УДК 004.94
DOI https://doi.org/10.26661/2413-6549-2020-1-18

ОБЪЕКТНО-ОРИЕНТИРОВАННАЯ РЕАЛИЗАЦИЯ БИБЛИОТЕКИ КОНЕЧНО-ЭЛЕМЕНТНОГО АНАЛИЗА НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON

Игнатченко М. С.
аспирант кафедры программной инженерии
Запорожский национальный университет
ул. Жуковского, 66, Запорожье, Украина
orcid.org/0000-0001-5753-2620
gomenjukmarija@gmail.com

Кудин А. В.
кандидат физико-математических наук,
доцент кафедры программной инженерии
Запорожский национальный университет
ул. Жуковского, 66, Запорожье, Украина
orcid.org/0000-0002-5917-9127
oleksiy.kudin@gmail.com

Гнездовский А. В.
старший преподаватель кафедры информационных технологий в туризме
Национальный университет «Запорожская политехника»
ул. Жуковского, 64, Запорожье, Украина
orcid.org/0000-0003-0392-3030
gnezdovskiy.alexey@gmail.com

Ключевые слова: *метод конечных элементов, объектно-ориентированное программирование, Python, библиотека классов, наследование, инкапсуляция.*

Необходимость оценки прочности и долговечности вновь создаваемых или уже эксплуатируемых инженерно-технических систем приводит к возникновению сложных задач механики деформируемого твердого тела. Их решение в большинстве случаев невозможно без использования компьютерного моделирования и численных методов. Наиболее популярным среди них является метод конечных элементов. Его практическое применение невозможно без использования вычислительной техники. Разработка соответствующего программного обеспечения является по-прежнему актуальной задачей. Большинство существующих популярных программ, автоматизирующих различные аспекты применения метода конечных элементов, являются закрытыми коммерческими системами. Соответствующее программное обеспечение с открытым исходным кодом в большинстве случаев написано на языке программирования C++, что в силу его сложности затрудняет адаптацию этих программ для решения новых классов задач. В последнее время большую популярность приобрел язык программирования Python. Его отличительными особенностями являются выразительность и простота, а также большое количество библиотек, что делает Python перспективным инструментом для разработки программ, выполняющих инженерно-технические расчеты.
В статье описана объектно-ориентированная реализация библиотеки классов, предназначенных для кончено-элементного анализа стационарных и динамических задач теории упругости на языке программирования Python.

Разработана иерархическая структура классов, инкапсулирующих объект расчета, статическую и динамическую реализации метода конечных элементов, конечные элементы разных типов, дискретную модель исходного объекта и т. д. Благодаря простоте реализации, удалось построить набор эффективных и интуитивно-понятных классов, позволяющих, с одной стороны, выполнять расчет различных типов задач механики, а с другой – дать возможность легко его расширять для повышения функциональности библиотеки при решении специализированных задач. Приведены примеры использования описанных классов для решения задач теории упругости.

ОБ'ЄКТНО-ОРІЄНТОВАНА РЕАЛІЗАЦІЯ БІБЛІОТЕКИ СКІНЧЕННО-ЕЛЕМЕНТНОГО АНАЛІЗУ МОВОЮ ПРОГРАМУВАННЯ PYTHON

Ігнатченко М. С.
аспірант кафедри програмної інженерії
Запорізький національний університет
вул. Жуковського, 66, Запоріжжя, Україна
orcid.org/0000-0001-5753-2620
gomenjukmarija@gmail.com

Кудін О. В.
кандидат фізико-математичних наук,
доцент кафедри програмної інженерії
Запорізький національний університет
вул. Жуковського, 66, Запоріжжя, Україна
orcid.org/0000-0002-5917-9127
oleksiy.kudin@gmail.com

Гнездовський О. В.
старший викладач кафедри інформаційних технологій у туризмі
Національний університет «Запорізька політехніка»
вул. Жуковського, 64, Запоріжжя, Україна
orcid.org/0000-0003-0392-3030
gnezdovskiy.alexey@gmail.com

Ключові слова: *метод скінченних елементів, об'єктно-орієнтоване програмування, Python, бібліотека класів, успадкування, інкапсуляція.*

Необхідність оцінювання міцності й довговічності розроблюваних або вже наявних інженерно-технічних систем призводить до виникнення складних задач механіки деформованого твердого тіла. Їх розв'язання в більшості випадків неможливе без використання комп'ютерного моделювання та чисельних методів, найбільш популярним серед яких є метод скінченних елементів. Його практичне застосування неможливе без використання обчислювальної техніки. Розроблення відповідного програмного забезпечення є, як і раніше, актуальною проблемою. Більшість наявних популярних програм, що автоматизують різні аспекти застосування методу скінченних елементів, є закритими комерційними системами. Відповідне програмне забезпечення з відкритим вихідним кодом у більшості випадків написано мовою програмування C++, що в силу її складності робить процес адаптації цих програм для вирішення нових класів задач досить непростим завданням. Останнім часом великої популярності набула мова програмування Python. Її відмінними рисами є виразність і простота, а також велика кількість наявних бібліотек, що робить Python перспективним інструментом для розроблення програм для інженерно-технічних розрахунків.

У статті описана об'єктно-орієнтована реалізація бібліотеки класів, призначених для скінчено-елементного аналізу стаціонарних і динамічних задач теорії пружності мовою програмування Python. Розроблена ієрархічна

структура класів, що інкапсулюють об'єкт розрахунку, статичну й динамічну реалізацію методу скінченних елементів, скінченні елементи різних типів, дискретну модель вихідного об'єкту тощо. Завдяки простоті реалізації вдалося побудувати набір ефективних та інтуїтивно-зрозумілих класів, що дають змогу, з одного боку, виконувати розрахунок різних типів задач механіки, а з іншого – легко його розширювати для підвищення функціональності бібліотеки. Наведено приклади використання описаних класів для розв'язання задач теорії пружності.

OBJECT-ORIENTED IMPLEMENTATION OF THE FINITE ELEMENT ANALYSIS LIBRARY IN THE PYTHON PROGRAMMING LANGUAGE

Ihnatchenko M. S.
Postgraduate Student at the Department of Software Engineering
Zaporizhzhia National University
Zhukovskoho str., 66, Zaporizhzhia, Ukraine
orcid.org/0000-0001-5753-2620
gomenjukmarija@gmail.com

Kudin O. V.
Candidate of Physical and Mathematical Sciences,
Associate Professor at the Department of Software Engineering
Zaporizhzhia National University
Zhukovskoho str., 66, Zaporizhzhia, Ukraine
orcid.org/0000-0002-5917-9127
oleksiy.kudin@gmail.com

Gnezdovskiy O. V.
Assistant Professor at the Department of Information Technology in Tourism
Zaporizhzhia Polytechnic National University
Zhukovskoho str., 64, Zaporizhzhia, Ukraine
orcid.org/0000-0003-0392-3030
gnezdovskiy.alexey@gmail.com

Key words: *finite element method, object-oriented programming, Python, class library, inheritance, encapsulation.*

The need to assess the strength and durability of newly created or already in use engineering and technical systems leads to the emergence of complex problems in the mechanics of a deformable solid. Their solution in most cases is impossible without the use of computer modeling and numerical methods. The most popular of these is the finite element method. The main idea of which is to replace a solid body with its discrete analogue. Its practical application is impossible without the use of computer technology. The development of appropriate software is still an urgent task. Most of the existing popular programs that automate various aspects of the application of the finite element method are closed commercial systems. The corresponding open source software is in most cases written in the C++ programming language, which, due to its complexity, makes it difficult to adapt these programs to solve new classes of problems. Recently, the Python programming language has become very popular. Its distinctive features are expressiveness and simplicity, as well as a large number of libraries, which makes Python a promising tool for developing programs that perform engineering calculations. The article describes an object-oriented implementation of a library of classes intended for finite element analysis of stationary and dynamic problems of elasticity theory in the Python programming language. A hierarchical structure of classes encapsulating the calculation object, static and dynamic implementations of the finite element method, finite elements of various types, a discrete model of the original object,

etc. was developed. calculation of various types of problems in mechanics, and on the other hand, make it easy to expand it to increase the functionality of the library when solving specialized problems. Examples of using the described classes for solving problems of the theory of elasticity are given. A possible direction for the development of the PyFEM library is both its extension by means of classes that implement the solution of new types of problems, and the addition of parallel computation support to PyFEM in order to increase the computational speed.

Постановка проблемы. В процессе разработки и производства современных инженерно-технических систем (высотных зданий и сооружений, аэрокосмической техники, транспортных машин, речных и морских судов и т. п.) постоянно возникают актуальные сложные проблемы, связанные с необходимостью оценки прочности и долговечности вновь создаваемых или уже эксплуатируемых конструкций. Решение соответствующих задач механики деформируемого твердого тела в большинстве случаев невозможно без использования компьютерного моделирования и численных методов, одним из наиболее эффективных среди которых является метод конечных элементов (далее – МКЭ) [1].

Применение МКЭ на практике без использования вычислительной техники не представляется возможным, поэтому к настоящему времени создано и постоянно продолжается разрабатываться большое количество разнообразного программного обеспечения (далее – ПО), автоматизирующего различные аспекты применения МКЭ: от генерации дискретных конечно-элементных моделей рассчитываемого объекта до визуального представления больших массивов полученных численных результатов. К наиболее известному ПО для конечно-элементного анализа различных типов задач относятся Abaqus [2], Ansys [3], COMSOL [4], MSC Nastran [5] и др. [6]. Однако большинство этих программ является закрытыми коммерческими системами. Альтернативой им является ПО с открытым программным кодом, среди которого можно выделить следующее: dial. II [7], FreeFEM [8], GetFEM [9], OpenCAD [10] и др. [11; 12]. В основном все эти программы написаны на языке программирования C++ [13], к сожалению, внесение в них изменений для адаптации данного ПО к решению новых классов задач является в силу сложности C++ весьма нетривиальной процедурой. Поэтому на практике возникает задача создания такого ПО для конечно-элементного анализа, которое, с одной стороны, имело бы открытую и доступную для понимания архитектуру и поддерживало бы возможность легкого внесения изменений для расширения его функциональности, а с другой – имело бы сопоставимую с написанными на C++ системами скорость работы.

В последнее время одним из наиболее популярных при разработке научного ПО является язык программирования Python [14], который изначально создавался для повышения эффективности работы программиста. Он имеет легкий для понимания синтаксис, и на данный момент для него разработано большое количество библиотек, реализующих различный математический функционал (например, NumPy [15] и SciPy [16]). Однако к настоящему времени систем конечно-элементного анализа на Python разработано сравнительно мало. Среди них можно выделить, например, fempy [17], FEniCS [18], PolyFEM [19], SfePy [20], впрочем часть из них только имеет python-интерфейс, но написана на C++. Это можно объяснить тем фактом, что принято считать скорость работы python-программ более низкой, чем у аналогов, написанных на C++. Тем не менее, динамическая типизация Python и его возможности по быстрой разработке приложений делают его перспективным средством создания нового эффективного ПО для численного решения широких классов задач механики с использованием МКЭ. Поэтому разработка открытой архитектуры ПО для МКЭ является актуальной в настоящее время задачей. С этой целью на языке Python разработана объектно-ориентированная библиотека классов PyFEM, реализующая решение статических и динамических задач теории упругости с помощью МКЭ. Она имеет простую открытую архитектуру, позволяющую легко расширять ее как специальными типами конечных элементов (КЭ), так и алгоритмами решения новых классов задач.

Изложение основного материала.

1. Конечно-элементная объектно-ориентированная модель классов PyFEM

ПО для конечно-элементного анализа в общем виде состоит из трех главных подсистем: препроцессора, процессора и постпроцессора. Препроцессор автоматизирует подготовку исходных для расчета данных (чаще всего под этим понимается генерация конечно-элементной модели исходного объекта расчета); процессор непосредственно выполняет расчет задачи, а постпроцессор – автоматизацию анализа полученных численных результатов.

Язык программирования Python поддерживает различные парадигмы программирования, в том числе и самую популярную на сегодняшний день объектно-ориентированную. Согласно ей, программа описывается в виде некоторой совокупности взаимодействующих друг с другом объектов, каждый из которых принадлежит к определенному типу данных (классу). Процессор библиотеки

PyFEM представляет собой набор классов, инкапсулирующих такие базовые сущности МКЭ, как дискретная модель рассчитываемого объекта (конечно-элементная сетка); КЭ заданного типа, реализующий построение локальной матрицы жесткости (массы и демпфирования); параметры расчета (упругие и физические свойства материала, краевые условия, нагрузки, температуру и т. п.); методы расчета (статика, динамика, упруго-пластичность и т. п.), включающие глобальные матрицы жесткости, массы и демпфирования и алгоритмы их построения; решатель системы линейных алгебраических уравнений (СЛАУ); хранилище результатов расчета и подсистему вывода информации о ходе расчета, возникших ошибках и т. п. (рис. 1).

Описание дискретной модели исходной области можно представить в виде иерархии классов, где базовый абстрактный класс TMesh содержит всю необходимую для расчета информацию о структуре сетки: тип КЭ; количество узлов; количество конечных и граничных элементов (ГЭ) и т. п., но не имеет реализации загрузки данных из файла конкретного формата (vol, mesh, trpa и т. д.). Производные от TMesh классы (например, TMeshVOL) содержат только методы чтения информации из файлов данных о сетке в заданном формате (рис. 2).

Наиболее важными с точки зрения программной реализации МКЭ являются классы, инкапсулирующие различные типы КЭ (их упругие и физические характеристики, локальные матрицы

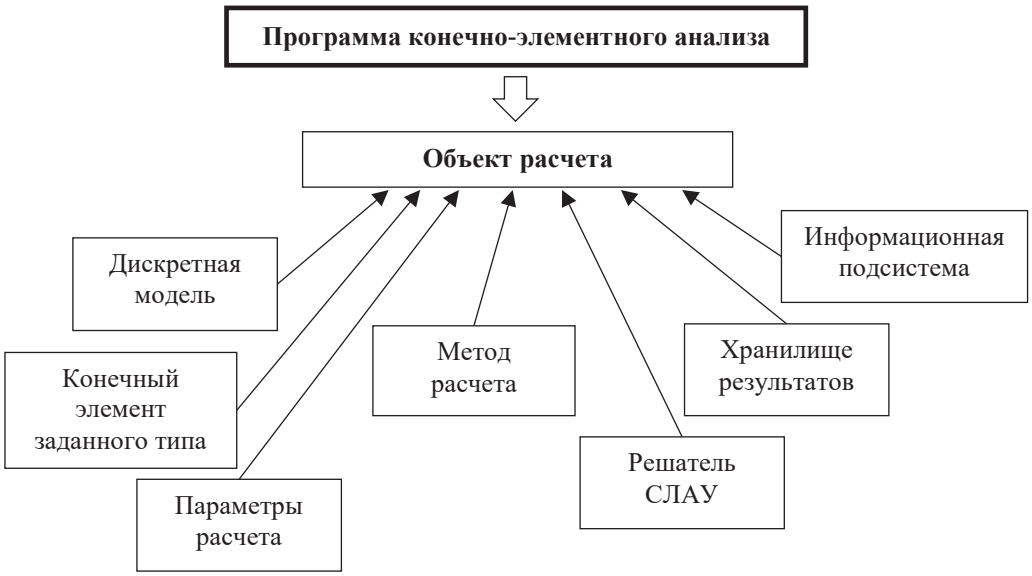

Рис. 1. Общая архитектура библиотеки конечно-элементного анализа PyFEM

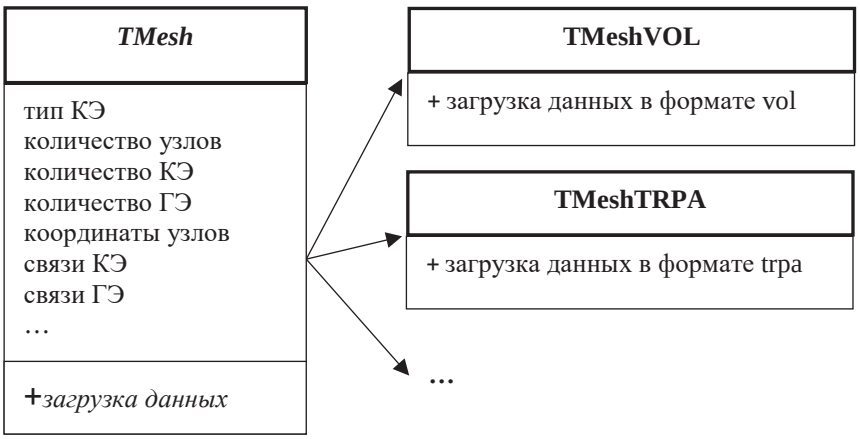

Рис. 2. Иерархия классов, описывающих дискретную модель рассчитываемого объекта

жесткости, массы и демпфирования и т. п.). Для учета их многообразия в PyFEM реализована следующая иерархическая структура классов (рис. 3). Базовым является абстрактный класс TFE, описывающий наиболее фундаментальные свойства изопараметрического КЭ: количество узлов (размерность); площадь сечения для одномерных или толщину для двумерных элементов; упругие свойства; температуру; коэффициент теплового расширения; плотность; коэффициент демпфирования; локальные матрицы жесткости, массы и демпфирования, параметры квадратур для численного интегрирования и т. п. В данном классе определена, но не реализована процедура построения локальных матриц, т. к. она зависит от типа конкретного элемента. Производные от TFE абстрактные классы TFE1D, TFE2D и TFE3D реализуют построение локальных матриц для стандартных одно-, двух- и трехмерных элементов, но не содержат соответствующие процедуры построения функций форм КЭ.

Класс TFE1D2 инкапсулирует линейный двухузловой КЭ и содержит методы построения соответствующих функций формы, которые используются в базовом классе TFE1D для генерации локальных матриц стержневых элементов. Аналогично, классы TFE2D3 и TFE2D4 описывают линейный треугольник и билинейный четырехугольник. Класс TFE2D6, реализованный как наследник от TFE2D3, поддерживает работу с квадратичным (шестиузловым) треугольником. Абстрактные классы TFEP и TFES уточняют построение локальных матриц жесткости, массы и демпфирования КЭ пластин и оболочек. Производные от TFE2D и TFEP классы TFE2D3P, TFE2D4P и TFE2D6P реализуют соответствующие КЭ пластин, а производные от TFE2D и TFES классы TFE2D3S, TFE2D4S и TFE2D6S – оболочек. В свою очередь, дочерние от TFE3D классы TFE3D4, TFE3D8 и TFE3D10 описывают тетраэдральные (4 и 10 узлов) и кубические (8 узлов) КЭ. Такая иерархия классов, описывающих КЭ, является интуитивно понятной и удобной как для использования в расчетах, так и для ее расширения путем добавления новых классов, описывающих специальные типы элементов (например, многослойные оболочки и пластины,

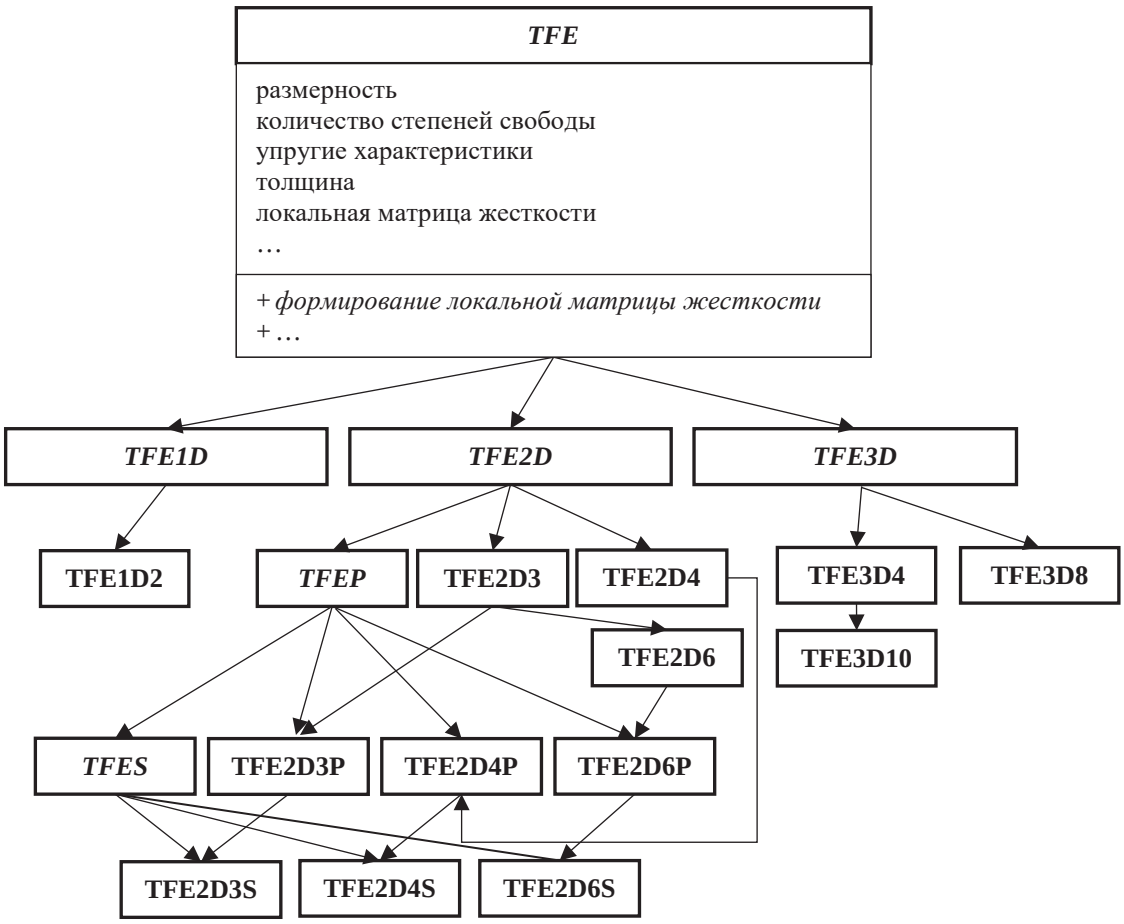

Рис. 3. Иерархия классов, инкапсулирующих изопараметрические конечные элементы разных типов

массивы, композиты и т. п.). Для их реализации пользователю достаточно внести соответствующие изменения в процедуру генерации локальных матриц КЭ.

Для описания параметров расчета в PyFEM используется специальный класс TFEMParams. В нем централизовано хранится информация о типе задачи (статика, динамика); способе решения СЛАУ (прямом или итерационном); точности вычислений; упругих и физических характеристиках, краевых условиях; нагрузках и т. д. (рис. 4). Следует отметить, что большинство параметров (например, упругие характеристики, краевые условия, нагрузки и т. п.) хранятся в виде списка объектов класса TParameter, содержащего информацию о типе параметра (упругие характеристики, начальные или граничные условия, объемная, сосредоточенная или поверхностная нагрузка и т. п.); а также функциональных выражений, описывающих значение параметра и предиката его отбора. С помощью такой организации данных пользователь имеет возможность задавать большинство параметров, различных для разных частей рассчитываемого объекта (например, согласно заданному выражению предиката, могут быть определены различные упругие характеристики объекта для разных его частей).

Наиболее важной составляющей любого ПО конечно-элементного анализа является модуль, непосредственно реализующий алгоритм решения конкретного типа задачи. В силу большого количества видов расчетов (статика, динамика, нелинейность, контактные задачи и т. п.) их универсальная практическая реализация сопряжена с определенными трудностями. В библиотеке PyFEM для этого разработан ряд взаимосвязанных классов, каждый из которых предназначен для решения конкретного типа задачи. Базовый абстрактный класс TFEM содержит наиболее общие свойства и методы, необходимые для программной реализации МКЭ: описание конечно-элементной сетки; все необходимые для расчета параметры; решатель СЛАУ; таблицу результатов и т. д. Центральным методом TFEM является процедура запуска расчета, которая в данном классе является абстрактной, т. к. ее конкретная реализация зависит от типа задачи. В PyFEM на данный момент реализованы два класса-наследника от TFEM: TFEMStatic и TFEMDynamic, предназначенные для решения соответствующих упругих статических и динамических задач (рис. 5).

Такой подход позволяет упростить процедуру расширения библиотеки для решения новых типов задач. Для этого достаточно добавить новый класс-наследник от TFEM (TFEMStatic или TFEMDynamic) и переопределить в нем метод конечно-элементного расчета соответствующего типа задачи. Все необходимые для расчета параметры уже доступны из базовых классов.

Ну и, наконец, центральной частью библиотеки PyFEM является класс TObject. Фактически он является оберткой для TFEM и его наследников и реализует пользовательский интерфейс доступа к библиотеке. Управляя методами данного класса, пользователь может определить новый объект расчета, задать для него конечно-элементную модель, упругие и физические характеристики, краевые условия и нагрузки, а также другие необходимые для расчета параметры. Пользователь также может выбрать способ решения СЛАУ (прямой или итерационный), формат вывода результатов расчета в файл или на экран и т. п. (рис. 6).

Помимо вывода результатов расчета на экран или в файл после расчета (в случае его успеха), автоматически формируется файл результатов в формате JSON [21], в котором сохраняется вся необходимая для последующего анализа информация об объекте расчета и всех полученных результатах. Данный файл можно использовать для анализа полученных результатов. С этой

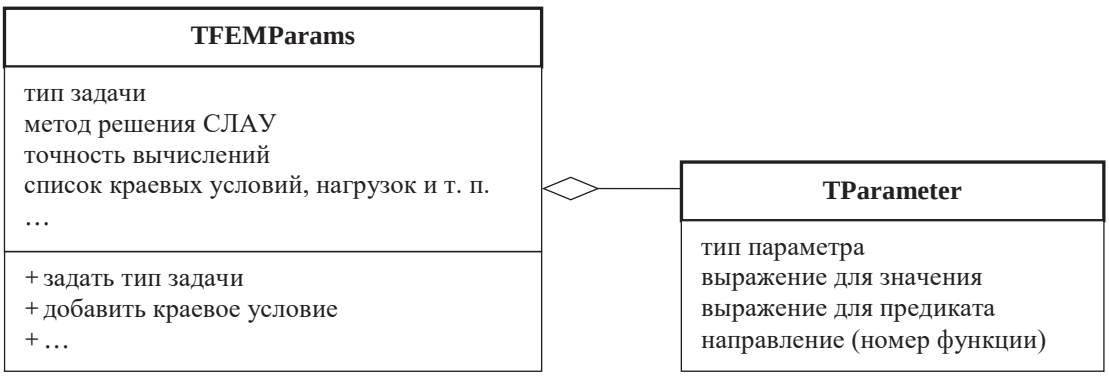

Рис. 4. Организация централизованного хранения параметров решения задачи с возможностью задания предиката отбора

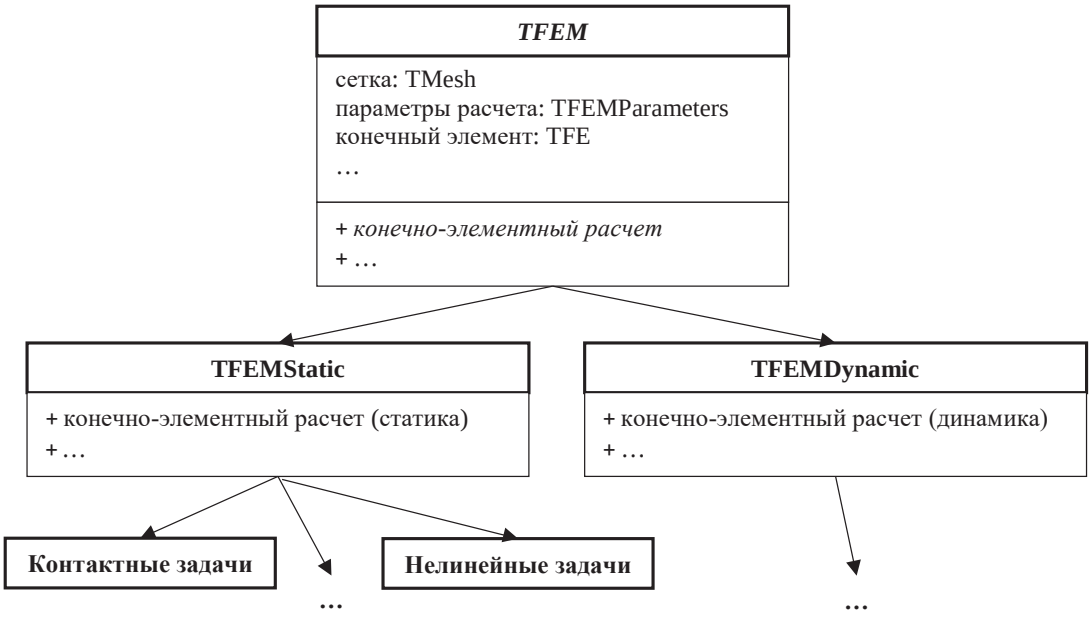

Рис. 5. Иерархия классов, реализующих конечно-элементный расчет разных видов

целью в PyFEM разработаны ряд вспомогательных классов, предназначенных для реализации функций постпроцессора. Главным среди них является TPlot. Он описывает графический интерфейс, в котором отображается визуализация распределения выбранного пользователем фазового параметра по области расчета. Данный класс реализован с помощью библиотеки PyQt5 [22] и графического стандарта OpenGL [23], поэтому он является кроссплатформенным (как и вся библиотека PyFEM) и может запускаться в среде операционных систем Linux, MacOS и Windows.

2. Пример решения задачи с помощью PyFEM

Для выполнения конечно-элементного расчета с помощью PyFEM необходимо выполнить следующие шаги:

1) подключить необходимые для расчета классы и константы:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from core.fem_defs import DIR_X, DIR_Y, DIR_Z
from core.fem_object import TObject
from plot.plot3d import TPlot;
```

2) создать экземпляр класса TObject и задать его параметры:

```
obj = TObject()
if obj.set_mesh(mesh_name):
obj.set_problem_type('static')
obj.set_solve_method('direct')
obj.add_young_modulus('203200')
obj.add_poisson_ratio('0.27')
```

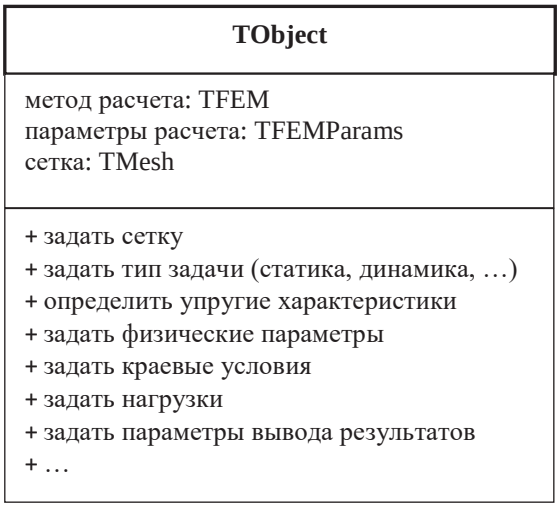

Рис. 6. Пользовательский интерфейсный класс взаимодействия с библиотекой PyFEM

```
obj.add_boundary_condition(DIR_X | DIR_Y | DIR_Z,
'0', 'x == -0.1 or x == 0.1 or y == -0.1 or y == 0.1')
obj.add_concentrated_load(DIR_X, '-2000', 'z == 0')
# …;
```

3) выполнить расчет и вывести/визуализировать результаты:

```
if obj.calc():
obj.print_result()
obj.save_result(res_name)
TPlot(res_name)
return True
# …
```

Например, программа на Python расчета цилиндрической оболочки с жестко защемленными краями под действием внутреннего давления с помощью PyFEM может иметь следующий вид:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from core.fem_defs import *
from core.fem_object import TObject
from plot.plot3d import TPlot

def shell6_test(res_name):
obj = TObject()
if obj.set_mesh('mesh/shell-tube6.trpa'):
obj.set_problem_type('static')
obj.set_solve_method('direct')
obj.add_young_modulus('203200')
obj.add_poisson_ratio('0.27')
obj.add_thickness('0.0369')
obj.add_boundary_condition(DIR_X | DIR_Y | DIR_Z, '0',
'z == 0 or z == 4.014')
obj. add_pressure_load('0.05')
if obj.calc():
obj.print_result()
obj.save_result(res_name)
TPlot(res_name)
return True
return False

if __name__ == '__main__':
shell6_test('shell6_test').
```

В данном примере с помощью метода класса TObject set_mesh() из заданного файла данных загружается информация о дискретной модели оболочки (тип КЭ, количество узлов, КЭ и ГЭ, связях КЭ и ГЭ). После этого, если загрузить данные не удалось, программа завершает свою работу, в противном же случае выполняется дальнейшая настройка исследуемого объекта и запуск его расчета. В случае успеха на экран будет выведена информация о результатах расчета и запущен постпроцессор.

Реализованная в библиотеке PyFEM подсистема вывода информации о ходе решения задачи (классы TProgress и TException) информируют пользователя как о состоянии процесса расчета, так и о возникающих ошибках (рис. 7).Пример работы постпроцессора библиотеки PyFEM приведен на рис. 8. Следует отметить, что время решения данной задачи на компьютере с процессором AMD Ryzen 7 2700X Eight-Core Processor тактовой частотой 3.70 ГГц и объемом оперативной памяти 32 Гбайт под управлением операционной системы Windows 10 время решения данной задачи (без использования параллельных вычислений) составило 4 мин 17 сек. Решение этой же задачи с помощью библиотеки QFEM [11], написанной на C++, заняло 2 мин 51 сек. Таким образом, можно констатировать тот факт, что использование языка программирования Python для реализации конечно-элементных расчетов является перспективным направлением.

```
D:\anaconda3\python.exe D:/Work/python/pyfem/fem_test.py
Object: shell-tube6
Points: 25752
FE: 12748 - triangular shell element (6 nodes)
Assembling global stiffness matrix... 100%
Computation of pressure load... 100%
Use of boundary conditions... 100%
Solving of equation system... 100%
Calculation results... 100%
**************** Success! ****************
Lead time 303.786180 sec
```

Рис. 7. Вывод информации о ходе расчета

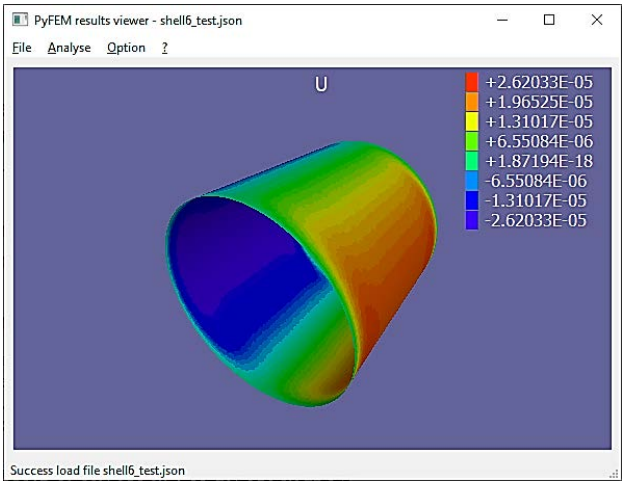

Рис. 8. Пример визуализации результатов расчета

Выводы. Представленный набор классов для реализации конечно-элементного анализа статических и динамических задач механики на языке программирования Python имеет открытую архитектуру и является простым в использовании. Приведенный пример расчета показал, что скорость работы интерпретируемого языка Python является сравнимой с языком C++. В то же время удобство и наглядность языка Python, а также высокая эффективность разработки программ на нем делают его перспективным языком программирования для реализации ПО численного анализа. Возможным направлением развития библиотеки PyFEM является как ее расширение за счет классов, реализующих решение новых типов задач, так и добавление в PyFEM поддержки параллельных вычислений с целью повышения быстродействия вычислений.

ЛИТЕРАТУРА

1. Zienkiewicz O.C., Taylor R.L., Zhu J.Z. The Finite Element Method: Its Basis and Fundamentals. Sixth edition. Butterworth-Heinemann, 2016. 753 p.
2. Design and Engineering Simulation | SIMULIA – Dassault Systemes. URL: https://www.3ds.com/products-services/simulia/.
3. Engineering Simulation & 3D Design Software | Ansys. URL: https://www.ansys.com/.
4. COMSOL Multiphysics ® Modelling Software. URL: https://www.comsol.com/.
5. MSC Nastran – Multidisciplinary Structural Analysis. URL: https://www.mscsoftware.com/product/msc-nastran.
6. Top Finite Element Analysis (FEA) Software List, Reviews, Comparison & Price | TEC. URL: https://www3.technologyevaluation.com/sd/category/finite-element-analysis-fea
7. The dial.II Finite Element Library. URL: https://www.dealii.org/.
8. FreeFEM – An open-source PDE Solver using The Finite Element Method: URL: https://freefem.org/.
9. GetFEM Homepage – GetFEM. URL: http://getfem.org/.
10. OpenSCAD The Programmers Solid 3D CAD Modeller. URL: https://www.openscad.org/.
11. QFEM – The Simple FEM Solver. URL: https://github.com/SeregaGomen/QFEM.
12. qzCAD. URL: https://github.com/qzcad.
13. Страуструп Б. Язык программирования C++. Специальное издание. Москва : Бином-Пресс, 2007. 1104 с.
14. Welcome to Python.org. URL: https://www.python.org/.
15. NumPy. URL: https://numpy.org/.
16. SciPy.org. URL: https://www.scipy.org/.
17. fempy – PiPy. URL: https://pypi.org/project/fempy/.
18. FEniCS Project. URL: https://fenicsproject.org/.
19. polyfem. URL: https://polyfem.github.io/.
20. SfePy: Simple Finite Elements in Python. URL: http://sfepy.org/doc-devel/index.html.
21. JSON. URL: http://json.org/json-ru.html.
22. PyQt5. PyPI. URL: https://pypi.org/project/PyQt5/.
23. OpenGL. The Industry Standard for High Performance Graphics. URL: https://www.opengl.org/.

REFERENCES

1. Zienkiewicz O. C., Taylor R. L., Zhu J. Z. The Finite Element Method: Its Basis and Fundamentals. Sixth edition. Butterworth-Heinemann, 2016. 753 p.
2. Design and Engineering Simulation | SIMULIA – Dassault Systemes. URL: https://www.3ds.com/products-services/simulia/.
3. Engineering Simulation & 3D Design Software | Ansys. URL: https://www.ansys.com/.
4. COMSOL Multiphysics ® Modelling Software. URL: https://www.comsol.com/.
5. MSC Nastran – Multidisciplinary Structural Analysis. URL: https://www.mscsoftware.com/product/msc-nastran.
6. Top Finite Element Analysis (FEA) Software List, Reviews, Comparison & Price | TEC. URL: https://www3.technologyevaluation.com/sd/category/finite-element-analysis-fea
7. The dial.II Finite Element Library. URL: https://www.dealii.org/.
8. FreeFEM – An open-source PDE Solver using The Finite Element Method: URL: https://freefem.org/.
9. GetFEM Homepage – GetFEM. URL: http://getfem.org/.
10. OpenSCAD The Programmers Solid 3D CAD Modeller. URL: https://www.openscad.org/.
11. QFEM – The Simple FEM Solver. URL: https://github.com/SeregaGomen/QFEM.
12. qzCAD. URL: https://github.com/qzcad.
13. Stroustrup B. (2007) Yazyk programmirovaniya C++. Specialnoe izdanie, Moscow: Binom-Press (In Russian).
14. Welcome to Python.org. URL: https://www.python.org/.
15. NumPy. URL: https://numpy.org/.
16. SciPy.org. URL: https://www.scipy.org/.
17. fempy – PiPy. URL: https://pypi.org/project/fempy/.
18. FEniCS Project. URL: https://fenicsproject.org/.
19. polyfem. URL: https://polyfem.github.io/.
20. SfePy: Simple Finite Elements in Python. URL: http://sfepy.org/doc-devel/index.html.
21. JSON. URL: http://json.org/json-ru.html.
22. PyQt5. PyPI. URL: https://pypi.org/project/PyQt5/.
23. OpenGL. The Industry Standard for High Performance Graphics. URL: https://www.opengl.org/.