f}=(f_x,f_y)^{\mathrm{T}}$为体力；$\hat{\boldsymbol{u}}=(\hat{u},\hat{v})^{\mathrm{T}}$为边界$\Gamma_{\mathrm{u}}$上的已知位移；$\bar{\boldsymbol{T}}=(T_x,T_y)^{\mathrm{T}}$为边界$\Gamma_{\mathrm{S}}$上的面力；$\boldsymbol{n}=(n_x,n_y)^{\mathrm{T}}$为边界$\Gamma_{\mathrm{S}}$的单位外法线向量；$\lambda$和$\mu$为拉梅常数。

该受力系统的总势能为

$$\prod(\boldsymbol{u})=\int_{\Omega}\frac{1}{2}\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\sigma}t\mathrm{d}\Omega-\int_{\Omega}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{f}t\mathrm{d}\Omega-\int_{\Gamma_{\mathrm{S}}}\boldsymbol{u}^{\mathrm{T}}\bar{\boldsymbol{T}}t\mathrm{d}\Gamma\tag{2}$$

式中：t为厚度；$\boldsymbol{\varepsilon}=(\varepsilon_x,\varepsilon_y,\gamma_{xy})^{\mathrm{T}}$为应变；$\boldsymbol{\sigma}=(\sigma_x,\sigma_y,\tau_{xy})^{\mathrm{T}}$为应力。

设T^h是求解域Ω上的三角形或四边形网格剖分，记单元数为n_{e}个，其中，h为T^h上所有剖分单元的最大尺寸。利用线性元或二次元进行有限元分析，并记节点数为N_n个。设节点位移为$\boldsymbol{a}_i=[u_i,v_i]^{\mathrm{T}}$，$i=1,2,\cdots,N_n$，则位移和应变为

$$\boldsymbol{u}^h(\boldsymbol{x})=\sum_{i=1}^{N_n}\boldsymbol{N}_i(\boldsymbol{x})\boldsymbol{a}_i,\ \boldsymbol{\varepsilon}^h(\boldsymbol{x})=\sum_{i=1}^{N_n}\boldsymbol{B}_i(\boldsymbol{x})\boldsymbol{a}_i\tag{3}$$

式中：$\boldsymbol{N}_i(\boldsymbol{x})$为形函数矩阵；$\boldsymbol{B}_i(\boldsymbol{x})$为协调应变梯度矩阵。

依据塑性理论，应力应变为

$$\mathrm{d}\boldsymbol{\sigma}=\boldsymbol{D}_{\mathrm{ep}}\mathrm{d}\boldsymbol{\varepsilon}\tag{4}$$

式中：$\boldsymbol{D}_{\mathrm{ep}}$为弹塑性矩阵。

$$\boldsymbol{D}_{\mathrm{ep}}=\boldsymbol{D}_{\mathrm{e}}-\boldsymbol{D}_{\mathrm{p}}=\boldsymbol{D}_{\mathrm{e}}-\frac{\boldsymbol{D}_e\left(\dfrac{\partial F}{\partial\boldsymbol{\sigma}}\right)\left(\dfrac{\partial F}{\partial\boldsymbol{\sigma}}\right)^{\mathrm{T}}\boldsymbol{D}_{\mathrm{e}}}{\boldsymbol{A}+\left(\dfrac{\partial F}{\partial\boldsymbol{\sigma}}\right)^{\mathrm{T}}\boldsymbol{D}_{\mathrm{e}}\left(\dfrac{\partial F}{\partial\boldsymbol{\sigma}}\right)}\tag{5}$$

式中：$\boldsymbol{D}_{\mathrm{e}}$为弹性矩阵；$F=F(\sigma_{ij})$为屈服函数；$\boldsymbol{A}=-\dfrac{\partial F}{\partial\boldsymbol{\kappa}}\sigma^{\mathrm{T}}\dfrac{\partial F}{\partial\boldsymbol{\sigma}}$。

采用增量法求解弹塑性力学问题，即通过将外载荷分成若干增量步，在每个增量上计算位移的增量，然后将位移增量累加得到总的位移。假设在第m增量步的位移$\boldsymbol{a}_m$已经计算出，则可施加载荷增量$\Delta\boldsymbol{F}_m=\boldsymbol{F}_{m+1}-\boldsymbol{F}_m$，使总的载荷达到$\boldsymbol{F}_{m+1}$，于是增量方程可表示为

$$\begin{aligned}&\boldsymbol{K}_m\boldsymbol{a}_m=\Delta\boldsymbol{F}_m\\ &\boldsymbol{K}_m=\int_{\Omega}\boldsymbol{B}^{\mathrm{T}}(\boldsymbol{D}_{\mathrm{ep}})_m\boldsymbol{B}t\mathrm{d}\Omega\end{aligned}\tag{6}$$

解出$\Delta\boldsymbol{a}_m$，则第$m+1$步的解可近似表示为

$$\boldsymbol{a}_{m+1}=\boldsymbol{a}_m+\Delta\boldsymbol{a}_m\tag{7}$$

在实际应用中，有效的方法是将Newton-Raphson法或修正的Newton-Raphson法与增量法结合起来，即在每个增量步上应用Newton-Raphson法或修正的Newton-Raphson法。

2 类设计及其Python实现

有限元分析流程主要包含：前处理、分析计算、后处理3个部分。根据弹塑性有限元的求解过程，并结合面向对象程序设计结构，设计了如图1所示的面向对象有限元框架。这样的设计使得程序结构清晰，更符合从事有限元编程人员的思维，也便于后续的扩充与维护。

2.1 前处理

有限元前处理是程序运行的基础，以几何区域离散为单元，为后续的分析计算提供数据。本文有限元框架的前处理，采用Gmsh[15]的接口API进行参数化建模及网格划分，可做到即时的几何、网格显示。Gmsh是一个开源的有限元网格生成器，内嵌了CAD引擎和后处理器，可处理2D、3D几何网格，提供了多种网格以及网格剖分算法。本文框架可通过Mesh类的派生类GmshReader获取采用Gmsh的Python API接口划分网格时的网格数据，如节点坐标、单元节点数

Published by Journal of System Simulation, 2024

Journal of System Simulation, Vol. 36 [2024], Iss. 5, Art. 6

组、载荷数据、边界条件数据等，所有数据构成了有限元程序的数据基础。也提供了对网格文件的数据提取的派生类MeshReader，通过Python中的meshio库可以方便地进行数据提取并转化为本文程序需要的数组，meshio可以读取和写入多种网格文件并在它们之间平滑转换，如ABAQUS (.inp)，ANSYS msh(.msh)，VTK(.vtk)等。

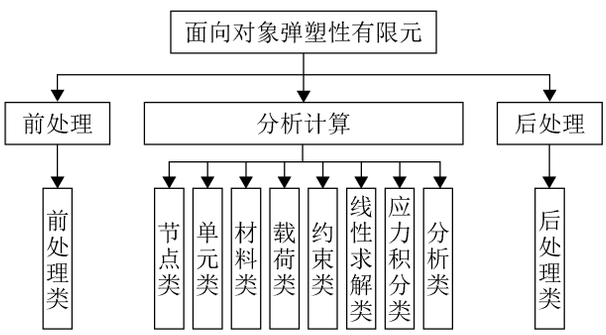


图1 有限元类的设计
Fig. 1 Design of finite element classes

2.2 分析计算

面向过程式的有限元分析是个高度耦合的过程，节点、单元、载荷等相互影响、相互耦合，使得整个程序难以维护和扩充。通过采用面向对象的设计，可使得整个有限元程序的耦合过程清晰可见，不需再关注所改变内容对其他部分的影响，类的修改和扩充仅仅是在某一个类中实现的，易于维护和扩充。本文按照有限元分析计算的主要功能组，抽象出了如下几类：

(1) 节点类

在有限元分析中，通过将区域离散成单元，所有的计算最终都是等效在节点上进行的，所以节点类是所有类中最普遍的类。在大多数面向对象有限元程序中，节点类不仅封装了坐标、自由度数据，还附加了节点位移、节点力等。由于节点类是有限元分析中实例化最多的类，本文简化了节点类的设计，节点类的作用仅为其他类提供节点坐标以及节点自由度，施加的节点力和节点位移将在载荷类和约束类中进行，产生的节点力和节点位移将在分析类中计算出结果后在后处理类中获得节点数据和相关云图。

(2) 单元类

单元类是整个有限元的核心，有限元通过将几何区域离散成单元，通过将单元刚度矩阵组装成整体刚度矩阵后完成计算。对于有限元的刚度矩阵，往往采用高斯积分，所有的计算均是在单元内的高斯点上进行的，如形函数及其导数、雅可比矩阵及其行列式等，应力应变及内力向量也在此水平上进行计算，所以单元类承载着最核心以及最繁重的计算。单元类按维度可分为一维单元、二维单元、三维单元，所有维度的单元都可以通过继承基础单元类实现。

(3) 材料类

材料类主要封装了材料编号、弹性模量、泊松比等材料属性。材料类是为单元类服务的，通过为不同的单元类中添加材料类实例为单元赋予材料类的属性和方法，如材料厚度、单元类的屈服应力、弹塑性矩阵等都是从材料类中调用得到的。材料类通过继承可派生出各向同性弹性材料类、塑性材料类以及各向异性材料类等。

(4) 载荷类

载荷类通过继承可以派生出节点载荷类、面载荷类、体载荷类等。载荷类通过节点类实例获得节点坐标、节点自由度，对于不同类型的载荷通过相应的派生类可得到整体的等效节点荷载向量。

(5) 约束类

约束类通过节点类实例获得节点坐标、节点自由度，通过继承，可采用置0化1法、乘大数法等多种边界条件处理方法，实现对总体刚度矩阵和等效节点载荷向量的修正。

(6) 线性求解类

线性方程组的求解方法有很多种，大致可分为直接法和迭代法。直接法是指在不产生舍入误差的前提下经过有限次运算得到的精确解。迭代法是从解的某个近似值出发，通过迭代格式构造一个无穷序列，使其收敛于方程组精确解。Python语言的第三方库NumPy[16]和SciPy[17]都提供

https://dc-china-simulation.researchcommons.org/journal/vol36/iss5/6
DOI: 10.16182/j.issn1004731x.joss.23-0084

Li and Xiao: Implementation and Numerical Simulation on Object-oriented Elasti

了线性方程组的直接求解方法，同时，Scipy还提供了稀疏矩阵的求解。对于有限元程序，矩阵规模通常是较大的，可派生出如预处理共轭梯度法等高效算法类求解稀疏矩阵方程组。

(7) 应力积分类

在进行弹塑性有限元计算中，应力的计算至关重要，由于应力应变不再是简单的线性关系，应力计算的精确性将直接影响到结果，选择合适的积分算法尤为重要。应力积分方法可分为显示算法和隐式算法，应力积分类可以派生出各种应力更新算法类，如显式积分类、完全隐式积分类、半隐式积分类等。隐式算法可参考文献[18]，下面以显式算法为例，简单介绍其主要步骤。

step 1：计算应变增量 $\Delta\boldsymbol{\varepsilon}_{n+1}$。假定为弹性状态，计算应力增量 $\Delta\boldsymbol{\sigma}^{\mathrm{e}}=\boldsymbol{D}\Delta\boldsymbol{\varepsilon}_{n+1}$，并累加前一步的应力状态计算试应力：$\boldsymbol{\sigma}^{\mathrm{trail}}=\boldsymbol{\sigma}_n+\Delta\boldsymbol{\sigma}^{e}$，计算试应力的等效应力 $\bar{\boldsymbol{\sigma}}^{\mathrm{trail}}$。

step 2：利用试应力的等效应力 $\bar{\boldsymbol{\sigma}}^{\mathrm{trail}}$ 与屈服应力判断第 n+1 步的屈服状态。对于高斯点第 n、n+1 步均处于弹性状态，所求的试应力即为真实应力；对于高斯点第 n 步处于弹性状态，第 n+1 步处于塑性状态，则高斯点在给定的应变增量下将进入弹塑性状态，存在一个比例因子 r，使 $f\left(\boldsymbol{\sigma}_n+(1-r)\Delta\boldsymbol{\sigma}^{\mathrm{e}},\boldsymbol{\kappa}_n\right)=0$，$r$的求解可参考文献[19-20]；对于第 n、n+1 步均处于塑性状态，此时为塑性加载，全部应力增量用来计算塑性应变增量，令 r=1。因此，应变增量被分为导致纯弹性变形的 $(1-r)\Delta\boldsymbol{\varepsilon}_{n+1}$。导致弹塑性变形的 $r\Delta\boldsymbol{\varepsilon}_{n+1}$。

step 3：对于屈服高斯点用公式 $\boldsymbol{\sigma}^{\mathrm{e}}_{n+1}=\boldsymbol{\sigma}_n+(1-r)\Delta\boldsymbol{\sigma}^{\mathrm{e}}$ 计算第 n+1 步应力的弹性部分 $\boldsymbol{\sigma}^{\mathrm{e}}_{n+1}$。

step 4：为了获得更好的积分精度，通常将构成弹塑性响应的应变增量分为足够多的小量，避免解的飘逸，即 $\Delta\boldsymbol{\varepsilon}_i=r\Delta\boldsymbol{\varepsilon}_{n+1}/m$，$\Delta\boldsymbol{\sigma}^{\mathrm{e}}_i=r\Delta\boldsymbol{\sigma}^{\mathrm{e}}/m$。

step 5：计算 $\Delta\boldsymbol{\sigma}_i=\boldsymbol{\sigma}^{\mathrm{e}}_i-\boldsymbol{D}_p\Delta\boldsymbol{\varepsilon}_i$。

step 6：更新应力 $\boldsymbol{\sigma}_{i+1}=\boldsymbol{\sigma}^{\mathrm{e}}_{n+1}+\Delta\boldsymbol{\sigma}_i$。

step 7：计算塑性应变增量和等效塑性应变增量。重复计算 step 5~7 共 m 次。

(8) 分析类

分析类是整个有限元计算的统筹部分，所有的类在分析类中通过互相调用以达到最终的计算目的。分析类的主要功能是组装单元刚度矩阵以及单元内力向量、计算有限元系统位移、内力。对于线弹性单元，只需要总体刚度矩阵和总体载荷向量即可计算系统的总体位移，并由此计算出应力应变等。对于弹塑性分析，应力应变不再是线性关系，需要进行多次的增量迭代求解，由此可派生出各种算法分析类。对于常见的非线性方程组的求解方法有直接迭代法、牛顿迭代法、拟牛顿迭代法、增量法、弧长法等。

目前常用的是牛顿迭代法和增量法的结合方法，通过将载荷划分成有限的增量步，在每一个增量步中应用牛顿迭代法。然而，当本构模型存在应力下降的趋势，牛顿迭代法由于无法越过极值点，则会导致计算无法收敛，很容易发生snap-through和snap-back现象，此时则必须采用弧长法等有效算法进行计算。通过对分析类的继承，可以实现多种算法分析类如NewtonRaphsonAnalysis类、ArcLengthAnalysis类等，不同的分析类可针对不同类型问题，体现了面向对象有限元程序的可扩充性和可维护性。

2.3 后处理

有限元的计算大多以位移为未知数，通过分析计算得到的是单元节点位移，然而实际工程问题中感兴趣的通常是应力分布，特别是最大应力的位置和数值[21]。由于应变矩阵是插值函数对坐标求导后的结果，这种导数运算使得应力应变的精度较位移降低，因此，必须通过最佳应力点外推的方式得到更精确的应力应变。

分析类计算出的单元节点位移和最佳应力点的应力应变，通过后处理类利用外推获得节点较精确的应力应变解。后处理类也提供了数据可视化功能，Python语言拥有强大的数据可视化能力，通过matplotlib[22]库可简便、快捷地绘制各类曲线、图表，如位移载荷曲线、应力应变曲线等；通过

Published by Journal of System Simulation, 2024

Journal of System Simulation, Vol. 36 [2024], Iss. 5, Art. 6

将位移、应力应变等数据写入 VTK 文件后利用 PyVista[23]、Mayavi[24]等 VTK 相关库可进行云图绘制，也可以用 Paraview[25]等第三方软件进行显示。

将面向对象方法与有限元程序设计相融合，并采用 Python 语言编写了面向对象弹塑性有限元框架，大大简化了数据结构，提高了设计效率，同时也提升了程序的可维护性与运行灵活性。通过 NumPy 库可进行各种矩阵运算操作，支持大量的维度数组与矩阵运算，针对数组运算提供大量的数学函数库，相比 C++语言，避免了矩阵类的设计。另外，Python 语法的简洁性也使面向对象有限元程序的开发难度大幅降低，可视化也使有限元程序更具有实用性。

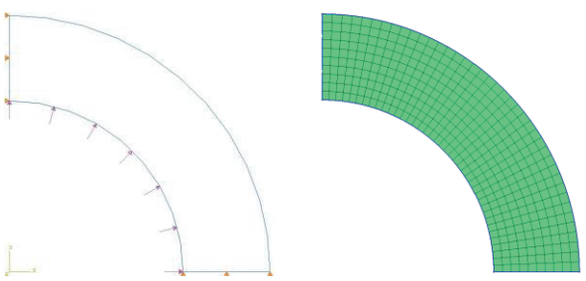
3 算例及结果分析

首先，将基于 Python 所编写的面向对象弹塑性有限元程序框架应用于受内压的厚壁圆筒弹塑性有限元分析中，并通过与理论解及 ABAQUS 解进行对比，以验证程序的正确性和计算结果的可靠性。然后，将本文框架应用于土石混合体、混凝土骨料的弹塑性细观力学分析中，并与 ABAQUS 计算结果进行对比，以进一步验证面向对象弹塑性有限元的有效性和对问题的适应性。

算例 1　受内压的厚壁圆筒

考虑受内压的厚壁圆筒，设其内径 $a=10$ mm，外径 $b=15$ mm，圆筒内部($r=a$)承受压力 $p=150$ MPa，外表面($r=b$)为自由表面，材料的弹性模量 $E=200$ GPa，泊松比 $v=0.30$，屈服强度 $\sigma_Y=380$ MPa。

考虑对称性，选取模型的 1/4 作为有限元计算模型，如图 2(a)所示。利用 Gmsh 的 Python API 接口进行四边形网格剖分，图 2(b)给出了单元数为 400 的网格剖分图。以此网格剖分为例，验证本文设计的面向对象弹塑性有限元程序框架对求解弹塑性力学问题的正确性。采用 8 节点平面应变四边形单元，选用理想弹塑性模型进行计算，并服从 V. Mises 屈服准则。图 3~4 分别给出了 $r=a\sim b$ 内($y=0$)11 个节点处的弹性阶段($p=120$ MPa)和弹塑性阶段($p=150$ MPa)的径向应力、环向应力与理论解[26]及 ABAQUS 解的比较结果。由此可见，本文程序的计算结果与理论解、ABAQUS 解吻合度高，误差较小。



(a) 有限元计算模型　(b) Gmsh网格剖分

图 2　模型与有限元网格

Fig. 2　Model and FEM mesh

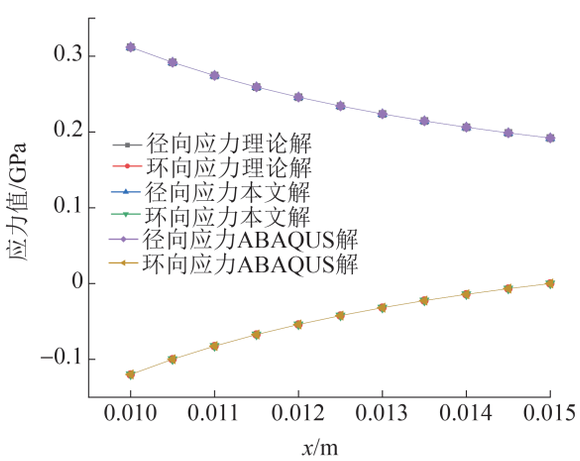


图 3　弹性阶段应力分布

Fig. 3　Stress distribution in elastic stage

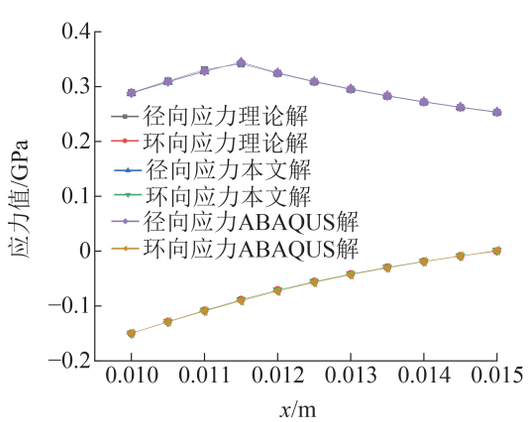


图 4　弹塑性阶段应力分布

Fig. 4　Stress distribution in elastic-plastic stage

https://dc-china-simulation.researchcommons.org/journal/vol36/iss5/6
DOI: 10.16182/j.issn1004731x.joss.23-0084

为进一步验证本文程序的实用性和有效性，取不同单元数进行对比计算，选取3个特征点，其坐标分别为(0.01, 0)、(0.012 5, 0)和(0.015, 0)，将本文程序计算得到的3个特征点处的径向应力分别与理论解、ABAQUS解进行对比，并计算出相应的相对误差，如表1所示。

(1) 与理论解对比：由于本文计算时采用的8节点四边形单元，在单元数为100时精度就较高，当单元数由100升至400时，计算精度有所提高，当单元数为1 600时，结果与单元数为400时接近，这是由于在单元数为400时计算结果已经很接近理论解，继续提高单元数量也无法提高精度，数值解将在理论解附近震荡。

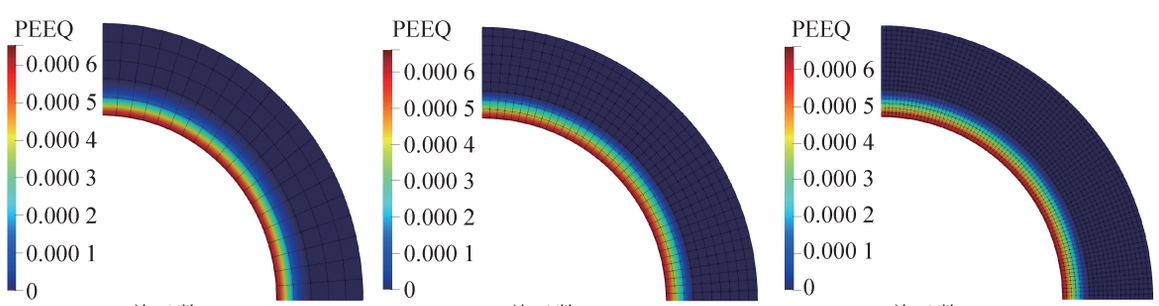
(2) 与ABAQUS解对比：本文解与ABAQUS解的误差基本保持在0.1%以下，随着网格数的增加，特征点应力增加、减少的趋势也始终保持一致。

如图5所示，不同网格数下的等效塑性应变云图，可以看出，厚壁圆筒内壁屈服并向外扩张，弹塑性交界线呈现弧形，符合理论实际。

针对Python运行效率问题，本文分别计算了纯Python语言程序以及利用Cython[27]对Python中一小部分代码加速后的运行平均时间，如表2所示。由表2可知，利用Cython对Python加速效果明显，并随着自由度的增加，加速效果越明显。Cython不需要对Python原有代码作大量修改，仍然保留着Python的语法特性，利用Cython对Python程序的加速可以有效解决Python的效率问题。

表1　3个特征点处的径向应力及其误差比较
Table 1　Radial stress and comparisons of error at three special points

单元数	坐 标	理论解	本文解	ABAQUS解	本文解与理论解的相对误差/%	ABAQUS解与本文解的相对误差/%
100	(0.01, 0)	0.288 900	0.288 539	0.289 178	0.124 96	0.221 46
	(0.012 5, 0)	0.308 651	0.310 844	0.310 733	0.710 50	0.035 71
	(0.015, 0)	0.252 992	0.254 025	0.253 935	0.408 30	0.035 43
400	(0.01, 0)	0.288 900	0.287 863	0.288 462	0.358 95	0.208 01
	(0.012 5, 0)	0.308 651	0.309 647	0.309 455	0.322 69	0.062 01
	(0.015, 0)	0.252 992	0.253 803	0.253 646	0.320 56	0.061 86
1 600	(0.01, 0)	0.288 900	0.287 691	0.288 282	0.418 50	0.205 43
	(0.012 5, 0)	0.308 651	0.309 684	0.309 458	0.334 68	0.072 98
	(0.015, 0)	0.252 992	0.253 839	0.253 654	0.334 80	0.072 88



(a) 单元数100　(b) 单元数400　(c) 单元数1 600

图5　不同单元数计算得到的等效塑性应变云图
Fig. 5　Nephograms of equivalent plastic strain obtained by different number of elements

表2　不同自由度下弹性阶段运行时间
Table 2　CPU times of elastic stage under different degrees of freedoms　s

自由度数	纯Python	Cython加速
2 602	1.04	0.20
10 002	7.33	0.81
39 202	54.76	3.81
87 602	181.03	9.10
155 202	436.15	18.54

为了体现面向对象弹塑性有限元框架的扩充性，本文基于所提框架快速实现了基于单元的光滑有限元(CS-FEM)[28]及基于单元的多边形光滑有限元(*n*CS-FEM)[29]，并给出了不同方法在(0.01, 0)处径向应力的比较，验证了本文扩充实现的光滑有限元法的正确性与有效性，如表3所示。

Journal of System Simulation, Vol. 36 [2024], Iss. 5, Art. 6

表3 特征点(0.01，0)处径向应力的比较
Table 3 Comparison of radial stress at the point (0.01, 0)

单元数	CS-FEM	*n*CS-FEM	FEM(Q4)
100	0.322 453	0.295 140	0.336 624
400	0.305 595	0.293 769	0.313 096
1 600	0.296 747	0.292 322	0.300 599
理论解		0.288 900	

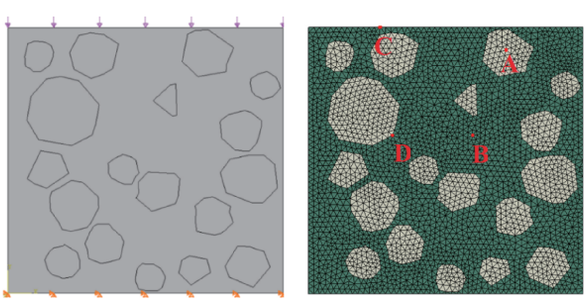
算例2 土石混合体的弹塑性分析

选取 100 cm × 100 cm 正方形试件，参考文献[30]，土石混合体中各项材料的基本参数见表4。

表4 土石混合体细观结构材料参数
Table 4 Material parameters of soil-rock mixture

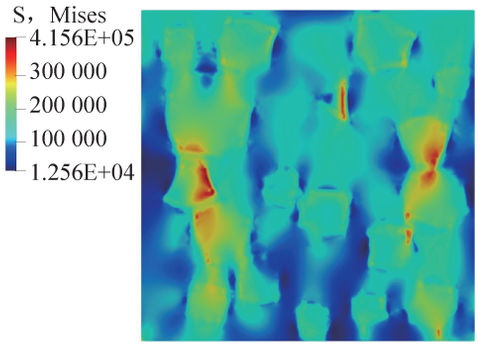
材料	弹性模量/MPa	泊松比	密度/(kg·m^{-3})	内摩擦角/(°)	黏聚力/KPa
土体	50	0.3	1 800	30	50
块石	20 000	0.2	2 500	42	500

考虑多边形块石含量为30%的土石混合体细观结构模型，材料采用关联流动法则的理想弹塑性Mohr-Coulomb模型，对试件进行三角形网格剖分，采用二次元进行计算，块石与土基体间不考虑接触的影响。对多边形土石混合体模型进行单向加载数值模拟实验，竖向施加大小为0.15 MPa的均布荷载，侧向自由边界，底端固定，如图6所示。图7和图8分别给出了本文计算得到的Mises等效应力(Mises)、等效塑性应变(PEEQ)各云图与ABAQUS计算结果的对比情况。由此可知，本文计算的结果均与ABAQUS计算结果高度吻合。在等效塑性应变云图中，塑性应变区也都出现在相同位置。

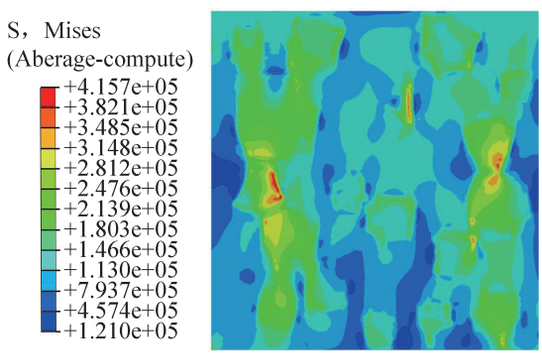


(a) 细观结构模型 (b) 有限元网格剖分

图6 土石混合体模型与有限元网格
Fig. 6 Model of soil-rock mixture and FEM mesh

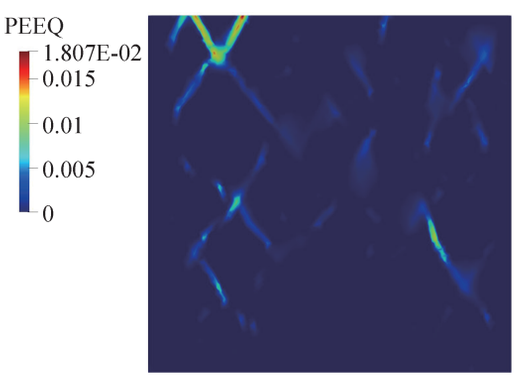


(a) 本文程序

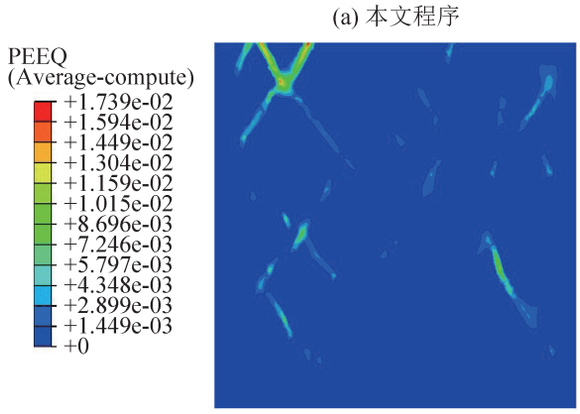


(b) ABAQUS

图7 算例2应力结果比较
Fig. 7 Comparison of stress results for example 2



(a) 本文程序



(b) ABAQUS

图8 应变结果比较
Fig. 8 Comparison of strain results

https://dc-china-simulation.researchcommons.org/journal/vol36/iss5/6
DOI: 10.16182/j.issn1004731x.joss.23-0084

为了定量分析，在土石混合体模型中，选取了4个特征点，分别位于块石内部、基体上、边界上、块石与基体的交接处，如图6(b)所示。对5个特征点，分别给出了其相应的Mises等效应力，y方向应力，x和y方向位移及等效塑性应变，如表5所示。由此表可知本文程序计算得到的结果与ABAQUS较为吻合。

表5　本文计算结果与ABAQUS结果比较

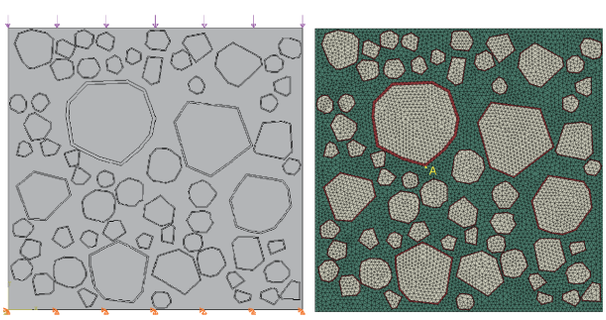
Table 5　Comparison between the results obtained by our program and by ABAQUS

特征点	Mises		S22		U1		U2		PEEQ	
	①	②	①	②	①	②	①	②	①	②
A	136 536	136 580	−160 265	−160 275	0.014 48	0.014 19	−0.145 5	−0.146 0	0	0
B	170 984	169 983	−171 182	−169 860	−0.037 91	−0.037 86	−0.100 08	−0.100 69	0.003 21	0.003 33
C	133 589	134 058	−136 942	−136 407	−0.021 69	−0.021 75	−0.136 76	−0.138 22	0.017 39	0.018 07
D	133 480	133 487	−139 969	−140 358	0.003 89	0.004 23	−0.091 55	−0.091 27	0.001 61	0.001 56

注：①为本文程序计算结果；②为ABAQUS计算结果。

算例3　骨料模型的弹塑性分析

考虑如图9(a)所示含量为50.30%的二级配骨料模型，该试件尺寸为150 mm×150 mm，取单位厚度。设基体材料的弹性模量$E_1=13.4$ GPa，泊松比$v_1=0.25$，屈服应力$\sigma_{s1}=25$ MPa，切线模量$E_t=1.4$ GPa；界面层的弹性模量$E_2=11$ GPa，泊松比$v_2=0.20$，屈服应力$\sigma_{s2}=21$ MPa，切线模量$E_t=1.031$ GPa；多边形骨料的弹性模量$E_3=74.5$ GPa，泊松比$v_3=0.15$。试件底端固定，在顶端施加均布压力$q=28$ N/mm^2。本算例应力单位均为MPa。

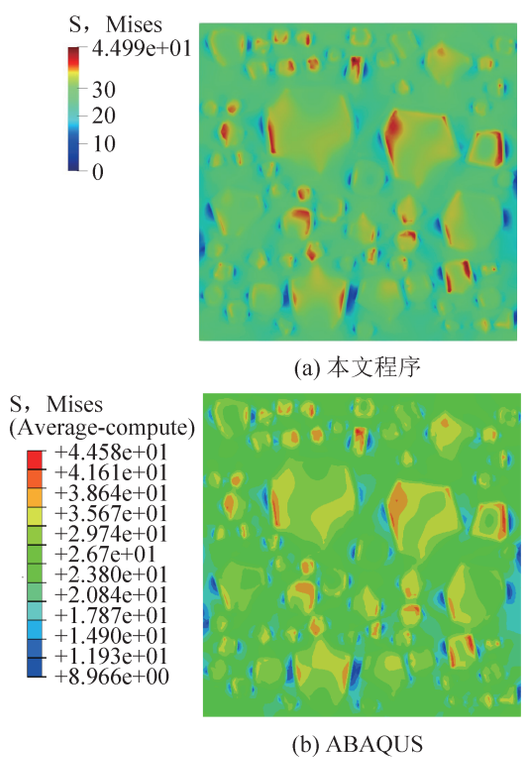


(a) 多边形骨料模型　　(b) 有限元网格剖分

图9　二级配骨料模型与有限元网格

Fig. 9　Two-graded aggregate model and FEM mesh

采用三角形网格剖分，如图9(b)所示。计算中采用平面应变三角形线性元和线性强化模型，并服从V.Mises屈服准则。为了对比，在相同网格剖分和参数下利用ABAQUS进行了计算。图10给出了利用本文程序及ABAQUS计算得到的Mises应力云图，从云图可以看出，两者基本吻合，最大值和最小值的位置也完全一致。对界面层上的A点(图9(b))，给出了利用本文程序及ABAQUS计算得到的等效应力−应变曲线，如图11所示。由此可见，两者结果趋势一致，符合线性强化模型，弹性区结果完全一致，但随着塑性的增大，两者结果存在一点小偏差，原因是本文程序使用的是显式应力更新算法，而ABAQUS采用的是默认的隐式算法。



(a) 本文程序

(b) ABAQUS

图10　算例3应力结果比较

Fig. 10　Comparison of stress results for example 3

Journal of System Simulation, Vol. 36 [2024], Iss. 5, Art. 6

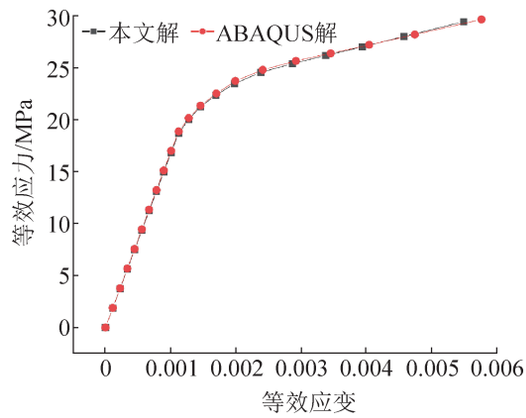


图 11 界面层上 A 点的等效应力-应变曲线
Fig. 11 Equivalent stress-strain curve of point A on the interface layer

上述算例的数值结果表明，本文设计的基于 Python 的面向对象弹塑性有限元框架对求解实际弹塑性力学问题具有很好的有效性和适应性。采用 Python 可以方便快捷地进行面向对象有限元程序的开发，其强大的第三方库与可视化能力也为自编有限元程序赋予了更多可能。

4 结论

有限元商业软件尽管功能强大，但在实际应用中需要处理的问题往往是多样的，很多情况下，将导致这些商业软件并不完全通用。当现有的功能不再满足需要时就需要自己添加新的功能，尽管商业软件都提供了用户子程序或内嵌语言等开放接口，但其灵活性始终是受限制的。面向对象非线性有限元框架的开发，可有效克服用户不能直接修改商业软件程序源代码的缺陷，更能适应实际问题的多样性和多变性。本文基于 Python 自主开发了一个面向对象的弹塑性有限元分析框架，其实用性和有效性也得到了算例验证。

采用基于 Python 的面向对象程序设计方法，使得非线性有限元程序整体开发周期降低、开发难度也大幅下降，并在此框架下可根据需要开发、扩充额外的功能。相比 C++ 语言，Python 中的 NumPy 库可避免矩阵类的建立，Python 强大的可视化能力也为有限元数据处理提供了重要支持。面向对象弹塑性有限元程序更加结构化，编写更简便，代码的可重用成分更大，也更易于维护和扩充，更能适应实际问题的多样性和多变性。本文仅做了弹塑性有限元方法相关的开发和有限的数值模拟试验，但所得到的数值结果还是令人鼓舞的。今后，仍需对更多实际问题进行更广泛的数值测试，以验证 Python 编程的有效性。另外，还需在几何大变形、接触非线性等方面开展更多工作，以研制完整的非线性有限元分析模块，并利用 Cython 进行持续优化以获得更好的计算效率。

参考文献：

[1] Archer G C, Fenves G, Thewalt C. A New Object-oriented Finite Element Analysis Program Architecture [J]. Computers & Structures, 1999, 70(1): 63-75.

[2] Bruce W R Forde, Ricardo O Foschi, Siegfried F Stiemer. Object-oriented Finite Element Analysis[J]. Computers & Structures, 1990, 34(3): 355-374.

[3] Fenves G L. Object-oriented Programming for Engineering Software Development[J]. Engineering with Computers, 1990, 6(1): 1-15.

[4] Miller G R. An Object-oriented Approach to Structural Analysis and Design[J]. Computers & Structures, 1991, 40(1): 75-82.

[5] Yves Dubois-Pèlerin, Thomas Zimmermann, Patricia Bomme. Object-oriented Finite Element Programming: II. A Prototype Program in Smalltalk[J]. Computer Methods in Applied Mechanics and Engineering, 1992, 98(3): 361-397.

[6] Ph Menétrey, Th Zimmermann. Object-oriented Non-linear Finite Element Analysis: Application to J2 Plasticity[J]. Computers & Structures, 1993, 49(5): 767-777.

[7] Lages E N, Paulino G H, Menezes I F M, et al. Nonlinear Finite Element Analysis Using an Object-oriented Philosophy-application to Beam Elements and to the Cosserat Continuum[J]. Engineering with Computers, 1999, 15(1): 73-89.

[8] 李会平, 曹中清, 周本宽. 弹塑性分析的面向对象有限元方法[J].西南交通大学学报, 1997, 32(4): 401-406.
Li Huiping, Cao Zhongqing, Zhou Benkuan. Object Oriented Finite Element Method for Elastoplastic Analysis[J]. Journal of Southwest Jiaotong University, 1997, 32(4): 401-406.

[9] 陈飙松, 陆旭泽, 张盛. 基于SiPESC平台的弹塑性分析

https://dc-china-simulation.researchcommons.org/journal/vol36/iss5/6
DOI: 10.16182/j.issn1004731x.joss.23-0084

Li and Xiao: Implementation and Numerical Simulation on Object-oriented Elasti

的软件框架[J]. 计算力学学报, 2016, 33(4): 599-604.
Chen Biaosong, Lu Xuze, Zhang Sheng. Software Framework for Elasto-plastic Analysis Based on SiPESC Platform[J]. Chinese Journal of Computational Mechanics, 2016, 33(4): 599-604.

[10] Martin Alnæs, Jan Blechta, Johan Hake, et al. The FEniCS Project Version 1.5[J]. Archive of Numerical Software, 2015, 3(100): 9-23.

[11] Rathgeber F, Ham D A, Mitchell L, et al. Firedrake: Automating the Finite Element Method by Composing Abstractions[J]. ACM Transactions on Mathematical Software, 2017, 43(3): 24.

[12] Robert Cimrman, Vladimír Lukeš, Eduard Rohan. Multiscale Finite Element Calculations in Python Using SfePy[J]. Advances in Computational Mathematics, 2019, 45(4): 1897-1921.

[13] Murat Yilmaz. Rapid Translation of Finite-element Theory into Computer Implementation Based on a Descriptive Object-oriented Programming Approach[J]. Turkish Journal of Electrical Engineering and Computer Sciences, 2018, 26(6): 3367-3382.

[14] Miroslav Marjanović, Marija Nefovska-Danilović, Emilija Damnjanović. Framework for Dynamic-stiffness-based Free Vibration Analysis of Plate-like Structures[J]. Shock and Vibration, 2019, 2019: 1369235.

[15] Christophe Geuzaine, Jean-François Remacle. Gmsh: A 3-D Finite Element Mesh Generator with Built-in pre- and Post-processing Facilities[J]. International Journal for Numerical Methods in Engineering, 2009, 79(11): 1309-1331.

[16] Harris C R, Millman K J, Stéfan J van der Walt, et al. Array Programming with NumPy[J]. Nature, 2020, 585 (7825): 357-362.

[17] Pauli Virtanen, Gommers R, Oliphant T E, et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python[J]. Nature Methods, 2020, 17(3): 261-272.

[18] Belytschko T, Liu W K, Moran B. 连续体和结构的非线性有限元[M]. 庄茁, 译. 北京: 清华大学出版社, 2002.
Belytschko T, Liu W K, Moran B. Nonlinear Finite Elements for Continua and Structures[M]. Translated by Zhuang Zhuo. Beijing: Tsinghua University Press, 2002.

[19] 朱伯芳. 有限元单元法原理与应用[M]. 3版. 北京: 中国水利水电出版社, 2009.

[20] 陈惠发, A F 萨里普. 弹性与塑性力学[M]. 余天庆, 王勋文, 刘再华, 译. 北京: 中国建筑工业出版社, 2004.
Chen Huifa, A F 萨里普. Elasticity and Plasticity[M]. Translated by Yu Tianqing, Wang Xunwen, Liu Zaihua. Beijing: China Architecture & Building Press, 2004.

[21] 王勖成. 有限单元法[M]. 北京: 清华大学出版社, 2003.
Wang Xucheng. Finite Element Method[M]. Beijing: Tsinghua University Press, 2003.

[22] Hunter J D. Matplotlib: A 2D Graphics Environment[J]. Computing in Science & Engineering, 2007, 9(3): 90-95.

[23] Sullivan C B, Kaszynski A A. PyVista: 3D Plotting and Mesh Analysis Through a Streamlined Interface for the Visualization Toolkit (VTK) [J]. The Journal of Open Source Software, 2019, 4(37): 1450.

[24] Prabhu Ramachandran, Gael Varoquaux. Mayavi: 3D Visualization of Scientific Data[J]. Computing in Science & Engineering, 2011, 13(2): 40-51.

[25] Ayachit U. The ParaView Guide: A Parallel Visualization Application[M]. Clifton Park, NY, USA: Kitware, Inc., 2015.

[26] 徐秉业. 塑性力学[M]. 北京: 高等教育出版社, 1988.

[27] Stefan Behnel, Bradshaw R, Craig Citro, et al. Cython: The Best of Both Worlds[J]. Computing in Science & Engineering, 2011, 13(2): 31-39.

[28] Liu G R, Nguyen-Thoi T, Nguyen-Xuan H, et al. A Node-based Smoothed Finite Element Method (NS-FEM) for Upper Bound Solutions to Solid Mechanics Problems[J]. Computers & Structures, 2009, 87(1/2): 14-26.

[29] Dai Ky, Liu G R, Nguyen T T. An N-sided Polygonal Smoothed Finite Element Method (nSFEM) for Solid Mechanics[J]. Finite Elements in Analysis and Design, 2007, 43(11/12): 847-860.

[30] 徐文杰, 胡瑞林, 岳中崎. 土–石混合体随机细观结构生成系统的研发及其细观结构力学数值试验研究[J].岩石力学与工程学报, 2009, 28(8): 1652-1665.
Xu Wenjie, Hu Ruilin, Yue Zhongqi. Development of Random Mesostructure Generating System of Soil-Rock Mixture and Study of Its Mesostructural Mechanics Based on Numerical Test[J]. Chinese Journal of Rock Mechanics and Engineering, 2009, 28(8): 1652-1665.

Published by Journal of System Simulation, 2024