Alkiviadis G. Akritas

УДК 519.67

FREE WORKING ENVIRONMENTS FOR COMPUTER ALGEBRA

Аннотация

В статье представлены два свободно распространяемых пакета с открытым исходным кодом для использования в качестве сред для работы с системами компьютерной алгебры. Они могут использоваться как в среднем, так и высшем образовании, а также учеными из различных областей науки. В статье показано решение двух задач в каждой и систем, на примере которых анализируются преимущества и недостатки представленных систем.

Ключевые слова: системы компьютерной алгебры, CAS, Sage, TeXmacs.

1. INTRODUCTION

This year marks the 15th anniversary of the Journal «Computer Tools in Education». Over these years a plethora of new computer tools has been developed for application in education and research. Included in these tools is a number of free, open source CASs such as Sage, Sympy, Xcas/Giac[1] et al and several Text Editing Platforms with special scientific features such as LyX, TeXmacs et al.

In this article two educational tools are presented as working environments for CASs: Sage, a computer algebra system that can also serve as an editor, and TeXmacs, a scientific text editor that can also serve as an interface for several computer algebra systems *including* Sage.

In the next section Sage is presented using TeXmacs as an interface. The main features of Sage are highlighted with the help of two examples and its advantages / disadvantages are explicitly stated at the end.

In the last section TeXmacs is presented as an interface to other CASs. Its main features are highlighted with the help of the same set of examples used in the previous section, and its advantages / disadvantages are also explicitly stated.

In both sections below, the «Output option» of TeXmacs Show timings has been enabled to compare the execution times of the examples.[2]

Note that this article was entirely written in TeXmacs from which a LaTeX version was exported for submission to the Journal.[3]

[1] Xcas is the interface for the Giac library.

[2] All examples in this paper were run on a Compaq computer with Ubuntu 12.04 (precise) 32-bit and a Genuine Intel CPU 585 at 2.16GHz.

[3] При подготовке к печати статья была свёрстана заново в принятой в журнале системе вёрстки статей.

2. SAGE

Sage was started by William Stein in 2004 but is now developed by a worldwide community of contributors; it is a free, open-source CAS which integrates many specialized mathematics software into a common interface, for which the user needs to know only the programming language Python. However, Sage is not pure Python, since it makes use of many external packages written in various other programming languages (C, C++, Fortran, Lisp, Python, etc); the list of these packages can be seen at http://www.sagemath.org/linkscomponents.html.

Of interest to us is the fact that Sage comes bundled with four well established and time-tested CASs: Gap, Maxima, Pari/Gp and Singular – as well as the Python-based SymPy.[1] This results in two-fold benefits:

– instead of writing functions anew, Sage uses the existing functions of these CASs: Gap for Group Theory, Maxima for Calculus, Pari/Gp for Number Theory, Singular for Commutative Algebra;

– each one of these CASs can be used *on its own*; in this way, the user can verify the results obtained in one system by comparing them against the results obtained using another.

Sage can be downlowded from http://www.sagemath.org/ and is available for Linux, MacOSX, Solaris and Windows; the author found that Debian is the easiest package distribution system to deal with. The user interface can be:

– a Sage notebook in a web browser, in which case Sage connects either locally to one's own Sage installation or to a Sage server on the network,

– a command line in a terminal, and

– TeXmacs, which is not mentioned neither in the Sage main web page nor in the Sage manuals/tutorials.

Inside the Sage notebook or inside TeXmacs one can create embedded graphics, typeset mathematical expressions, etc.

In the sequel, Sage is demonstrated by:

a) factoring the polynomial x^2-9 in it, as well as in all the CASs that come bundled with it, and

b) counting the number of prime numbers ≤ 100. TeXmacs is used as the interface and to start the whole process we initiate an interactive session with Sage with the help of the menu Insert → Session → Sage.[2]

After TeXmacs has been linked to Sage the following system information and `Sage]` prompt appear indicating we are ready to begin; note that the red color of the prompt is the only difference between the interface (original document) and the pdf file produced by LaTeX:

```
| Sage Version 5.5, Release Date: 2012-12-22                    |
| Type "notebook()" for the browser-based notebook interface.   |
| Type "help()" for help.                                       |
```

```
Sage]
```

To factor x^2-9 in Sage itself we first define the polynomial ring $\mathbb{Z}[x]$:

```
Sage] R.<x> = ZZ['x']
                                                       31 msec
```

and then enter the polynomial in the (already defined) variable x.

```
Sage] poly = x^2 - 9
                                                       60 msec
```

[1] At the site https://github.com/sympy/sympy/wiki/SymPy-vs. Sage a comparison can be found between Sage and SymPy.

[2] It is assumed that Sage is installed on the computer and «visible» from a terminal window.

The factors of $x^2 - 9$ are obtained using the function `factor()` of Sage either in the classical way:

```
Sage] factor(poly)
(x – 3) · (x + 3)
                                                689 msec
```

or in Python syntax:

```
Sage] poly.factor()
(x – 3) · (x + 3)
                                                 11 msec
```

Obviously, since `factor()` was «imported» the first time it was used, it runs much faster the second time.

As already mentioned, Sage is not «reinventing the wheel» but, when possible, uses functions from the other free CASs. In this case, the Sage manual informs us that the function **`factor()`** calls the corresponding function of the computer algebra system Pari/Gp.[1] Moreover, since we, the users, also have direct access to all these CASs, we can factor the polynomial $x^2 - 9$ using directly the function `factor()` of Pari/Gp – without first declaring the polynomial ring $\mathbb{Z}[x]$:

```
Sage] pari( 'factor(x^2-9)' )
[x – 3; 1; x + 3; 1]
                                                134 msec
```

or in Python syntax:

```
Sage] f = gp.factor('x^2-9')
                                                520 msec
Sage] f
[x – 3, 1; x + 3, 1]
                                                 39 msec
```

In the above results, along with each factor we obtain its multiplicity, 1.

In the same spirit we can factor the polynomial $x^2 - 9$ using the function `factor()` of Maxima:

```
Sage] maxima( 'factor(x^2-9)' )
(x – 3) (x + 3)
                                              2.726 sec
```

or in Python syntax:

```
Sage] maxima.factor('x^2-9' )
(x – 3) (x + 3)
                                                 68 msec
```

To factor $x^2 - 9$ in Singular we again first define the polynomial ring:

```
Sage] R = singular.ring(0, 'x' )
                                                749 msec
Sage] R
    // characteristic : 0
    // number of vars : 1
    // block 1        : ordering lp
    //                : names x
    // block 2        : ordering C
                                                174 msec
```

[1] In http://pari.math.u-bordeaux.fr we see that «Pari is a C library (C++ compatible), whereas Gp is an interactive shell giving access to Pari functions, easier to use. Low-level scripts are typically 4 times slower than directly written C code».

and then enter the polynomial in the (already defined) variable x.

```
Sage] polyS = singular('x^2 - 9')
                                                           3 msec
```

The factors of $x^2 - 9$ are obtained using the function `factorize()` of Singular either through the library:

```
Sage] singular( "factorize(x^2 - 9)" )
  [1]:
    _[1]=1
    _[2]=x+3
    _[3]=x-3
  [2]:
    1,1,1
                                                          13 msec
```

or in Python syntax:

```
Sage] polyS.factorize()
  [1]:
    _[1]=1
    _[2]=x-3
    _[3]=x+3
  [2]:
    1,1,1
                                                          12 msec
```

Here again, the factors come with their multiplicities.

To factor the polynomial $x^2 - 9$ in SymPy we have to import the function factor and the variable x:

```
Sage] from sympy import factor
                                                       3.394 sec

Sage] from sympy.abc import x
                                                          63 msec

Sage] factor(x^2-9)
(x – 3)*(x + 3)
                                                          37 msec
```

or in Python syntax:

```
Sage] (x^2-9).factor()
(x – 3)*(x + 3)
                                                          63 msec
```

Finally to factor the polynomial $x^2 - 9$ in the computer algebra system Gap we have to first define the polynomial ring (along with the variable) and then to take into consideration its special interface with Sage as described in http://www.sagemath.org/doc/reference/sage/interfaces/gap.html:

```
Sage] x = gap('Indeterminate(Integers,"x")')
                                                       1.074 sec

Sage] poly = gap.new(x^2-9)
                                                          76 msec

Sage] poly.Factors()
[x – 3; x + 3]
                                                         257 msec
```

or

```
Sage] gap.Factors(poly)
[x – 3; x + 3]
                                                          16 msec
```

Next, we address the second problem and write a small Sage program (actually a Sage for loop) to count the number of prime numbers ≤ 100. Sage, of course, does have the functions `prime_pi()` and `is_prime()` that could be used, but we avoid them in order to demonstrate both the advantages and disadvantages of Sage. Instead, we use the function `IsPrime()` of Gap inside the Sage for loop.

To know what to expect, though, we first compute the answer directly using the function `primepi()` of Pari/Gp:

```
Sage] pari( 'primepi(100)' )
25
                                                          11 msec
```

or in Python syntax:

```
Sage] gp.primepi(100)
25
                                                          13 msec
```

So, there are 25 prime numbers ≤ 100. And below is the same answer using the Sage for loop with the computer algebra system Gap:

```
Sage] counter = gap(0)
                                                          1.6 sec

Sage] for i in range(100):
        if gap(i+1).IsPrime():
          counter = counter + 1
                                                         671 msec
Sage] counter
25
                                                         463 msec
```

Note that although Maxima does have the function `primep()`, to test if an integer is a prime number, the system cannot be used in the for loop above because in http://www.sagemath.org/doc/reference/sage/interfaces/maxima.html it is stated that «Maxima does not seem to have a non-interactive mode which is needed for Sage». So, the viable candidates for use inside the Sage for loop are Gap and Pari/Gp, where in the latter `isprime()` tests an integer for primality.

From the above examples one can easily draw the following conclusions regarding the advantages and disadvantages of Sage:

Advantages: The main advantage of Sage is the ability to use the various CASs packaged with it by learning just one general purpose language, Python! Moreover, Sage can connect to optional CASs, like Magma, Maple and Mathematica (all commercial), and Giac (free).

Also, another advantage is the ease with which one can get the archive of files (tarball) and compile all the packages simultaneously, as opposed to finding and compiling packages individually.

Disadvantages: Obviously it is quite impressive to be able to call different packages in the same program; by integrating these packages into a common (Python) interface Sage gets a lot of functionality without having to «reinvent the wheel». However, this very interface can also be a bane – as in the case of Maxima not been able to run the Sage for loop above. Moreover, due to this interface at times it is impossible to take advantage of the very efficient and fast routines that exist in some libraries. To wit:

Consider the (C++) library LinBox, which is one of the packages loaded with Sage and where there is an efficient implementation of large integer determinants that can be called from Sage. However, if Sage needs to call Maxima (to compute, say, the antiderivative of a function), and during this computation Maxima needs to compute a large determinant, then Maxima will never be able to call that efficient implementation in LinBox![1]

Finally, regarding Sage installations on Windows, it is worth mentioning that there is no Windows port; instead, Sage provides a VirtualBox that you can run from Windows – an arrangement that requires a lot of memory. This is no problem if you have a powerful PC, however, Sage will probably not be able to run on most Windows netbooks or older PCs | since a precompiled version of Sage works on a linux netbook.

3. TEXMACS

TeXmacs is a free wysiwyw (what you see is what you want) editing platform with special features for scientists. It was designed and written by Joris van der Hoeven, who was inspired both by the TeX system, written by D. Knuth, and by Emacs, written by R. Stallman; hence the name TeXmacs. Today it is supported by a worldwide team of contributors and is available for Knoppix, Linux, MacOSX and Windows; for more details see http://www.texmacs.org.

The software aims to provide a unified and user friendly framework for editing structured documents with different types of content (text, graphics, mathematics, interactive content, etc.). The rendering engine uses high-quality typesetting algorithms and produces professionally looking documents.

Included in the software is a a text editor with support for mathematical formulas, a small technical picture editor and a tool for making presentations from a laptop. New features can be added to the editor using the Scheme extension language.

Of interest to us is the fact that TeXmacs can be used as an interface for – among others – several external CASs like Maxima, Pari/Gp, Sage, Xcas/Giac et al. The complete list of packages that can be plugged into TeXmacs can be seen in http://www.texmacs.org.

In this section we use TeXmacs as an interface to the computer algebra systems Maxima, Pari/Gp, Sage and Xcas/Giac,[2] and we solve both problems of the previous section; that is, we: (a) factor the polynomial $x^2 - 9$ in each system, and (b) count the number of primes ≤ 100, by writing a for loop in the language of each system. This way we are able to draw our own conclusions regarding TeXmacs' advantages and disadvantages.

We begin by initiating a session with Maxima. To save space, the system information that appears after linking TeXmacs with Maxima is not included; and the same is true for subsequent CASs in this section.

The factorization is easily performed:

```
(%i1) factor(x^2-9)
(%o1) (x – 3)(x + 3)
                                                  54 msec
```

and so is the counting of the prime numbers ≤ 100 with the following for loop:

```
(%i2) counter : 0
(%o2) 0
                                                  18 msec
(%i3) for i:1 thru 100 do
        if primep(i) then counter : counter + 1
(%o3) done
                                                  76 msec
```

[1] For that, it is required to glue the C libraries at the C level, not at the Python level, or everything must be implemented at the Python level – something Sympy is trying to do. But even then, Python was not designed to be an efficient language for computer algebra systems.

[2] All assumed installed on the computer and «visible» from a terminal window.

```
(%i4) counter
(%o4) 25
                                                               17 msec
```

Obviously, when Maxima is called directly by TeXmacs, and not through Sage, the for loop is executed without any problems and we obtain the correct answer.

Next we initiate a session with Pari. Again, the factorization is easily done:

```
Pari] factor(x^2-9)
```

$$\%1 = \begin{pmatrix} x-3 & 1 \\ x+3 & 1 \end{pmatrix}$$

```
                                                               80 msec
```

and so is the for loop to count the prime numbers ≤ 100:

```
Pari] counter = 0
%2 = 0
                                                               12 msec
Pari] for(i=1,100,
          if( isprime(i), counter = counter + 1 )
        )
                                                               54 msec
Pari] counter
%3 = 25
                                                               11 msec
```

Out of curiosity, we initiate a session with Sage and, through it, we call Pari/Gp to perform the same for loop and compare times:

```
Sage] counter = pari(0)
                                                               73 msec
Sage] for i in range(100):
         if pari(i+1).isprime():
      counter = counter + 1
                                                              356 msec
Sage] counter
25
                                                               74 msec
```

Clearly, Sage's interface slows things down quite a bit.

Finally, we initiate a session with Giac. Here again the factorization is as was in the previous CASs:

```
> factor(x^2-9)
```

$$(x-3)(x+3)$$

```
                                                               92 msec
```

and the prime numbers ≤ 100 are counted in about the same way using the function isprime():

```
> counter:=0
                                  0
                                                               58 msec
> for (j:=1;j<=100;j++) {
    if (isprime(j)) {counter := counter + 1}
  }
                                  0
                                                               83 msec
> counter
                                  25
                                                               83 msec
```

Xcas/Giac – also known as the «Swiss knife for Mathematics» – is a powerful CAS (combined with programming, interactive 2-d and 3-d geometry and a spreadsheet) that is under active development by Bernard Parisse and supported by a worldwide community of contributors. Besides its ability to link to TeXmacs, Xcas/Giac can also link to the computer algebra systems CoCoA and Pari as well as to several other free, open source libraries, utilizing all the efficient and fast routines that are available.[1]

Giac is linked with Pari through the command pari(), which makes available to Giac all the functions of the Pari library – provided they are prefixed by pari . So we can have:

```
> pari()
                                                                55 msec
> pari_factor(x^2-9)
```

$$\begin{pmatrix} x-3 & 1 \\ x+3 & 1 \end{pmatrix}$$

```
                                                                85 msec
```

From the above examples we can easily draw the following conclusions regarding the advantages and disadvantages of TeXmacs:

Advantages: A first advantage of TeXmacs is that it offers a wider choice of CASs to use than Sage; namely, it offers most of the packages that come bundled in Sage, Sage itself plus additional ones – listed in http://www.texmacs.org – that are not bundled in Sage. The fact that these CASs have to be installed separately is not a problem at all for package distribution systems like Debian/Ubuntu.

Secondly, the interface of TeXmacs with each individual CAS is direct and not prone to problems that arise using Sage's interface – as for example Sage's interface with Maxima that we saw in the previous section.

Finally, if the user already has experience with a certain CAS, then s/he can go on and use it right away with TeXmacs as an interface, without having to worry about learning Python first.

Disadvantages: The only disadvantage that one might attribute to TeXmacs is the fact that one cannot use it as an editor for the Greek language; however, Greek letters are available in math mode. Hopefully, this situation will be remedied in the future.

4. CONCLUSIONS

From the previous discussion it is obvious that – for the packages discussed in this paper – we have the following «dependencies»:

$$\texttt{TeXmacs} \xrightarrow[\textit{Plugin}]{} \begin{cases} \texttt{Giac} \\ \texttt{Maxima} \\ \texttt{Pari} \\ \texttt{Sage} \xrightarrow[\textit{Python}]{} \begin{cases} \texttt{Gap} \\ \texttt{Maxima} \\ \texttt{Pari/Gp} \\ \texttt{Singular} \\ \texttt{Sympy} \\ \vdots \end{cases} \\ \vdots \end{cases}$$

Therefore, if your favorite CAS has a TeXmacs plug-in and you need a powerful editor, then TeXmacs is the interface to use. It is worth noting that after TeXmacs' session with the corresponding CAS has been initiated, along with the system information there appear instructions on how to

[1] That is, unlike Sage, Giac glues the C libraries at the C level.

open the CAS's manuals and tutorials in case you need them. The only exception is Maxima where only the system information appears; however, simultaneously, a big question mark appears – without any warning | at the right end of the menu bar and it is by pressing this question mark that the user's manual opens up.[1]

The choice of the CAS itself depends on the user's needs. The argument in favor of Sage is that by learning one programming language, Python, the user can utilize several CASs, whereas to program in the other CASs the user has to learn one programming language for each one of them. However, as we saw in the examples of the previous section, the syntaxes of all these languages are related, and it is not hard to switch from one to the other. Moreover, like with natural languages, when circumstances demand it one easily becomes a «polyglot»!

[1] For more details see «A tutorial on the Maxima plug-in» by Andrey Grozin, which can be found in http://www.texmacs.org; strongly recommended.

Abstract

Two free, open-source packages are presented for use as working environments for Computer Algebra Systems (CASs). They can be used in Secondary and Higher Education as well as by scientists in various fields. In this paper a set of two problems is solved in both systems revealing their advantages and disadvantages.

Keywords: Computer Algebra Systems, CAS, Sage, TeXmacs.

Alkiviadis G. Akritas,
University of Thessaly
Dept. of Communication and Computer
Engineering, Volos Greece,
ag.akritas@gmail.com