Introduction to Sage: History, Goals, and a Demo

William Stein

August 2010

Sage: History and Goals

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)
- Commercial support (e.g., customized notebook servers)

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: SAGE started 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)

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 about 200 people over the last 5 years.

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

Example: The Development of Symbolic Calculus in Sage

To give a sense of how Sage is developed, I'll give a very brief glimpse into the development of Symbolic Calculus in Sage.

2006: Maxima/Sage interface

- I taught Calculus (in San Diego) in 2006, and used Maxima + GNUplot for demos and plots.
- I integrated Maxima into Sage.
- David Joyner wrote *Sage Constructions* – included examples of Calculus using the Sage/Maxima interface.

Using the Maxima Interface

```
sage: f = maxima('1/sqrt(x^2+2*x-1)')
sage: f.integrate(x)
log(2*sqrt(x^2+2*x-1)+2*x+2)
sage: maxima.plot2d('cos(2*x) + 2*exp(-x)','[x,0,1]',
                    '[plot_format,openmath]')
[[a plot should appear (just crashes for me!)]]
```

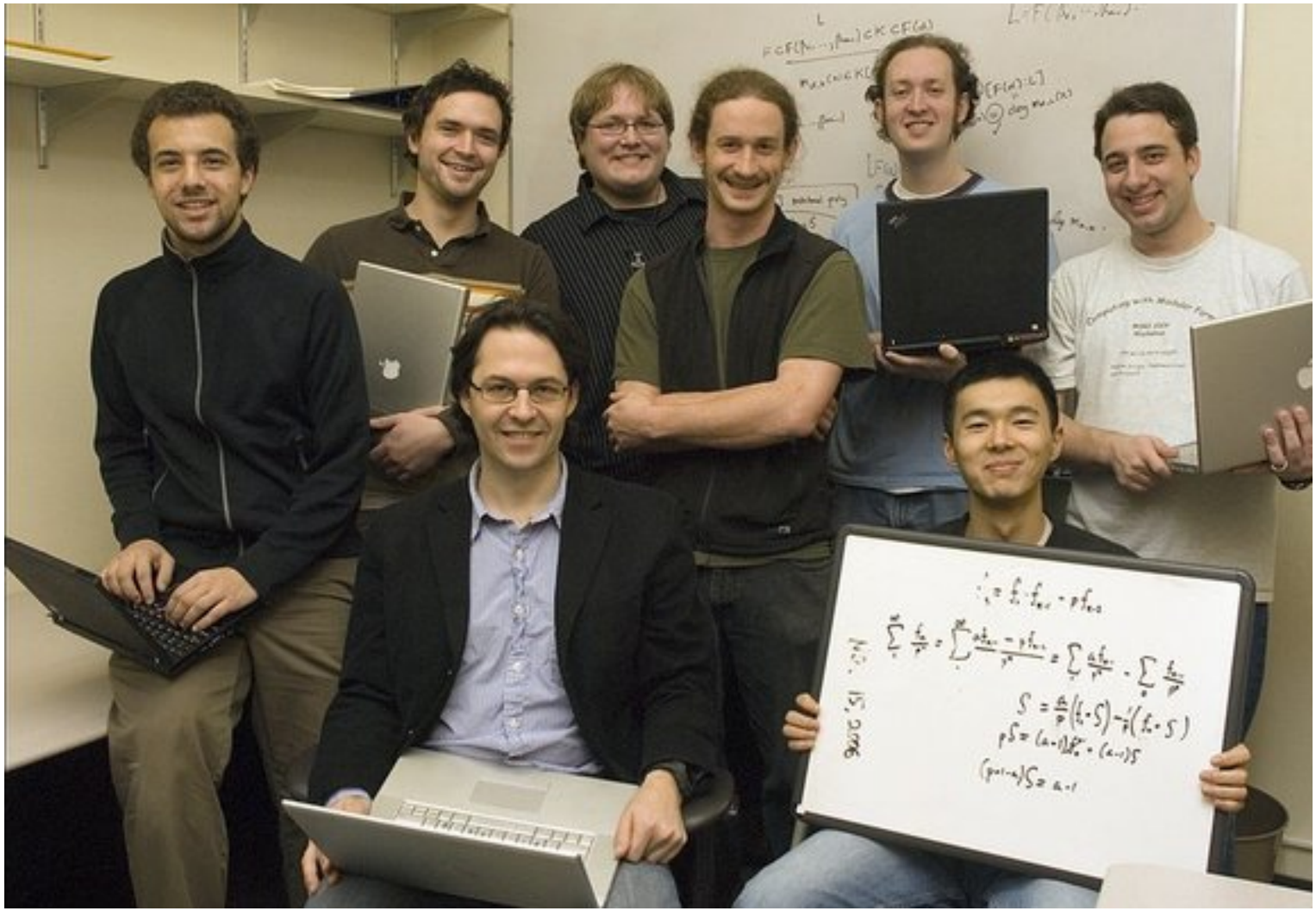
2007: Bobby Moretti

- 2007: Bobby Moretti (UW undergraduate): wrote a pure Python + Maxima symbolic Calculus package.
- Idea and implementation was Bobby's from start to finish
- Made Sage an option for Calculus classes, but too slow compared to Ma* and not flexible enough.

First steps toward symbolic calculus in Sage:

```
sage: x = var('x')
sage: f = 1/sqrt(x^2 + 2*x - 1); f
1/sqrt(x^2 + 2*x - 1)
sage: f.integrate(x)
log(2*x + 2*sqrt(x^2 + 2*x - 1) + 2)
```

Picture: Students working on Sage in 2007



B. Moretti, R. Miller, W. Stein, P. Clark, T. Boothby, J. Kantor, Y. Qiang, R. Bradshaw

2008–2009: Development snapshot

- July 2008: ISSAC – Burcin suggested investigating integrating the C++ library **GiNAC** into Sage.
- Big problem: GiNAC depends on the huge C++ CLN library (field arithmetic), which takes a long time to build, and provides nothing that isn't already in Sage.
- Aug 2008: I wrote the first version of Pynac during an intense two week coding sprint in San Diego.

Key Idea

Rip out the CLN dependence of GiNAC and replace it by Sage/Python. Amazingly, this worked.

- Pynac is a low level C++ library that depends on Sage/Python and provides fast symbolic arithmetic. No integration or other high level algorithms.

Aug 2008 – May 2009: Build complete new symbolic system in Sage on Pynac

- Finish Pynac and build a new fast symbolic system on top: Burcin (mostly) + Mike Hansen + Carl Witty + Robert Bradshaw + Nick Alexander + me + $\cdots$
- Main new feature of Sage-4.0 (Summer 2009).
- Maxima is still used: symbolic integration, some comparisons, system solving, etc.

Example: Pynac-based Symbolics

```
sage: f = 1/sqrt(x^2 + 2*x - 1); 1/(f*f)
x^2 + 2*x - 1
sage: timeit('1/(f*f)')
625 loops, best of 3: 146 microseconds per loop
sage: g = maxima(f)
sage: timeit('g*g')
125 loops, best of 3: 4.29 milliseconds per loop
sage: 4.29/.146
29.3835616438356
```

May 2009 - today: Further development

- May 2009 - today: Burcin (mostly) has further developed symbolics in Sage
- Pynac has been very stable (hardly any changes needed).
- Nils Bruin (and Robert Bradshaw): A new C library interface to Maxima.

Burcin Erocal

Sage: Future Directions

A Typical Sage Release

- "Sage 4.5.2 was released on August 7, 2010. The following *83 people* contributed to this release (of those, 15 made their *first contribution* to Sage):

  Adam Webb, Alex Ghitza, Alexandre Blondin Massé, Alyson Deines, ...
  Willem Jan Palenstijn, William Stein, Yann Laigle-Chapuy

- Release managers: Dan Drake, Mitesh Patel
- Statistics: We closed 159 tickets (coverage score: 83.8%)
- Tickets:

Closed tickets

```
#8489: Nicolas M. Thiéry: New ‘sageexample‘ environment for sagetex
#8667: Simon King: New version of modular group cohomology spkg
... 156 other tickets omited ...
#9665: Rishikesh: Make lcalc accessible as a library on OS X
```

See http://trac.sagemath.org/sage_trac/milestone/sage-4.5.2

Future Roadmap for the Sage project

- Sage-5.0 (2010)
- Sage-6.0 (2011)
- Sage-7.0 (2012)

Roadmap of Sage project

- Sage-5.0 (2010)
  - Microsoft Windows port (via Cygwin)
  - Upgrade PARI to the latest SVN version
  - Upgrade MPIR to version 2.x
  - Get doctest coverage to 90% (currently at 84.6%)
- Sage-5.x (rest of this year...)
  - Function fields (Hess's algorithms, 2-descent, L-functions)
  - Symbolic summation, integration, and special functions:
    - arbitrary precision numerical evaluation (mpmath)
    - series expansion
    - differentiation
    - shifts (difference analogue of differentiation)
  - **And much more!**
- Sage-6.0 (2011)
- Sage-7.0 (2012)

Roadmap of Sage project

- Sage-5.0 (2010)
- Sage-6.0 (2011):
  - Noncommutative symbolic computation (Plural, Weyl Algebras, Graded algebras, Lie algebras).
  - Much improved Sage notebook (scalability, customizability)
  - Textbooks, interacts: included with Sage.
  - Commercial support (custom notebook servers)
  - Switch to using C library interfaces for GAP and Maxima.
  - Get doctest coverage to 100%
  - **And much more!**
- Sage-7.0 (2012)

Roadmap of Sage project

- Sage-5.0 (2010)
- Sage-6.0 (2011)
- Sage-7.0 (2012):
  - Faster Groebner basis computation
  - Vast improvements in Sage for Science and Engineering (documentation, diff'eq, data workflows, reproducible research, instrument support, data formats like HD5)
  - Statistics: something Pythonic/Cythonic and built on top of R + scipy.stats + GSL, which competes with SAS, etc.
  - Switch to Python 3.x
  - Randomized testing that goes beyond doctesting
  - **And much more!**

Demo

A Demo...

Use Sage From the Command Line

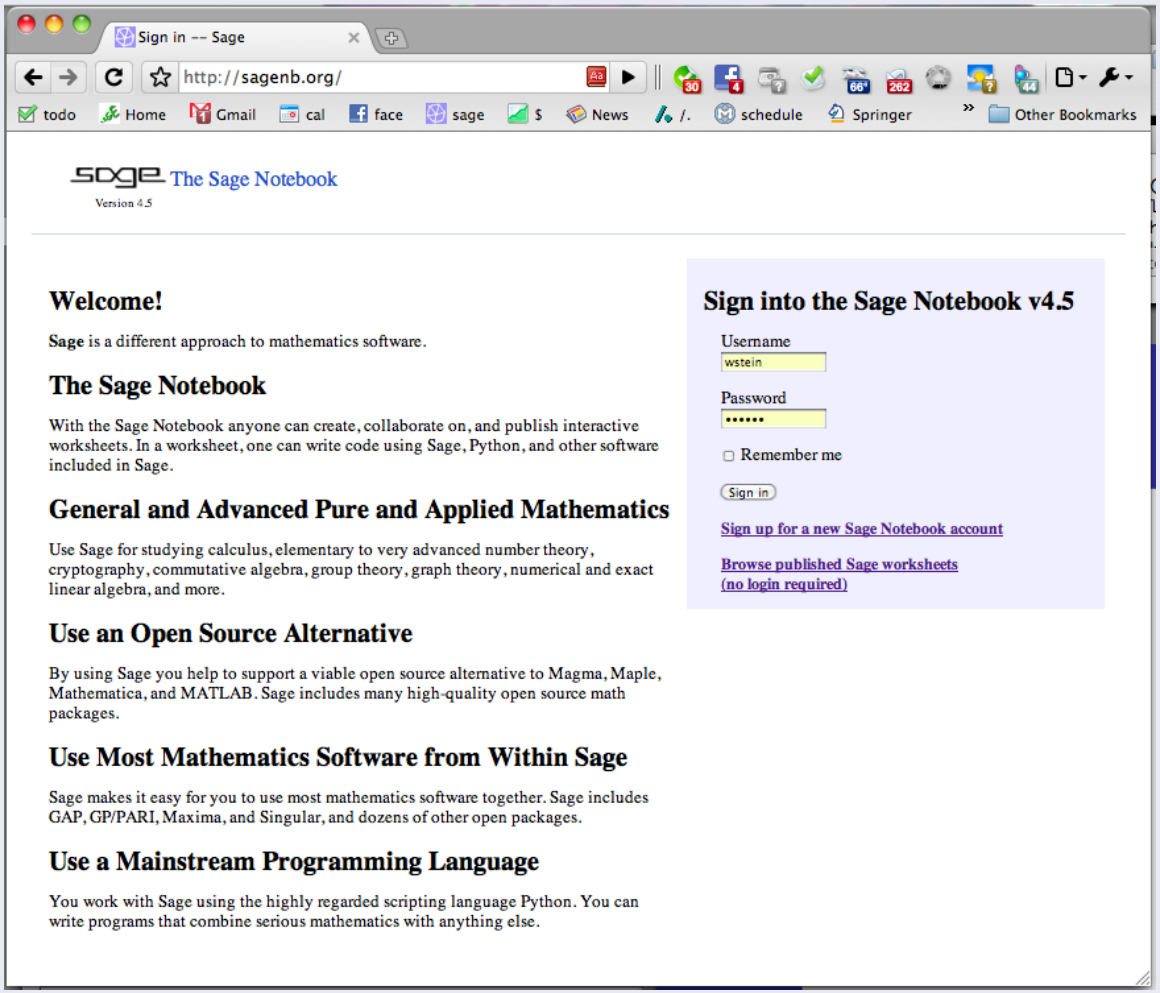
Command Line Sage

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
```

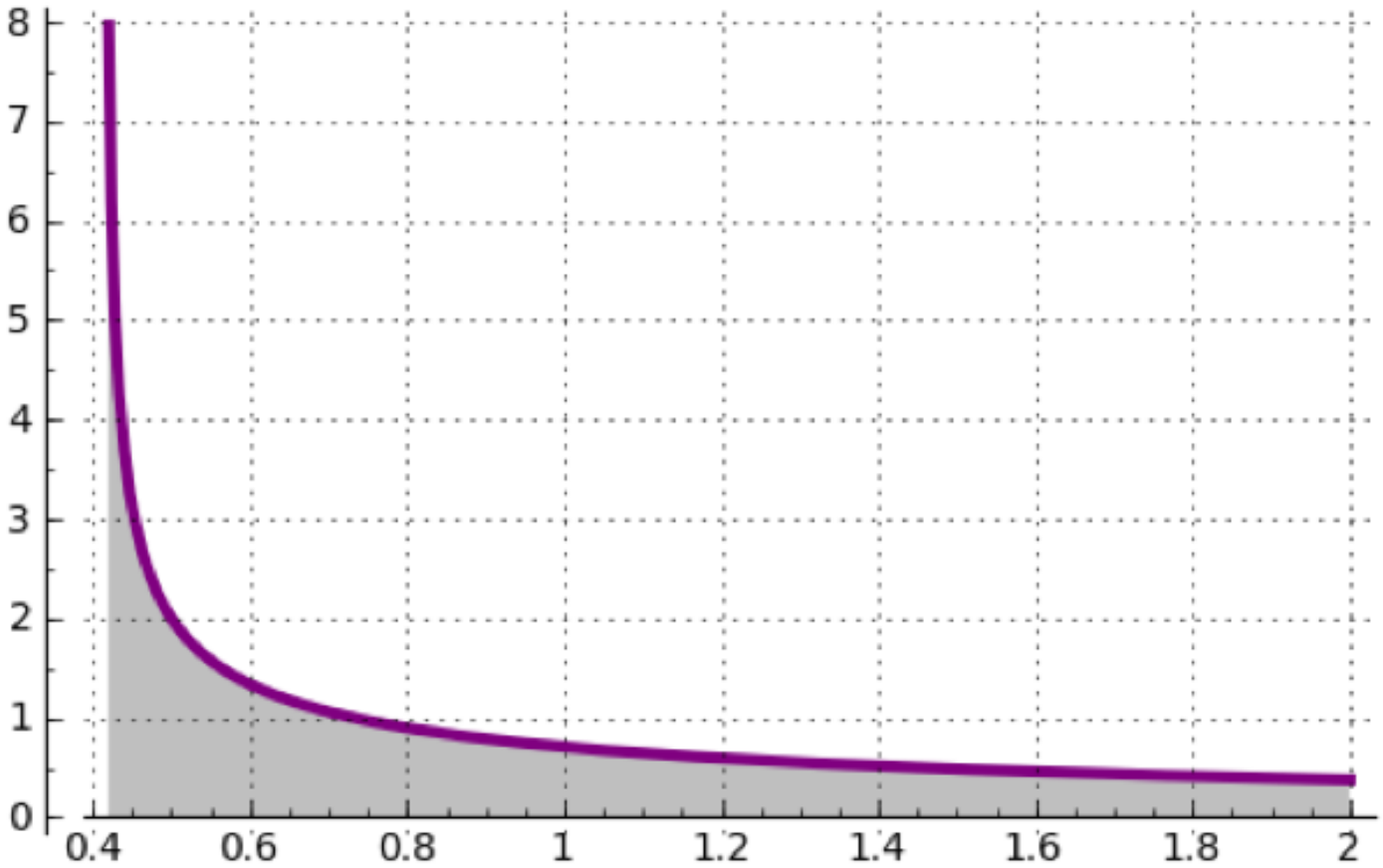
$$\frac{-\cos(x)}{2\,(\sin(x)-1)} + \frac{\cos(x)}{2\,(\sin(x)+1)} - \cos\left(x\right)$$

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

$$0$$

Demo: Plotting a 2D Function

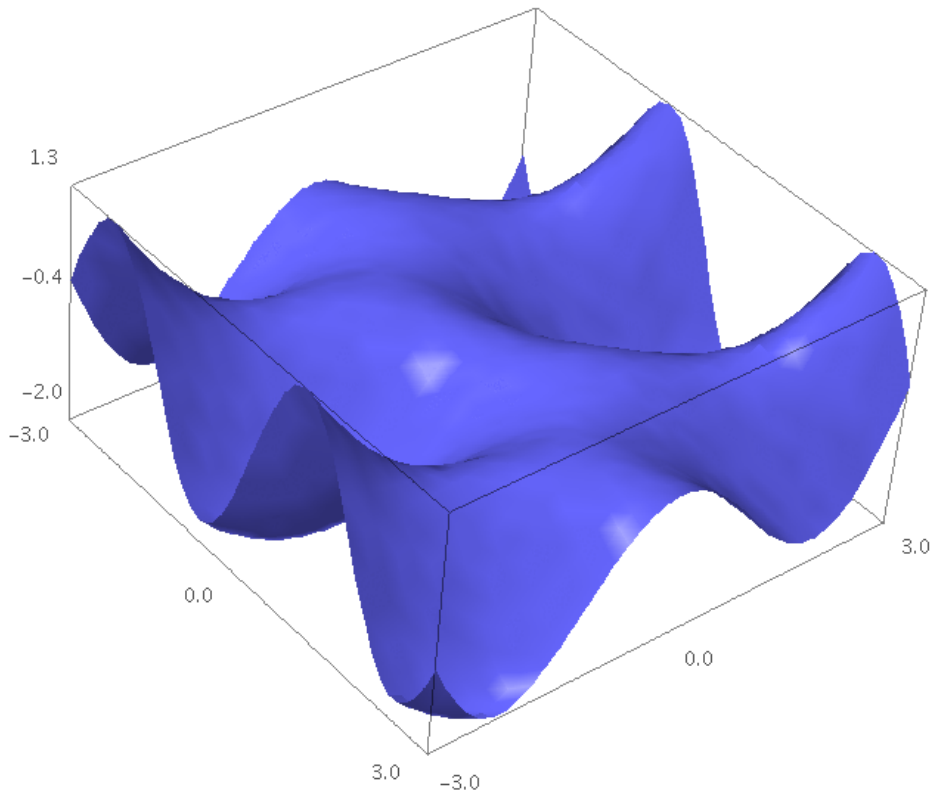

```
plot(1/sqrt(x^2 + 2*x - 1), (x,.4,2), thickness=3,
                color='purple', fill=True, gridlines=True)
```



Demo: Plotting a 3D Function


```
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.