УДК 519.6

ИСПОЛЬЗОВАНИЕ ВОЗМОЖНОСТЕЙ СРЕДЫ ПРОГРАММИРОВАНИЯ PYTHON ПРИ ИЗУЧЕНИИ МАТЕМАТИЧЕСКИХ ДИСЦИПЛИН В ТЕХНИЧЕСКОМ ВУЗЕ

Б.А. Акишин

Донской государственный технический университет, Ростов-на-Дону

akiboralex@mail.ru

Аннотация

Исследованы возможности и приведены примеры использования библиотек Python при решении типовых математических задач. Проанализированы особенности интерпретации полученных результатов.

Ключевые слова*: система компьютерной математики, язык программирования Python, библиотеки функций, символьные расчеты, линейная алгебра, математический анализ, дифференциальные уравнения*

В работе [1] приведено обоснование целесообразности активного использования в процессе изучения математики в школе и вузе некоммерческих систем компьютерной математики (СКМ), в частности, программ *Maxima* и *GeoGebra*. В то же время, студенты ряда специальностей в технических вузах, в том числе, Донском государственном техническом университете, изучают язык программирования *Python*, поэтому естественно привлекать их и к решению математических задач именно в среде *Python*, тем более, что в его основных научных библиотеках *numpy, sympy, scipy, pandas, numpy* и других реализовано большинство известных аналитических и численных алгоритмов решения уравнений, задач линейной алгебры, вычисления пределов, производных и интегралов, аппроксимации, решения дифференциальных уравнений и их систем, задач теории вероятностей и математической статистики и т. д., а пакет *matplotlib* обладает хорошо развитыми возможностями визуализации двумерных и трехмерных данных.

Приведем несколько примеров аналитических (символьных) решений в Python задач линейной алгебры и математического анализа.

Использовав интерактивную оболочку *IPython*, загрузим библиотеку символьных вычислений sympy и установим режим графической печати Latex, который позволяет отображать исходные данные и результаты решения в привычном математическом виде:

```
In [1]: from sympy import *
   ...: init_printing(use_latex=True)
```

Пример 1. Решить неопределенную систему линейных алгебраических уравнений (СЛАУ) [2]:

$$\begin{cases} x_1 + x_2 - x_3 + x_4 = 2, \\ 2x_1 + x_2 + 3x_3 - x_4 = -1, \\ 3x_1 + 2x_2 + 2x_3 = 1, \\ x_1 + 4x_3 - 2x_4 = -3. \end{cases}$$

При решении подобных СЛАУ на практических занятиях по линейной алгебре обычно: а) проверяют условия выполнения теоремы Кронекера–Капелли; б) решают СЛАУ методом Гаусса (или одной из его модификаций).

Решим пример средствами *Python.* Для записи уравнений обычно применяют функцию *Eq(expr1, expr2)* в библиотеке *sympy.* Если выражения *expr1* и *expr2* равны, то функция *Eq()* возвращает значение True.

```
In [2]: x1,x2,x3,x4=symbols('x1 x2 x3 x4')
   ...: eq1=Eq(x1+x2-x3+x4,2)
   ...: eq2=Eq(2*x1+x2+3*x3-x4,-1)
   ...: eq3=Eq(3*x1+2*x2+2*x3,1)
   ...: eq4=Eq(x1+4*x3-2*x4,-3)
   ...: eq1,eq2,eq3,eq4
Out[2]:
```

$$(x_1 + x_2 - x_3 + x_4 = 2, \quad 2x_1 + x_2 + 3x_3 - x_4 = -1,$$
$$3x_1 + 2x_2 + 2x_3 = 1, \quad x_1 + 4x_3 - 2x_4 = -3)$$

Для решения систем линейных уравнений предназначена, в первую очередь, функция *linsolve(...)*, в качестве аргументов которой задают списки уравнений и неизвестных переменных:

```
In [3]: linsolve([eq1,eq2,eq3,eq4],[x1,x2,x3,x4])
Out[3]:
```

$$\{(-4x_3 + 2x_4 - 3, \quad 5x_3 - 3x_4 + 5, \quad x_3, \quad x_4)\}$$

Общее решение этой неопределенной СЛАУ получено в привычном для студентов виде кортежа, где базисные переменные выражены через свободные.

Таким образом, аналитическое решение систем линейных алгебраических уравнений небольшой размерности, в том числе, и неопределенных, в среде *Python* не представляет особых сложностей, не требует дополнительных проверок совместности и нагляднее, чем в некоторых СКМ, например, MathCAD.

Пример 2. Вычислить предел [2]: $\lim\limits_{x\to\infty}\left(1+\frac{5}{x}\right)^{3x}$.

Заметим, что решение данного примера на практике сводится ко второму специальному пределу.

Для аналитического вычисления пределов в *sympy* имеется функция *limit(...);* для записи символа ∞ (бесконечность) используется запись oo (две буквы «o»):

```
In [2]: x=symbols('x')
   ...: limit((1+5/x)**(3*x), x, oo)
Out[2]:

 15
e
```

У функции *limit()* есть невычисляемый эквивалент – оператор *Limit()*, который возвращает символьный объект типа *'sympy.series.limits.Limit'* (невычисленный предел). Для вычисления символьного объекта, созданного невычисляемым оператором, нужно использовать метод *doit()*. Используя этот прием, а также упомянутую выше функцию *Eq(),* можно окончательный результат вычисления предела представить в наглядной форме:

```
In [3]: y=Limit((1+5/x)**(3*x), x, oo)
   ...: Eq(y,y.doit())
Out[3]:

lim (1 + 5/x)^(3x) = e^15
x→∞
```

Если функция *limit()* не может вычислить предел, например, он не существует, то в зависимости от режима *init_printing* она возвращает на экран либо строку с невычисляемым эквивалентом *Limit(),* либо невычисленное исходное выражение.

Таким образом, результаты вычисления в *Python* достаточно простых пределов весьма наглядны, однако в более сложных случаях нужно использовать возможности других СКМ: *Maxima, GeoGebra* или *MathCAD.*

Пример 3. Вычислить неопределенный интеграл [2]: $\int x^3 \sin(4x^2)dx$.

Отметим, что достаточно часто при вычислении такого типа интегралов методом «по частям» студенты неправильно выбирают части.

Для символьного интегрирования в *Python* предназначена функция *integrate(...)* из библиотеки *sympy.* С ее помощью можно вычислять как неопределенные, так и определенные интегралы. Первым аргументом функции должно быть символьное выражение, которое будет интегрироваться, вторым – переменная интегрирования или кортеж, состоящий из имени переменной и ее нижнего и верхнего пределов. Если второй аргумент – только имя, то вычисляется неопределенный интеграл, т. е. первообразная подынтегральной функции.

У функции *integrate()* также есть невычисляемый эквивалент – оператор *Integral().* Так же, как и в случае пределов, чтобы затем вычислить интеграл, нужно использовать метод *doit().* Воспользуемся этим фактом, чтобы сразу получить ответ в наглядной форме:

```
In [2]: Fx=Integral(x**3*sin(4*x**2),x)
   ...: Eq(Fx,Fx.doit())
Out[2]:
```

$$\int x^3 \sin(4x^2)\,dx = -x^2\cos(4x^2)/8 + \sin(4x^2)/32$$

Заметим, что sympy не включает в результат произвольную постоянную интегрирования. Однако константу можно добавить, если сформулировать задачу как решение соответствующего дифференциального уравнения (см. пример 4)

Проверим результат дифференцированием (к объекту применяется метод *diff()*):

```
In [3]: Fx.diff(x)
Out[3]:
```

$$x^3 \sin(4x^2)$$

Если первообразную не удалось найти, то, как было указано ранее, отображается невычисляемый объект *Integral()* в виде исходного выражения, а функция *Eq()* возвращает значение True, например, [1].

```
In [4]: integrate((3*x+1)/sqrt(5*x**2-2*x+1),x)
Out[4]:
```

$$\int \frac{3x+1}{\sqrt{5x^2-2x+1}}\,dx$$

```
In [5]: Fx1=Integral((3*x+1)/sqrt(5*x**2-2*x+1),x)
   ...: Eq(Fx1,Fx1.doit())
Out[5]:
```

True

Считается, что вычисление первообразных является для студентов одной из наиболее сложных задач математического анализа, да и не все СКМ справляются с отдельными примерами. Как показывает практика, во многих случаях вычисление первообразных в среде *Python* предпочтительнее, так как оно просто и наглядно, но в сложных примерах нужно пробовать использовать другие программы, например, *Maxima, GeoGebra* и т. д.

Пример 4. Найти общее решение обыкновенного дифференциального уравнения (ОДУ) [2]:

$$x \cdot y' - 2y = 2x^4.$$

Данное ОДУ является линейным первого порядка.

При решении ОДУ в *sympy* обычно используется следующая последовательность инструкций: а) объявляются символьная независимая переменная *x* и символьная функция *f*, которая будет представлять решение; б) создается объект, представляющий уравнение; в) решается дифференциальное уравнение с помощью функции *dsolve()*, у которой первым аргументом является объект уравнения, а вторым –искомая функция. Итак, получаем:

```
In [2]: x = Symbol('x')
   ...: f = Function('f')

In [3]: deq=Eq(x*f(x).diff(x)-2*f(x), 2*x**4)
   ...: deq
Out[3]:
```

$$x\frac{d}{dx}f(x) - 2f(x) = 2x^4$$

```
In [4]: dsolve(deq, f(x))
Out[4]:
```

$$f(x) = x^2(C_1 + x^2)$$

Обращаем внимание на то, что решение содержит произвольную переменную C_1.

Функция *dsolve()* может использовать несколько различных методов решения ОДУ. Чтобы получить список методов, которые можно использовать для решения конкретного уравнения, следует использовать инструкцию *classify_ode (уравнение, выражение/функция)*.

Функция *dsolve()* решает аналитически большинство известных типов ОДУ и систем, интегрируемых в квадратурах. Решения представляются, как правило, в наглядном виде.

Опция *ics* функции *dsolve()* позволяет задавать граничные условия и решать задачу Коши, однако в настоящий момент эта опция реализована только для отдельных типов уравнения и методов решения, так что получать аналитические решения задачи Коши студентам проще в других СКМ, например, в *Maxima*[2].

Выводы. Системы компьютерной математики весьма полезны при изучении вузовской математики. Опыт показывает, что в процессе общения с компьютером студент не только приобретает навыки работы с программами, которые пригодятся ему в дальнейшем, но и углубляет свои знания по математике, что зачастую приводит к освоению новых математических методов, заложенных в современные программы.

Заметим, что ответы, получаемые студентом на бумаге, зачастую отличаются от ответов, выдаваемых СКМ. В таких случаях студент должен провести углубленный анализ, разобраться в причинах несоответствия и довести решение до конца.

Если студент знаком с основами программирования на Python, то при освоении разделов математики он может с успехом использовать его библиотеки и пакеты. В настоящей статье приведены примеры аналитического (символьного) решения лишь некоторых типовых математических задач и не рассматривались численные алгоритмы. В чем-то процесс решения и представление результатов на Python имеют преимущества перед другими СКМ, а в чем-то недостатки.

СПИСОК ЛИТЕРАТУРЫ

1. *Акишин Б.А., Воронцова В.А.* Особенности использования систем компьютерной математики при изучении математических дисциплин в техническом вузе // Математическое образование в школе и вузе: инновации в информационном пространстве (MATHEDU' 2018): материалы VIII Международной научно-практической конференции (Казань, 17–21 октября 2018 г.). Казань: Изд-во Казан. ун-та, 2018, С. 202–206.

2. *Акишин Б.А., Черкесова Л.В., Галабурдин А.В.* Решение математических задач с помощью пакета Maxima. Учеб. пособие. Ростов н/Д: Издательский центр ДГТУ, 2015, 100 с.

USING THE PYTHON POSSIBILITIES IN STUDYING OF THE MATHEMATICAL SUBJECTS IN TECHNICAL HIGHER EDUCATION

B.A. Akishin

Don State Technical University, Rostov-on-Don

akiboralex@mail.ru

Abstract

Opportunities are investigated and examples are provided of using Python libraries at the solution of the typical mathematical tasks. Features of interpretation of the received results are analyzed.

Keywords*: computer mathematics system, Python programming language, function libraries, symbolic calculations, linear algebra, mathematical analysis, differential equations*

REFERENCES

1. *Akishin B.A., Voroncova V.A.* Osobennosti ispol'zovaniya sistem komp'yuternoj matematiki pri izuchenii matematicheskih disciplin v tekhnicheskom vuze // Matematicheskoe obrazovanie v shkole i vuze: innovacii v informacionnom prostranstve (MATHEDU' 2018): materialy VIII Mezhdunarodnoj nauchno-prakticheskoj konferencii (Kazan', 17–21 oktyabrya 2018 g.). Kazan': Izd-vo Kazan. un-ta, 2018, S. 202–206.

2. *Akishin B.A., Cherkesova L.V., Galaburdin A.V.* Reshenie matematicheskih zadach s pomoshch'yu paketa Maxima. Ucheb. posobie. Rostov n/D: Izdatel'skij centr DGTU, 2015, 100 s.

СВЕДЕНИЯ ОБ АВТОРЕ

АКИШИН Борис Алексеевич – кандидат технических наук, доцент кафедры математики и информатики Донского государственного технического университета.

Boris Alekseevich AKISHIN – candidate of technical Sciences, Associate Professor, Department of mathematics and Informatics, Don State Technical University.

email: akiboralex@mail.ru

Материал поступил в редакцию 7 сентября 2019 года