MATLAB and Python in Teaching Calculus

Árpád Horváth
Alba Regia Technical Faculty of Óbuda University,
H-8000 Székesfehérvár, Budai út 45., Hungary
Email: horvath.arpad@amk.uni-obuda.hu

***Abstract*—This year Obuda University have been started a course for the students to teach calculus with the aid of computer. As of many students learn programming in Python on an other course there is two group of students who use Python for learning calculus and three groups of students who use MATLAB for it. In this article I summarize the advantages and disadvantages of the symbolic mathematical module of both programming language.**

I. Introduction

In the new curriculum of Obuda University there is a course unit named Mathematics I. that is common for most of the BSc students of the university. In that curriculum there is lessons (2 hours/week), practical course (2 hours/week) and there is a laboratory practice (1 hours/week in average). This laboratory practice lasts two hours in every second weeks or – for other groups – three hours in every third weeks.

Those students who learns Mathematics I. at the Alba Regia Tehchnical Faculty of the Obuda University learns Python[1], [2] as the first programming languages too in an other course.

There are many differences between the Python and MATLAB languages, so the students can be confused. Just to mention some differences: MATLAB has a 1-based indexing while Python 0-based, Python uses square brackets for indexing, MATLAB the round brackets; Python uses the `return` keyword for the return values of a function while MATLAB has a radically different method; Python uses the operator `**` to calculate the power, while MATLAB uses the `^` operator.

These and the many other differences can be confusing for the students who have not written any program code before. In this article I will discuss how easy they are to install at home and to learn.

It is worth to mention that using Python in the curriculum of the Obuda University is not new. There were a successful experiment on the Obuda University in teaching combinatorics using Python.[3]

II. Installation

A. MATLAB and Symbolic math toolbox

If one want to use MATLAB. They need to buy MATLAB, or to have an e-mail address that belongs to the domain of the university that has a MATLAB licence for its students. In our university there is a separate process to get an e-mail address that ends with uni-obuda.hu. If someone wants to get a Microsoft Office 365 as students, he or she also need an e-mail like that. The student can get the e-mail address at the MS Office homepage of the University. This is not well documented on the webpage of the university, were there is a documentation about the installation. This is a quite good documentation except that.

If someone have the proper e-mail address he or she must register on the webpage of Mathworks, the developer of MATLAB and the software can be downloaded or used online in a web browser. For teaching calculus we need the Symbolic math toolbox[4] which is quite simple to install.

B. Python and sympy package

If you want to use python, you can use more Python distributions and Integrated Development Environments (IDE). In this article I will focus on installation on Windows. On Linux it is easier to install Python, but most of the students use Windows.

I installed most distributions of Python on Windows recently. The most robust combination was the one can be downloaded from the python.org with the PyCharm IDE to ease the usage. This Python distribution includes a simple IDE called IDLE. This includes an Interactive shell, like the Command Window in MATLAB, and one can edit Python files with it. It can fulfill the needs of the course. Without PyCharm you need the command line to install the sympy package, but it is not too difficult to do and a normal user without administrator privilege can do it. I created and shared a video to help the students to do it at home.

I think that PyCharm Professional is the most advanced IDE for Python right now. It can be reached for the students of universities without any fee for one years. For this one needs to have an e-mail address belonging to the domain of the university. The Professional version helps the advanced usage of Python, like web development (django, flask). But the most important features is available in the Community and Edu version of PyCharm. These versions are free for anyone and allows to install packages easily.

I have tried two other distributions on Windows: Anaconda and Enthought Python. The main goal of these packages is to help data analysis and big data applications. Both of them comes with a lot of packages by default including sympy. The installation of these distributions have difficulties on systems where there is a space or accentuated letter (á) in the path name. On one of my computers I had a user name with space and accent, and I have not managed to install that Python distributions there.

The screen shots were made by the IPython 6.1.0 interactive shell in Linux with Python 3.5.2 and SymPy 1.1.1. This is like the one can be found in the Anaconda Python distribution.

C. *Sympy live*

There is an online way to run sympy. In the live.sympy.org website we can do basic calculation with sympy. The mathematical expression are rendered with LaTeXso they looks like in a math book. However there is no possibility to run more complex code like cycles and function definition.

III. Creating symbols

A. *Symbolic math toolbox*

Creating the real symbols x, y and t with the Symbolic math toolbox is as easy as:

```
syms x y t
```

We can add some assumptions for some special tasks:

```
syms a positive
syms l n integer
```

but we did not used this feature in the course.

B. *SymPy package*

When we use SymPy package we need to import its functions and the variables we need. If we want to render the expressions in a nice looking way (in the environment that support LaTeX or UTF-8) we can initialize the printing of expressions too:

```
from sympy import *
from sympy.abc import x, y, theta
init_printing()
```

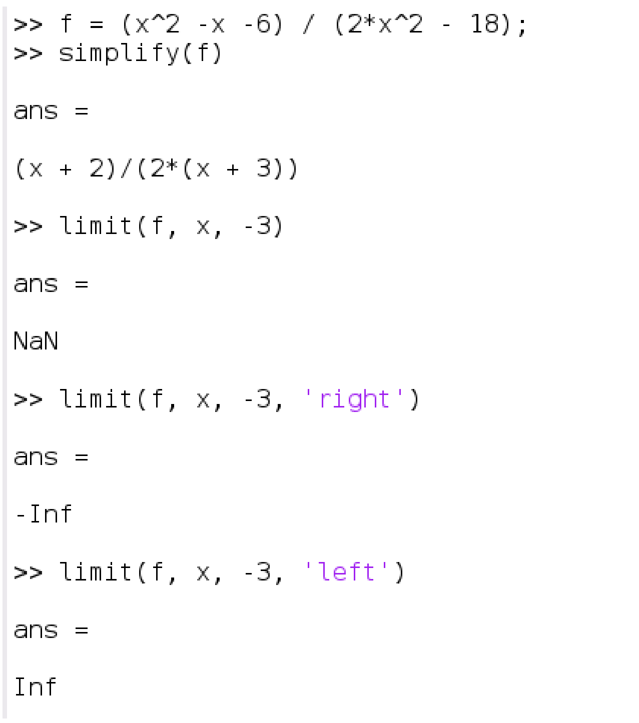
IV. Expanding, simplifying expressions, substitute values, limes of the functions

Both of the too examined tools (the Symbolic math toolbox and SymPy) has the same function (or method) names, but its usages is sometimes different. Both has the names `expand`, `simplify` and `subs`, `limit`. They have usually almost the same syntax. The main difference is that the is a method of an expression instance in Python as we can see in the example below.

They have some constants like `pi` and we can get the numerical values of expressions if we substituted all of the variables with a given value. If we want to give expressions the most important difference is that we use different operator for the power.

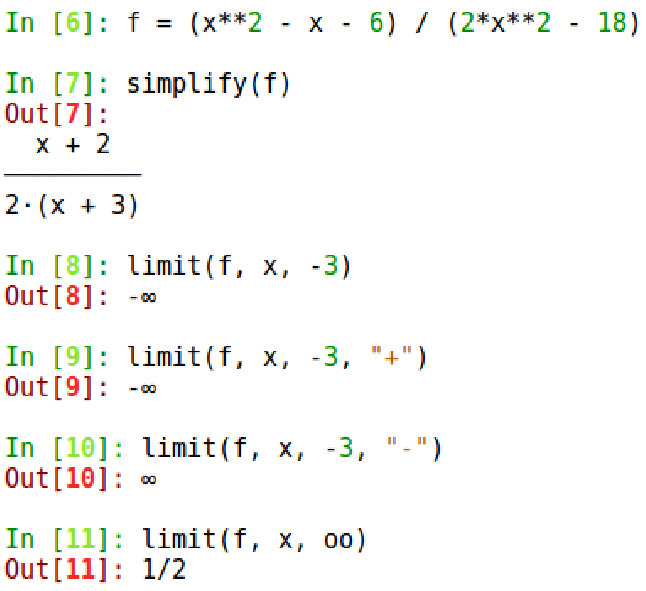
In the `limit` function we can use $\pm$inf as the value the variable goes to, but we have different sign for it. In MATLAB we use `Inf`, in sympy `oo` (two small O letters). If the right and left limit is different, MATLAB gives NaN value for the "simple" `limit` function. In that case Python uses right limit as default.

The examples for simplifying function and limit calculation can be found in the Figures 1 and 2, examples for expansion and substitution can be found in the Figures 3 and 4, for MATLAB and Python.



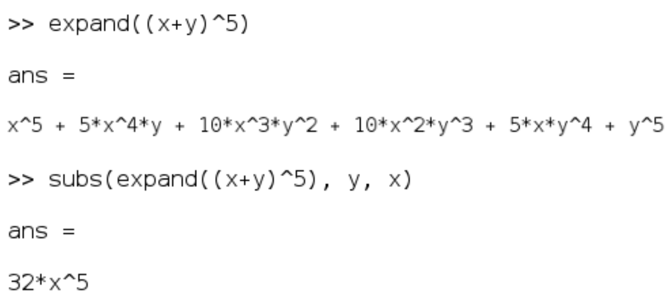
```
>> f = (x^2 -x -6) / (2*x^2 - 18);
>> simplify(f)

ans =

(x + 2)/(2*(x + 3))

>> limit(f, x, -3)

ans =

NaN

>> limit(f, x, -3, 'right')

ans =

-Inf

>> limit(f, x, -3, 'left')

ans =

Inf
```

Figure 1. Simplifying function and limits in MATLAB



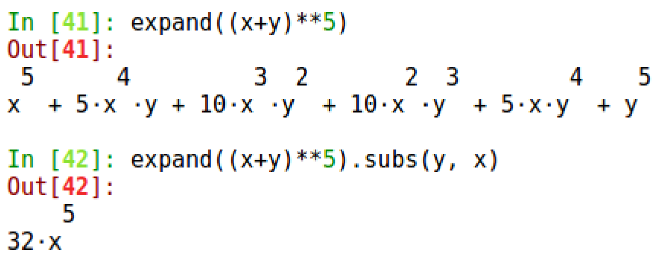
```
In [6]: f = (x**2 - x - 6) / (2*x**2 - 18)

In [7]: simplify(f)
Out[7]:
  x + 2
─────────
2⋅(x + 3)

In [8]: limit(f, x, -3)
Out[8]: -∞

In [9]: limit(f, x, -3, "+")
Out[9]: -∞

In [10]: limit(f, x, -3, "-")
Out[10]: ∞

In [11]: limit(f, x, oo)
Out[11]: 1/2
```

Figure 2. Simplifying function and limits in Python



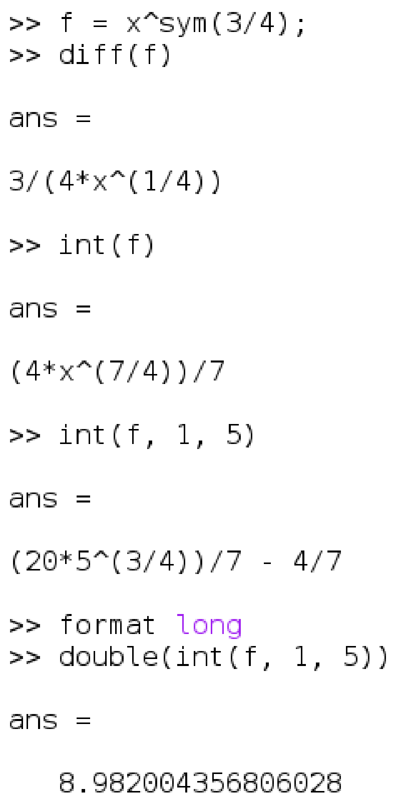
```
>> expand((x+y)^5)

ans =

x^5 + 5*x^4*y + 10*x^3*y^2 + 10*x^2*y^3 + 5*x*y^4 + y^5

>> subs(expand((x+y)^5), y, x)

ans =

32*x^5
```

Figure 3. Extension and substitution in MATLAB



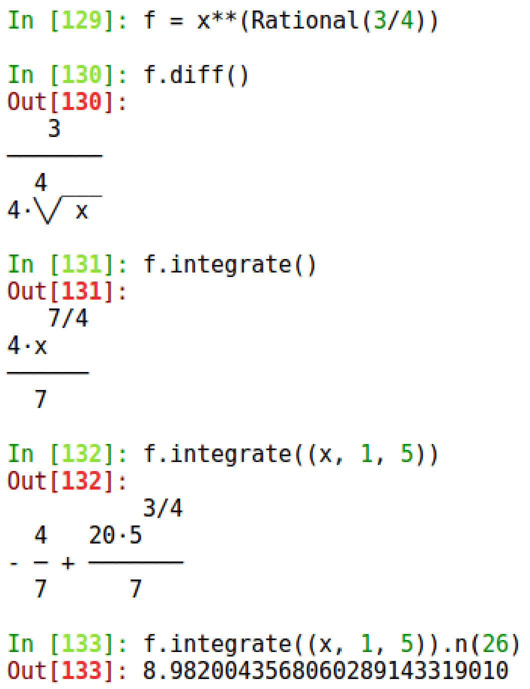
```
In [41]: expand((x+y)**5)
Out[41]:
 5      4          3  2        2  3         4    5
x  + 5·x ·y + 10·x ·y  + 10·x ·y  + 5·x·y  + y

In [42]: expand((x+y)**5).subs(y, x)
Out[42]:
    5
32·x
```

Figure 4. Extension and substitution in Python



```
>> f = x^sym(3/4);
>> diff(f)

ans =

3/(4*x^(1/4))

>> int(f)

ans =

(4*x^(7/4))/7

>> int(f, 1, 5)

ans =

(20*5^(3/4))/7 - 4/7

>> format long
>> double(int(f, 1, 5))

ans =

   8.982004356806028
```

Figure 5. Derivative and integral of a function in sympy



```
In [129]: f = x**(Rational(3/4))

In [130]: f.diff()
Out[130]:
   3
───────
  4
4·√ x

In [131]: f.integrate()
Out[131]:
   7/4
4·x
──────
  7

In [132]: f.integrate((x, 1, 5))
Out[132]:
          3/4
  4   20·5
- ─ + ───────
  7      7

In [133]: f.integrate((x, 1, 5)).n(26)
Out[133]: 8.9820043568060289143319010
```

Figure 6. Derivative and integral of a function in sympy

V. Differentiation and Integration

We can see the differentiation and integration of the $\sqrt[4]{x^3}$ on Figure 5. At the end we calculate the numerical value of the definit integral.

In MATLAB we use the functions `int` and `diff` to get the (definit or indefinit) integral or the derivative of the function respectively.

More functions can be used in two form in sympy. As we used `limit` earlier, and as a method of the expression, as we used the `subs`. We can use `integrate`, `diff` and `limit` as function and as a method. Usually the letter is the easier to read, so we use it in such a way. Using as a method we must not give the expression as parameter, so we have one less parameter.

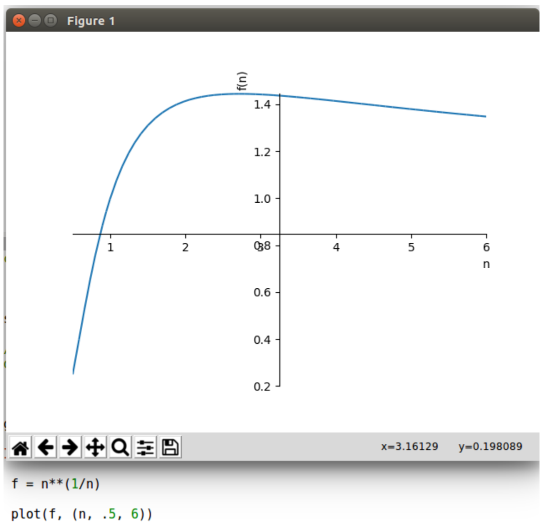
We can see the differentiation and integration of the $\sqrt[4]{x^3}$ on Figure 6. At the end we calculate the numerical value of the definit integral using 26 significant digit.

VI. Plotting

Both sympy and MATLAB can plot function given by symbolic values. MATLAB have the ezplot function , sympy the plot function to plot the figures. Both of these functions have the opportunity to control the parameters of the plot. A plot made by MATLAB and SymPy can be seen in Figures 7 and 8 respectively.

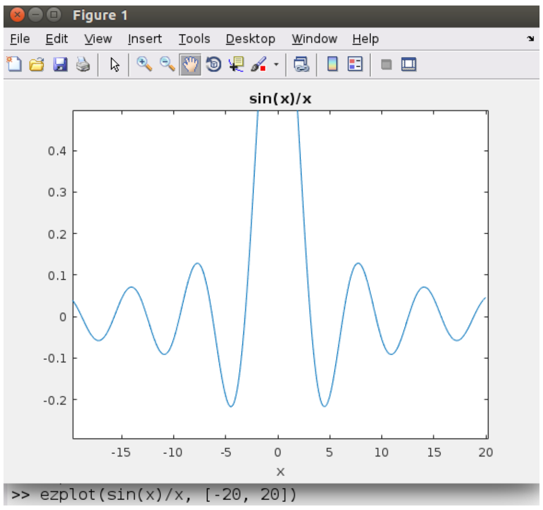
VII. Conclusion

Both MATLAB and Python can perform symbolic calculation that needs in a basic calculus course. Both have its advantages and disadvantages. MATLAB has one integrated development environment to do the calculation, while Python



```
f = n**(1/n)
plot(f, (n, .5, 6))
```

Figure 7. Plot of the $\sqrt[n]{n}$ made by sympy



```
>> ezplot(sin(x)/x, [-20, 20])
```

Figure 8. Plot of the $\frac{\sin x}{x}$ made by MATLAB

has several with different advantages. To choose one of them to teach calculus (or other part of mathematics) is can depend on their needs in their later subjects. For example geographer students can use Python for scripting QGIS or ArcGIS program later and achieve geographical databases, so Python can be a good choice to learn as first language.

REFERENCES

[1] "Official Python site." [Online]. Available: http://www.python.org

[2] M. Summerfield, *Programming in Python 3: A Complete Introduction to the Python Language*. Addison-Wesley Professional, 2010.

[3] V. István, "Learning and teaching combinatorics with sage," *Teaching Mathematics and Computer Science*, vol. 10, no. 2, pp. 389 – 398, 2012.

[4] "Official site of Symbolic Math Toolbox." [Online]. Available: https://ch.mathworks.com/products/symbolic.html