A dream of computing and LaTeXing together: A reality with SageTeX

Manjusha Joshi

Abstract

Researchers search for some computational packages to find their results. At the time when they have the desired output, they begin worrying about how to insert it in their LaTeX document. They have to keep track of their output, formatting and inserting it at the appropriate places in the document.

The SageTeX package is a blessing in these situations. It calls the powerful open source maths server Sage, to compute and embed the result into a TeX document.

1 Sage: Math server

Sage [1] can handle numeric calculations, symbolic calculations, and plotting 2D–3D functions.

Sage is based on the scripting language Python. Sage is free and open source software which is intended to provide an alternative to proprietary programs such as Mathematica, Matlab, Maple and Magma.

1.1 Numerical outputs

Sage can handle multiprecision calculations.

```
factorial(100)
```

$(100)! = 93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864000000000000000000000000$

1.2 Symbolic calculations

Sage can also handle symbolic computations.

$$\int e^x x \, dx$$

can be computed in Sage using this command:

```
integral(exp(x)*x,x)
```

and then the output can be inserted in the TeX document like this:

```
$\sage{integral(exp(x)*x,x)}$
```

and the output will be generated like this: $(x-1)e^x$.

1.3 Graphics

Sage can produce graphics, both 2D and 3D. Fig. 1 shows a twisted torus generated in Sage.

2 Necessity of computed outputs in LaTeX?

Here are a few of the places where it can be useful to dynamically compute values from directly within a LaTeX document:

Figure 1: A twisted torus generated in Sage

- In research papers
- In questions on exams
- Answers in solution sets in a book
- E-learning systems

3 Calculations from mathematical software

Researchers may need varies types of computations. Here is the list of related free software with the main focus on this job:

- R: Statistics, data handling
- Scilab: Numerical Computing
- Maxima: Symbolic Computing, Graph Theory
- GAP: Abstract Algebra
- Pari-gp: Number Theory
- Singular: Commutative Algebra

Sage can integrate any of these packages. The usual way of inserting results in a `.tex` file is by drawing it with some other software and inserting it as a figure.

Here is a figure of a vector field computed and drawn through the software Scilab [2]:

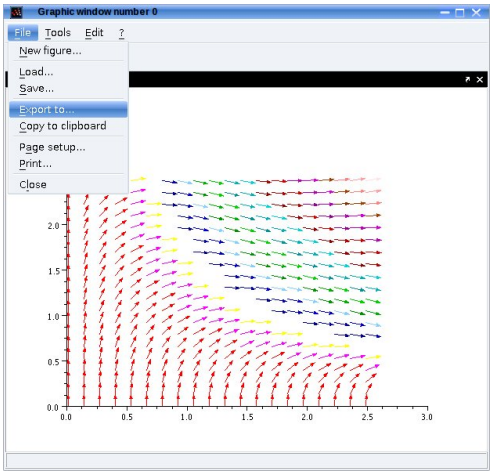


Figure 2: Exporting figure

4 Challenges

- While writing the paper first think of the places where the author needs to insert the computed outputs/figures.

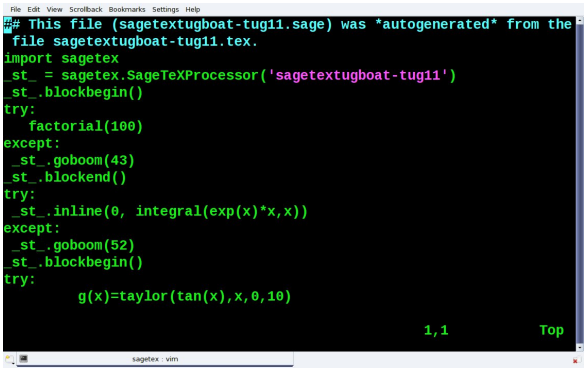


Figure 3: Generated `.sage` file

- In each case one needs to switch between TEX and the software for computing, and search for the place in the TEX file where it has been decided to insert the outputs.
- While pasting calculations from other software, the author may miss some part of the answer. Loss of important calculated data is possible.
- Some files also may be lost, simply because the author may forget where the different output files have been stored, or because the software stores them in an unknown location.
- This laborious process does not help the author trying to compose the paper.

5 With SageTEX

While writing research papers one can insert appropriate examples which can be calculated or drawn immediately by calling Sage directly from the LATEX compiler, and then be inserted at the appropriate places. Thus, one can maintain some orderliness in thinking.

Pictures, calculations become part of the `.tex` file. The author need not given special attention for picture insertion, etc. Also, it is not required to collect such files separately.

6 Work flow

To use Sage through LATEX, just use the package `sagetex` in the `.tex` file. (Of course you need to have Sage installed.) The sequence for compilation is as follows:

1. Start with your file `abc.tex` and `sagetex.sty`.
2. `pdflatex abc.tex` which contains the usual `\usepackage{sagetex}`.
3. This generates a `.sage` file in the same folder.
4. `sage abc.sage` to generate a `.sout` file. An example is shown in fig. 4.
5. `pdflatex abc.tex` once more.

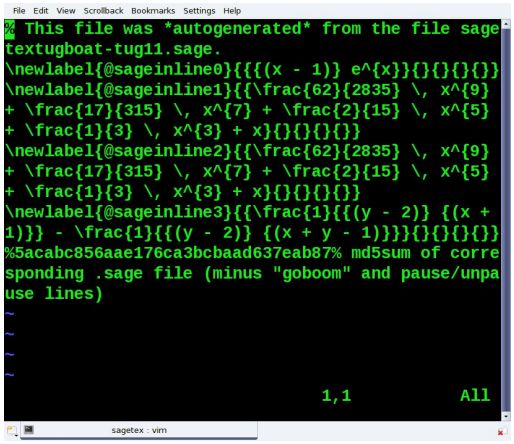


Figure 4: Generated `.sout` file

7 Sage commands

There are a few commands with which the document can communicate with Sage in variety of ways.

To insert only the output from Sage, use the command `\sage{...}`:

```
\sage{factorial(100)}
```

To plot a function through sage, the command is `\sageplot{...}`:

```
\sageplot{plot(exp(x),-5,5)}
```

It is also possible to display Sage commands in the `.tex` file as verbatim text and at the same time pass these commands to Sage. This is done with the `sageblock` environment:

```
\begin{sageblock}
g(x)=taylor(tan(x),x,0,10)
\end{sageblock}
```

After this, `g(x)` is known to Sage with the definition declared in the `sageblock`.

Then the TEX command `\sage{g(x)}` will pass the value of `g(x)` and the computed output will be inserted, as in:

```
$$\tan(x)=\sage{g(x)}$$
```

7.1 Predefined graphs

Sage predefines many graphs commonly used in graph theory. One can have complete graphs on a given number of vertices. Here is the famous Petersen graph (fig. 5):

```
import sets
g=graphs.PetersenGraph()
```

The LATEX command to display such graphs:

```
\sageplot{ g.plot().show() }
```

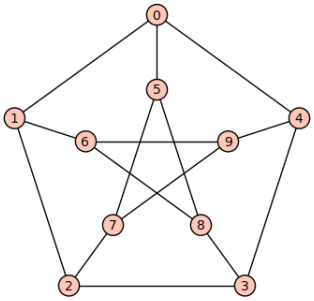


Figure 5: Petersen graph output.

7.2 Dynamic input to Sage

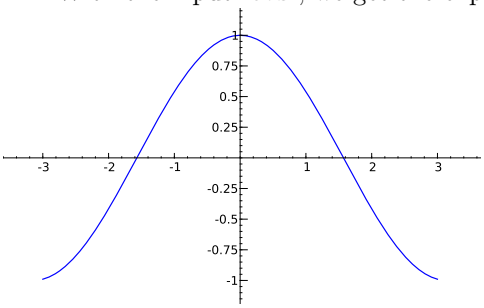
The input to Sage can be determined at the time of LaTeX compilation. For example, one can generate output from Sage after collecting input with `\typein`: the author can insert functions to be plotted at the time of compilation and plot the graphs. With this, at each compilation one can generate different graphs. This can be useful with online exam systems, since we want to generate different figures or randomly generated matrices in the exam paper.

As another example, one can have a collection of examples one of which is inserted randomly; required calculations and figure choice will be inserted on the fly at compilation time.

Here is an example of how one can get input and send it to Sage at the time of compilation:

```
\typein[\function]{Enter function name}
Here is the graph of $\function$:
\sageplot{plot(\function, -3,3)}
```

With the input '`cos`', we get the expected:



8 Pregenerating results from Sage

For submission of a paper or book, it is not necessary for Sage to be installed on the publisher's systems. The author should send the generated `.sout` file along with the `.tex` file for publishing.

9 Installation of sagetex

The package `sagetex` is available on CTAN [3], and also from `http://www.sagemath.org/`. Install it either in a `texmf` tree or in your working document folder.

If Sage is properly installed on the system and available then no additional settings are required. More installation information can be found at [4].

Also, one can use Sage remotely from the Internet while still using the `sagetex` package to compute and insert outputs. More details about this can be found at [5].

References

[1] `http://www.sagemath.org/`

[2] `http://www.scilab.org/`

[3] `http://mirror.ctan.org/macros/latex/contrib/sagetex`

[4] `www.sagemath.org/doc/installation/sagetex.html`

[5] `http://www.sagenb.org/`

◇ Manjusha Joshi
Pune, India
`manjusha dot joshi (at) gmail dot com`