A Short Introduction to Sage

William Stein

January 13, 2011

A Short Introduction to Sage

The Sage Project

Mission Statement

Create a viable free open source alternative to Magma, Maple, Mathematica, and Matlab

A "viable alternative" will have...

- The *mathematical features* of Magma, Maple, Mathematica, and Matlab with *comparable speed*.
- Beautiful interactive 2d and 3d graphics.
- A notebook interface and an IDE.
- Many books (full undergraduate curriculum)
- A web application interface.

Sage ain't Octave (=open source MATLAB clone)

Sage need *not* run programs written in the custom math-only languages of Magma, Mathematica, etc.

2005: I started Sage

SAGE

Software for Arithmetic Geometry Experimentation

- I needed an *open source* alternative to Magma. David Joyner (coding theorist) had similar concerns.
- SAGE in 2005 – number theory (PARI) and coding theory (GAP) – no symbolic calculus or numerical computation.

Number theory & Coding theory: started out very technical

```
sage: E = EllipticCurve('389a'); E
Elliptic Curve defined by y^2 + y = x^3 + x^2 - 2*x
sage: E.gens()
[(-1 : 1 : 1), (0 : -1 : 1)]
sage: G = matrix(GF(5), 4, 7, [1,1,1,0,0,0,0,1,0,0,1,1,...
sage: C = LinearCode(G); C
Linear code of length 7, dimension 4 over Finite Field ...
sage: C.minimum_distance()
3
```

Why not Magma?

1. **Commercial:** Expensive for my collaborators and students ("third world discount" = 3 months salary)
2. **Closed:** Implementation of algorithms often secret
3. **Frustrating:** Too tight control of development
4. **Static:** Users can't define their own classes (data types)
5. **Copy protection:** A pain in the arse
6. **Language:** No `eval`, no exception handling, no namespaces, little development of math-only language
7. **Developer community:** too small, no public mailing list
8. **Graphics:** No graphics, symbolic calculus, or GUI
9. **Bugs:** No public bug tracker or list of reported bugs
10. **Compiler:** No compiler (nothing like Cython)

(Related remarks for Maple, Mathematica, and MATLAB.)

What is Sage?

1. **A self-contained distribution** of over 90 open source packages that is easy to build from source.
2. **Interfaces** that smoothly tie together all these libraries and packages.
3. **A new library** that implements novel algorithms. About a half million lines of code written by a worldwide community of over 200 people over the last 5 years. `http://sagemath.org/development-map.html`

Demo

A Demo...

Use Sage From the Command Line

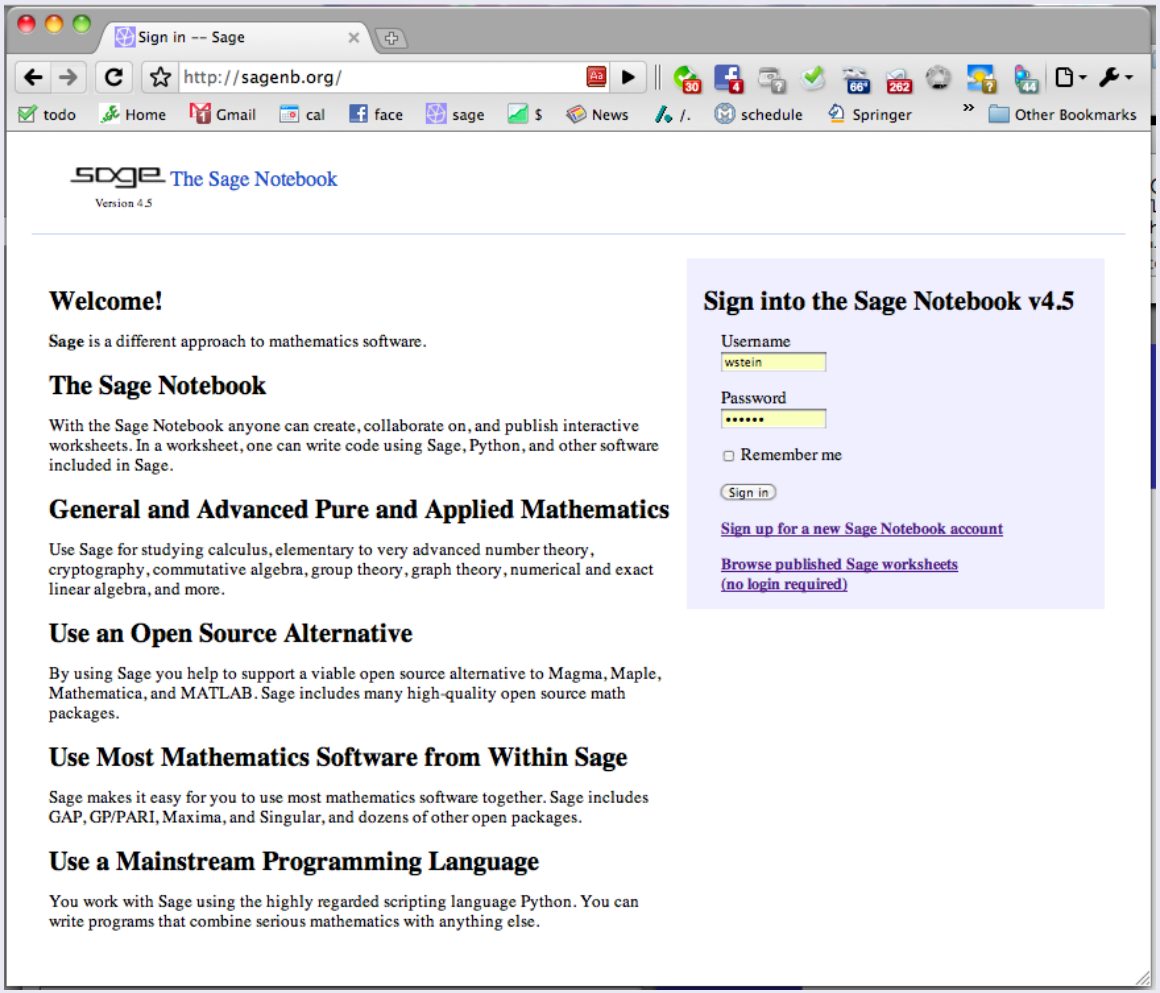
Command Line Sage

Terminal — ssh — 72×18

```
----------------------------------------------------------------------
| Sage Version 4.5.1, Release Date: 2010-07-19                       |
| Type notebook() for the GUI, and license() for information.        |
----------------------------------------------------------------------
sage: factor(2010)
2 * 3 * 5 * 67
sage: f = 1/sqrt(x^2 + 2*x - 1); f
1/sqrt(x^2 + 2*x - 1)
sage: f^2
1/(x^2 + 2*x - 1)
sage: f.integrate(x)
log(2*x + 2*sqrt(x^2 + 2*x - 1) + 2)
sage:
```

Use Sage Via the Notebook

Web Browser Based Interface to Sage



Demo: Factoring

Factoring an integer:

```
factor(2012)
```

$$2^2 \cdot 503$$

Factoring a symbolic expression:

```
x,y=var('x,y'); factor(x^3 - sin(y)^3)
```

$$(x - \sin(y))\left(x^2 + x\sin(y) + \sin(y)^2\right)$$

Factoring a polynomial over a nontrivial finite field:

```
F.<alpha> = GF(49); x = polygen(F)
factor(x^4 + x^3 - 2)
```

$$(x + \alpha + 1) \cdot (x + 6\alpha + 2) \cdot (x + 6)^2$$

Demo: Solving Equations

Solve a quadratic equation:

```
x = var('x'); solve(x^2 + 7*x == 5, x)[0]
```

$$x = -\frac{1}{2}\sqrt{69} - \frac{7}{2}$$

Solve a system of two linear equations with one unknown coefficient α:

```
var('alpha, y')
solve([3*x + 7*y == 2, alpha*x + 3*y == 8],
x,y)
```

$$\left[\left[x = \frac{50}{7\,\alpha - 9}, y = \frac{2\,(\alpha - 12)}{7\,\alpha - 9}\right]\right]$$

Demo: Computing Symbolic Integrals


```
f = 1/sqrt(x^2 + 2*x - 1); f.integrate(x)
```

$$\log\left(2\,x + 2\,\sqrt{x^2 + 2\,x - 1} + 2\right)$$

```
g = integrate(sin(x)*tan(x), x); g
```

$$-\tfrac{1}{2}\,\log\left(\sin\left(x\right) - 1\right) + \tfrac{1}{2}\,\log\left(\sin\left(x\right) + 1\right) - \sin\left(x\right)$$

```
h = g.diff(x); h
```

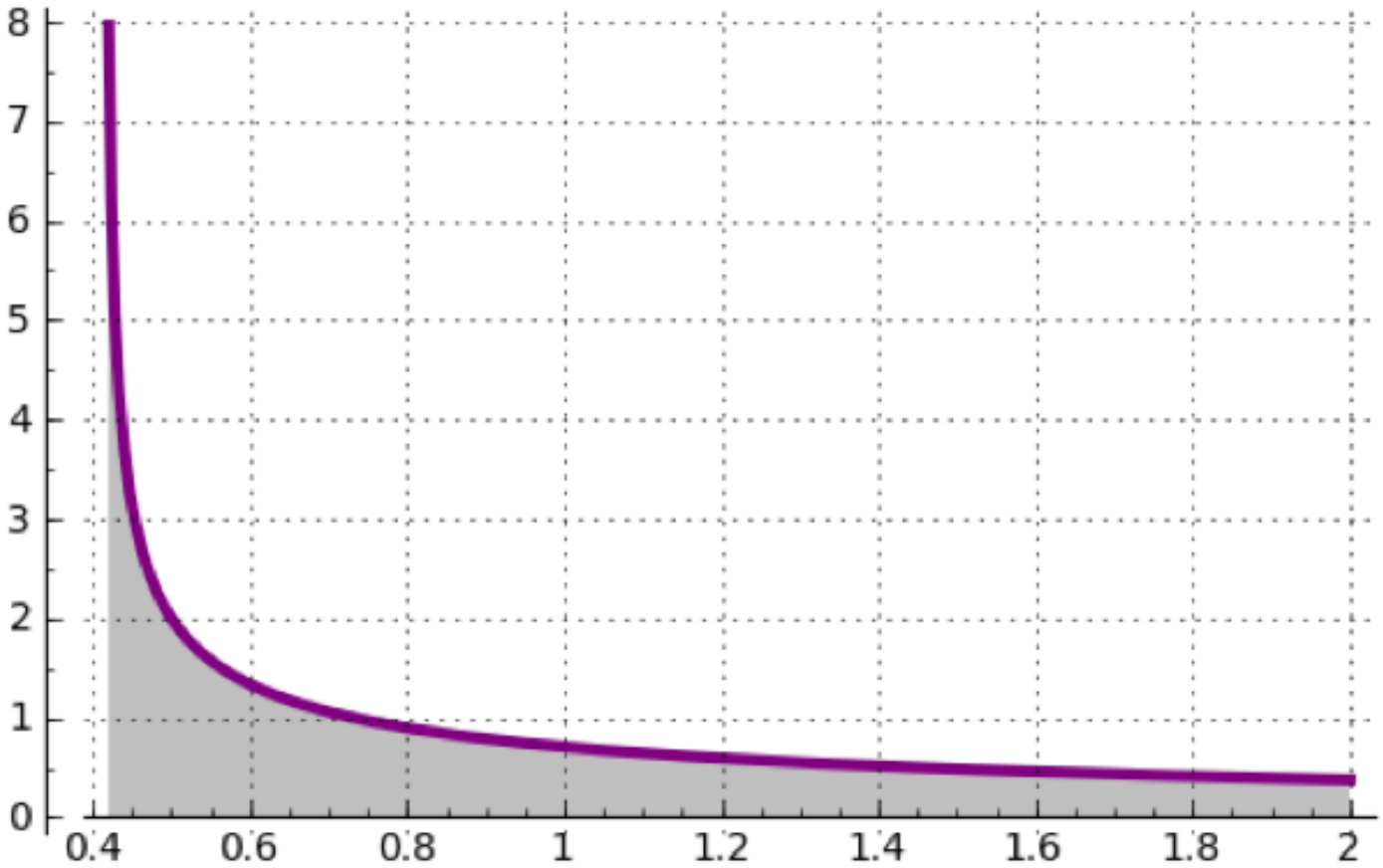
$$\frac{-\cos(x)}{2\,(\sin(x)-1)} + \frac{\cos(x)}{2\,(\sin(x)+1)} - \cos\left(x\right)$$

```
(h - sin(x)*tan(x)).simplify_full()
```

$$0$$

Demo: Plotting a 2D Function


```
plot(1/sqrt(x^2 + 2*x - 1), (x,.4,2), thickness=3,
                color='purple', fill=True, gridlines=True)
```



Demo: Plotting a 3D Function

Introduction to Sage

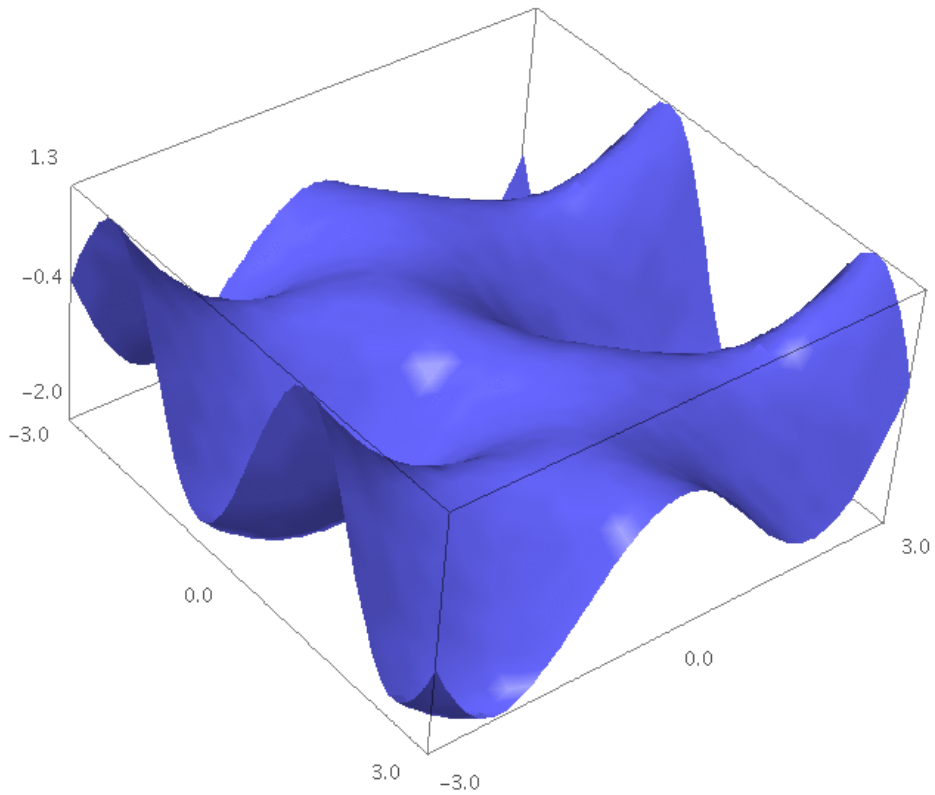
William Stein

History and Goals

A Demo

```
var('x,y');plot3d(sin(x-y)*y*cos(x),(x,-3,3),(y,-3,3))
```

(x, y)

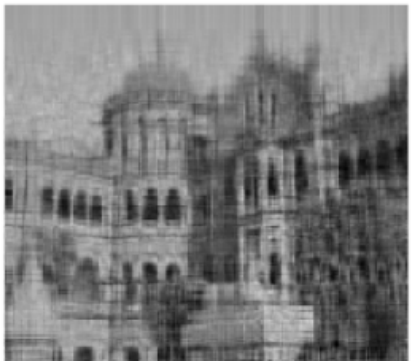


Demo: Interactive image compression


```
import pylab; import numpy
A_image = numpy.mean(pylab.imread(DATA + 'mumbai.png'), 2)
u,s,v = numpy.linalg.svd(A_image)
S = numpy.zeros( A_image.shape )
S[:len(s),:len(s)] = numpy.diag(s)
n = A_image.shape[0]
@interact
def svd_image(i = ("Eigenvalues (quality)",(20,(1..A_image.shape[0])))):
    A_approx = numpy.dot(numpy.dot(u[:,:i], S[:i,:i]), v[:i,:])
    g = graphics_array([matrix_plot(A_approx), matrix_plot(A_image)])
    show(g, axes=False, figsize=(8,3))
    html("Compressed to %.1f%% of size using %s eigenvalues."%(
                    100*(2.0*i*n+i)/(n*n), i))
```

Eigenvalues (quality) 20

```
Compressed to 12.5% of size using 20 eigenvalues.
```



Website Tour

Quick Tour of Website

Questions

Questions?