МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ

Вестник Сыктывкарского университета.
Серия 1: Математика. Механика. Информатика.
Выпуск 4 (33). 2019

УДК 539.3

О ПРИМЕНЕНИИ БИБЛИОТЕК PYTHON ДЛЯ РАСЧЕТА ПЛАСТИН

А. В. Ермоленко, К. С. Осипов

В данной статье приведен пример расчета задачи механики упругих тел по расчету прогиба пластины под нагрузкой с применением пакетов NumPy, SciPy и визуализация результата с помощью пакета MatPlotLib.

Ключевые слова: Python, численные методы, механика упругого тела.

Язык программирования Python применяется в различных областях [1]. Не стала исключением и научная составляющая, а именно математические расчеты. Python «из коробки» имеет инструментарий для решения тривиальных математических задач. Данный функционал можно расширить, подключив различные библиотеки. Рассмотрим некоторые доступные возможности.

Первым, наиболее простым вариантом использования Python в расчетах являются расчеты со встроенным модулем math. Модуль не име-

© Ермоленко А. В., Осипов К. С., 2019.

ет обширного функционала, но вполне подходит для выполнения базовых расчетов. Включает в себя тригонометрические функции, например math.cos, math.sin, логарифмические функции, а также наиболее известные математические константы. Для начала работы с модулем необходимо его подключить командой *import*. Пример использования показан в листинге 1.

```
>>> import math
>>>math.cos(0)
1.0
>>> math.exp(1)
2.718281828459045
>>> math.e
2.718281828459045
>>> math.pi
```

Листинг 1

Для интегрирования, дифференцирования, операций из линейной алгебры для Python разработаны такие библиотеки, как **SciPy, NumPy, SymPy**. Если необходимо использовать символьные вычисления (по аналогии с Maple, MATLAB), то для этого используется библиотека SymPy. Для начала использования библиотеки необходимо ее загрузить. Далее в программном коде следует импортировать либо всю библиотеку SymPy, либо отдельный ее компонент. Важно отметить, что при использовании символьных вычислений необходимо каждую переменную объявлять как символьную. Пример символьного вычисления интеграла указан в листинге 2.

```
>>> from sympy import *
>>> x = Symbol('x')
>>>integrate(x**2 + x + 1, x)
x**3/3 + x** 2/2 + x
```

Листинг 2

Пример символьного вычисления первой и второй производных показан в листинге 3.

```
>>> from sympy import *
>>> x = Symbol('x')
>>> diff(sin(2*x),x)
2*cos(2*x)
>>> diff(sin(2*x),x,2)
-4*sin(2*x)
```

Листинг 3

Для выполнения численных расчетов используется связка библиотек SciPy и NumPy. SciPy — это целая экосистема, ориентированная на выполнение математических, научных и инженерных вычислений. В качестве основной структуры данных используются массивы, реализованные в NumPy. Библиотека представляет обширные возможности для выполнения численных расчетов. Для ее применения в программе необходимо выполнить импортирование функций NumPy следующей командой:

```
>>import numpy as np
```

В дальнейшем можно вызывать функции NumPy через имя np.

На примере расчета круглой пластины [2] покажем использование связки библиотек SciPy и NumPy. Для этого рассмотрим круглую шарнирно закрепленную пластину радиуса R и толщиной h, которая находится под действием осесимметричной нагрузки q_n. Требуется найти прогиб пластины.

Для того чтобы показать использование Python и не погружаться в сложные расчеты, решение поставленной задачи найдем с помощью следующего уравнения Софи Жермен – Лагранжа [3]:

$$D\Delta^2 w = q_n. \tag{1}$$

Здесь w — прогиб пластины, $D = Eh^3/12/(1-\nu^2)$ — цилиндрическая жесткость, E — модуль Юнга, ν — коэффициент Пуассона, $\Delta = \frac{1}{\rho}\frac{d}{d\rho}(\rho\frac{d}{d\rho})$ — оператор Лапласа в полярных координатах.

Граничные условия шарнирного закрепления примем в виде

$$w(R) = 0, w''(R) = 0, \tag{2}_1$$

$$|w(0)| < +\infty, |w''(0)| < +\infty. \tag{2}_2$$

Функция Грина, соответствующая краевой задаче $\{(1), (2)\}$ имеет вид

$$G(\rho,\xi) = \frac{\xi}{4}H(\rho-\xi)(\ln\frac{\rho}{\xi}(\rho^2+\xi^2)+\xi^2-\rho^2)-$$
$$-\frac{\xi}{8}(2\ln R\xi(\rho^2+\xi^2)+3\xi^2-3R^2+\rho^2-\frac{\xi^2\rho^2}{R^2}). \tag{3}$$

Функция Хевисайда H имеет реализацию в пакете NumPy, что и будет использовано в дальнейшем решении задачи. Для представления функции Грина в языке программирования Python воспользуемся лямбда-функциями. Они имеют более строгий и лаконичный вид,

чем обычное, классическое определение функции через def и понятнее отображают смысл математической функции в коде. Лямбда-функции встроены в Python по умолчанию, поэтому для их использования нет необходимости подключать какой-либо пакет. В результате функция Грина в Python имеет следующий вид:

```
>>> green = lambda xi, rho: (xi / 4) * \
np.heaviside(rho-xi, 1) * (np.log(rho / xi) * (rho ** 2 + \
xi ** 2) + xi ** 2 - rho ** 2) - (xi / 8) * \
(np.log(R / xi) * (2 * rho ** 2 + 2 * xi ** 2) + \
3 * xi ** 2 - 3 * R ** 2 + rho ** 2 - \
(xi ** 2 * rho ** 2) / R ** 2)
```

Как можно увидеть, здесь функция Хевисайда вызывается с двумя параметрами. Первый параметр это само выражение, которое передается в функцию. Второй параметр определяет поведение функции при случаях, когда ее выражение равно нулю.

Решение краевой задачи {(1), (2)} с использованием функции (3) имеет вид

$$w(\rho) = \frac{1}{D} \int_0^R G(\rho, \xi) q_n(\xi) d\xi. \tag{4}$$

Для расчета возьмем $q_n(x) = \sin(x)$. Для упрощения представления задачи введем функцию qnGreen, которая будет представлять подынтегральное выражение и выглядит так:

```
>>> qnGreen = lambda xi, rho : np.sin(xi) * green(xi, rho)
```

Далее введем точки, на которых рассматривается данная задача. Если принять $R = 100$, а количество точек равным 100, то тогда получится

набор точек от 0 до 100 с шагом 0,01. Для этого в программе воспользуемся основной структурой данных — массивами numpy и функцией arange, которая задает последовательность чисел в массиве. Для этого запишем следующую команду:

```
>>> rho = np.arange(0, R, 0.01)
```

Функция arange принимает 3 значения — начало и конец последовательности, а также ее шаг. После всех выполненных подготовительных процедур можно приступить к численному интегрированию, для этого используется набор функций integrate из пакета SciPy. При решении данной задачи использовалась функция quad. Загрузка функций integrate выглядит так:

```
>>> import scipy.integrate as integrate
```

При использовании функции quad вычисление прогиба имеет вид

```
>>> for i in range(1,100):
w[i] = (1 / D) * integrate.quad(qnGreen, 0, 1,
args=(rho[i],))[0]
```

Функция quad возвращает массив значений из двух элементов, в котором первый элемент — это само значение интеграла, а второй — погрешность вычислений.

Для визуализации расчетов используем библиотеку matplotlib. Визуализируем решение данной задачи с помощью данного пакета. Для этого импортируем набор функций pyplot, которые позволяют строить графики в стиле MATLAB. В таком случае используется команда

```
>> import matplotlib.pyplot as plt
```

Теперь к функциям pyplot можно обращаться через алиас plt. Отрисовка графиков показана в листинге 4.

```
rho = np.arange(0, R, 1)
fig, axs = plt.subplots()
plt.gcf().canvas.set_window_title("Прогиб")
plt.plot(rho1, w)
progibla = [round(progib, 8) for progib in w]
axs.yaxis.set_ticklabels(progibla)
plt.xlabel('rho, см')
plt.ylabel('w, см')
plt.show()
```

Листинг 4

В листинге 4 первая строчка отвечает за создание макета для размещения графика со вспомогательными элементами. Если функцию subplots вызывать без параметров, то по умолчанию она вернет окно с одним графиком. Функция возвращает массив значений из двух элементов, первый из которых — это объект фигуры, а второй — это объекты осей графика. Во второй строке указывается заголовок вызываемого окна, в котором будет отрисован график. Третья строка рисует линии осей и саму фигуру, координаты которой указаны в массивах точек. В четвертой и пятой строках инициализируется массив с точками, округленными до определенного значения, далее эти точки отмечаются на оси Y. Функция label в шестой и седьмой строках отмечает подписи

под осями X и Y. Последняя строка обеспечивает отображение подготовленной ранее фигуры на экране со всеми указанными свойствами.

На рис. 1 представлен пример визуализации задачи по расчету прогиба пластины со следующими параметрами:

$$E = 2 \cdot 10^6 \text{ кГ/см}^2, R = 100 \text{ см}, h = 1 \text{ см}, q_n = \sin x \text{ кГ/см}^2.$$

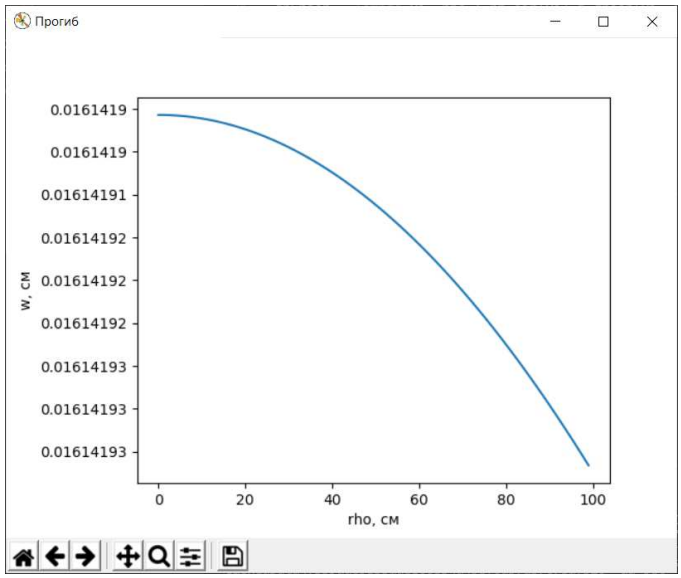

Рис. 1. Визуализация расчета прогиба пластины

Подводя итог, отметим, что Python с математическими пакетами представляет собой мощный инструмент для решения задач механики пластин и оболочек. Решение задачи вычисления прогиба пластины достаточно легко решается средствами пакетов NumPy и SciPy, а пакет matplotlib позволяет визуализировать полученный результат, при этом, чтобы отрисовать двумерный график, достаточно лишь определить numpy массив с точками x и y, инициализировать графическое окно для отрисовки графика, передать массивы с точками и задать

дополнительные параметры. Таким образом, решение задач с помощью языка Python также просто, как и в математических пакетах Maple или MATLAB, но при этом разработчик получает более мощный и более производительный инструмент в виде математических пакетов Python.

Список литературы

1. **Вандер Плас Дж.** Python для сложных задач. Наука о данных и машинное обучение. СПб.: Питер, 2018. 576 с.

2. **Ермоленко А. В.** Расчет круглых пластин по уточненным теориям // *Вестник Сыктывкарского университета. Сер. 1: Математика. Механика. Информатика. 2006. Вып. 6. С. 78–86.*

3. **Михайловский Е. И., Бадокин К. В., Ермоленко А. В.** Теория изгиба пластин типа Кармана без гипотез Кирхгофа // *Вестник Сыктывкарского университета. Сер. 1: Математика. Механика. Информатика. 1999. Вып. 3. С. 181–202.*

Summary

Yermolenko A. V., Osipov K. S. On using Python libraries to calculate plates

This article provides an example of calculating the elastic mechanics problem of calculating plate deflection under load using NumPy, SciPy packages and visualizing the result using the MatPlotLib package.

Keywords: Python, numerical methods, elastic body mechanics.

References

1. **Vander Plas Dzh.** *Python dlya slozhnykh zadach. Nauka o dannykh i mashinnoye obucheniye* (Python for complex tasks. Data Science and Machine Learning), St. Petersburg: Peter, 2018, 576 p.

2. **Yermolenko A. V.** Raschet kruglykh plastin po utochnennym teoriyam (Calculation of round plates according to revised theories), *Bulletin of Syktyvkar University*, Series 1: Mathematics. Mechanics. Informatics, 2006, № 6, pp. 78–86.

3. **Mihajlovskii E. I., Badokin K. V., Yermolenko A. V.** Teorija izgiba plastin tipa Karmana bez gipotez Kirhgofa (The theory of bending of Karman-type plates without the Kirchhoff's hypotheses), *Bulletin of Syktyvkar University*, Series 1: Mathematics. Mechanics. Informatics, 1999, № 3, pp. 181–202.

Для цитирования: Ермоленко А. В., Осипов К. С. О применении библиотек Python для расчета пластин // *Вестник Сыктывкарского университета. Сер. 1: Математика. Механика. Информатика. 2019. Вып. 4 (33). С. 86–95.*

For citation: Yermolenko A. V., Osipov K. S. On using Python libraries to calculate plates, *Bulletin of Syktyvkar University. Series 1: Mathematics. Mechanics. Informatics*, 2019, 4 (33), pp. 86–95.

СГУ им. Питирима Сорокина *Поступила 11.11.2019*