Sage: Math in Your Dorm Room, from Calculus to Research

(Presented May 28th, 2009 ACMS Conf.)

Karl-Dieter Crisman

Gordon College

Note for proceedings readers: all commands were evaluated live in a Sage notebook. They should be attempted by the reader by pasting the commands following the `sage:` *prompts into a cell of a Sage notebook session, then either typing Shift-Enter or clicking "Evaluate" under the cell. Or, one can "Edit a Copy" on the following copy of the original talk in notebook format:* `http://www.sagenb.org/home/pub/188/`

This talk introduces Sage, a very useful free tool for the undergraduate curriculum (and beyond). To begin, let's take a look at some calculations one might want to do in an undergraduate course at some point, but not necessarily want them to do by hand:

```
sage: factorial(15)
```
1307674368000

```
integral(3*x^2/sqrt(25*x^2-3),x,1,2)
```
$-\frac{3}{50}\sqrt{22}+\frac{3}{25}\sqrt{97}+\frac{9}{250}\ln\left(10\sqrt{97}+100\right)-\frac{9}{250}\ln\left(10\sqrt{22}+50\right)$

These certainly come up in combinatorics, arc length-type computations, Taylor series, or wherever. It's also pretty clear that if you needed one of these as an intermediate result, you wouldn't mind the assistance of computer technology to get it. The same would be true of a numerical approximation.

```
sage: numerical_integral(3*x^2/sqrt(25*x^2-3),1,2)
```
$(0.9262503194124578 3, 1.0283444311781162e-14)$

This even gives the (thirteen place) accuracy involved! Of course, there are many other areas where this could be useful. In linear algebra, for instance, you might want to make a matrix and a vector:

```
sage: A = Matrix([[1,2,3],[3,2,1],[1,0,1]])
sage: w = vector([1,1,-4])
sage: A;w
```
$$\begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$
$(1, 1, -4)$

Actually, you might want to use them to solve a linear system, like the following:

$$\begin{aligned} x + 2y + 3z &= 1 \\ 3x + 2y + z &= 1 \\ x + z &= -4 \end{aligned}$$

```
sage: A \ w
```
$\left(-2, \frac{9}{2}, -2\right)$

It's good to know the answer. Actually, I can check that this works, too:

```
sage: A*vector([-2,9/2,-2])
```
$(1, 1, -4)$

Or I can get myself a list of more complicated things I can do. For instance, what else can I do with the matrix A? I can use the 'tab' key to find out what eigen-objects there are:

```
sage: A.eigen[tab]
 A.eigenmatrix_left    A.eigenspaces_right
 A.eigenmatrix_right   A.eigenvalues
 A.eigenspaces         A.eigenvectors_left
 A.eigenspaces_left    A.eigenvectors_right
```

I can even ask for help if I'm confused on the definition or functionality of something.

```
sage: A.eigenspaces_left?

File: /Users/.../sage/matrix/matrix2.pyx
Type: <type 'builtin_function_or_method'>
Definition: A.eigenspaces_left(var='a', algebraic_multiplicity='False')
Docstring:
   Compute left eigenspaces of a matrix.

   If algebraic_multiplicity=False, return a list of pairs (e, V)
   where e runs through all eigenvalues (up to Galois conjugation) of
   this matrix, and V is the corresponding left eigenspace.
   If algebraic_multiplicity=True, ...
```

Oops, not what I was looking for! Here is what I wanted:

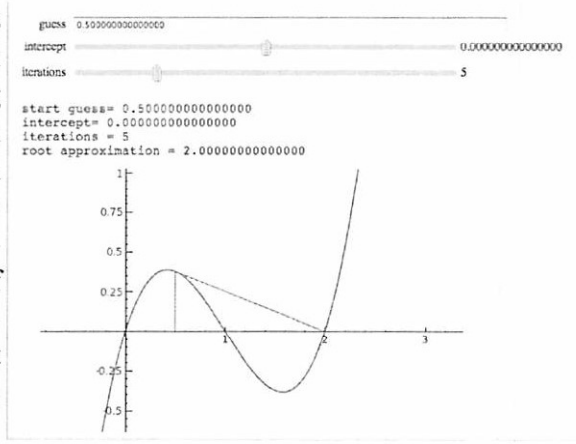
```
sage: A.eigenvalues()
```
$[-1.603875471609677?, 1.109916264174743?, 4.493959207434934?]$

Now, many of us are familiar with doing examples like this in class. But ideally, we should expect students to use the same tools that we do, even if in a more basic manner - perhaps in a lab or as part of a homework assignment. What are some obstacles to this?

- There are many wonderful (free!) applets and demonstrations available for this (e.g. for fractals, modular arithmetic):
  - But, these usually handle one specific type of exercise only.
- There are powerful and impressive comprehensive software packages students can buy:
  - These may be expensive, at least for students who may only use them in one course.
- There are computer labs where students can use such software, if your college can afford the site license:
  - But if the hours are inconvenient or the lab is distant, they may never go there.

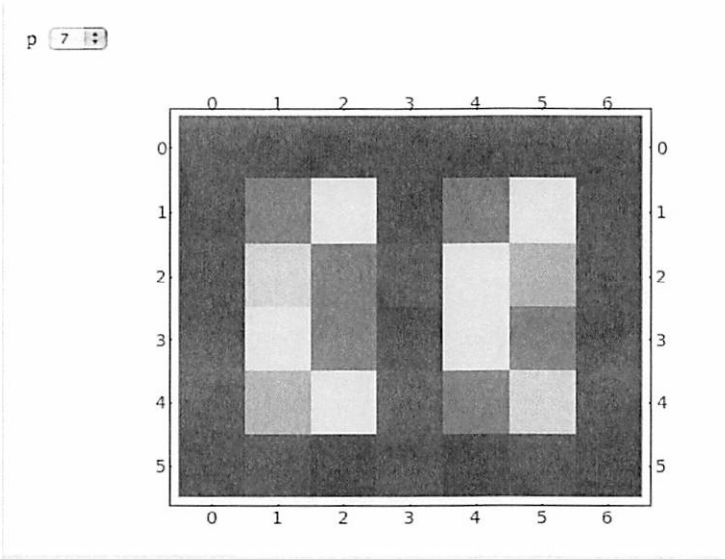
On the other hand, Sage, the software I am currently using in this talk, is a free, powerful option to consider for such things. It can be either downloaded by the student OR run off a server on campus. In either case, your web browser is your interface, just as I am doing now; it's math in your dorm room!

Sage can do more than classroom examples or routine computations. Next is an example from a lab I recently used in Calculus I. After introducing Newton's method and going over some homework on this topic, they were to examine basins of attraction (or whatever else they found interesting) of Newton's method for certain cubic polynomials as a continuation of a semester-long introduction to the ideas of chaos. One can move the sliders to change the number of iterations of Newton's method or the intercept of the cubic, and type in different starting guesses to a root of the cubic.



In this case I (not the students) had to do a little bit of programming, true; however, this mostly consisted of taking an example from the Sage Wiki and modifying it for my own needs. Here is another example, this time from an upper-level number theory course.

```
@interact
def power_table_plot(p=(7,prime_range(50))):
    P=matrix_plot(matrix(p-1,[mod(a,p)^b for a in range(1,p) \
        for b in srange(p)]),cmap='jet')
    show(P)
```



The second column gives the colors for all integers modulo p, and ensuing columns give their powers. How many theorems can you observe in this data? (You should be able to find Fermat's Little Theorem, some quadratic residue facts, perhaps even the existence of primitive roots for primes...)

A little more about Sage:

- Sage is a very actively developed open source project, with about the same functionality as well-known proprietary programs (in some cases more).
- It uses the power of many high-quality packages, such as GAP for groups, Maxima for symbolic calculus, GMP for arbitrary precision integers... yet includes hundreds of thousands of lines of new code, particularly in number theory, exact linear algebra, etc.
- It fully integrates (and is largely written in) Python, has a nice command-line interface... NONE of which you need to know about to use it, of course!
- It's easy to annotate lecture notes, such as the ones I wrote for number theory this past semester - not just with typical WYSIWYG things like itemized lists and text, but also LaTeX-style notation, e.g.

$$\sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_{p} \frac{1}{1-p^{-s}}.$$

- It's easy to share files, publish to the web, collaborate with other users. At Gordon, we have a dedicated server for use by just our students.
- You can try it out online now at `http://www.sagenb.org`!

It is quite usable for undergraduate research as well; in fact, it is heavily funded for research purposes by NSF, Microsoft Research, Sun, and other agencies and companies. A student of mine used it this past summer to formulate several conjectures, and to disprove another one of the originator of the area of research!

There is a lot more to discuss which time prevents discussing. For instance, there is much more functionality, such as interactive 3D plotting and image processing. Another topic would be the appropriateness of using open source software at a Christian college, not simply in terms of stewardship, but also in terms of service. But for now, just visit `http://www.sagemath.org` for more information!