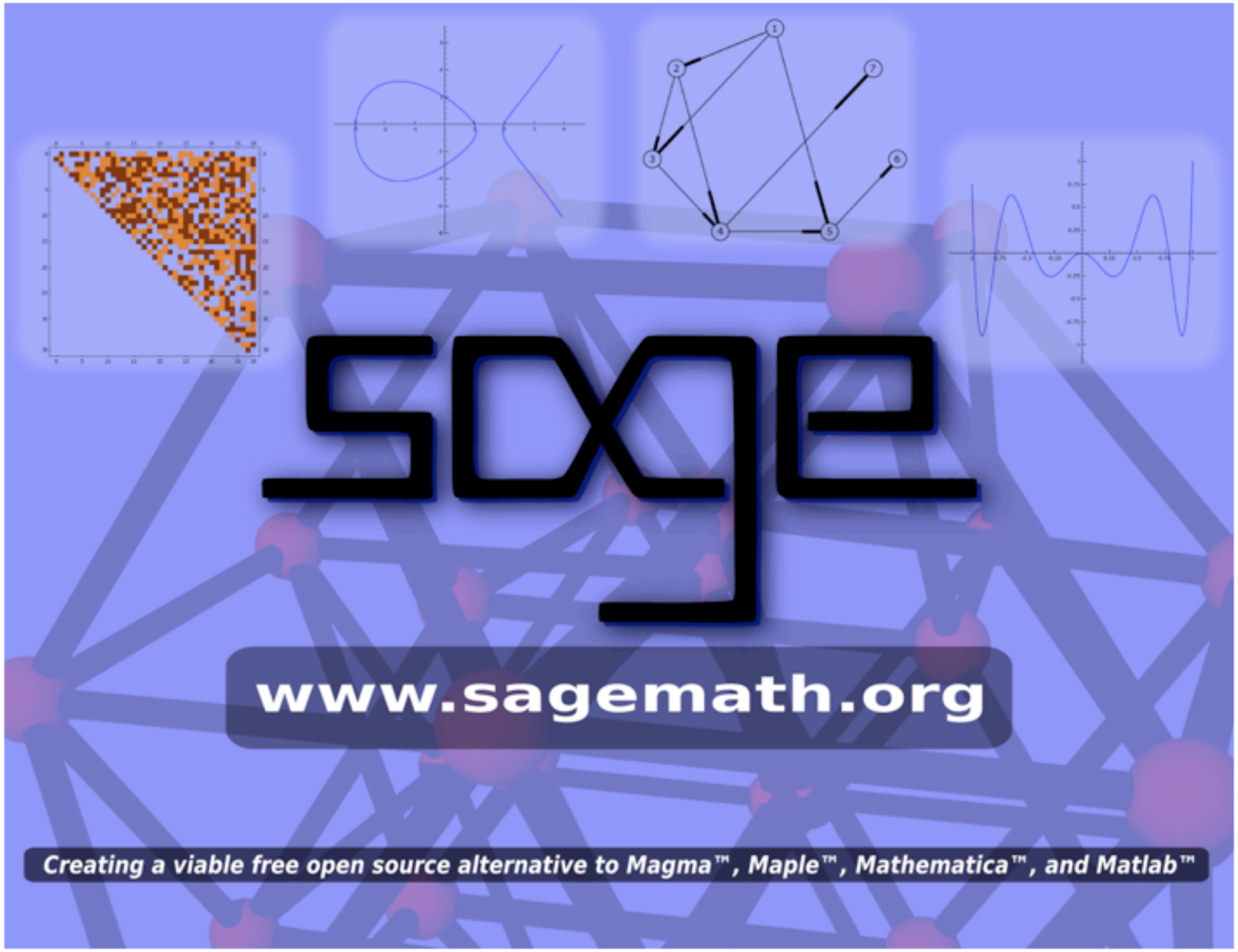
Sage

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

William Stein, Associate Professor, University of Washington

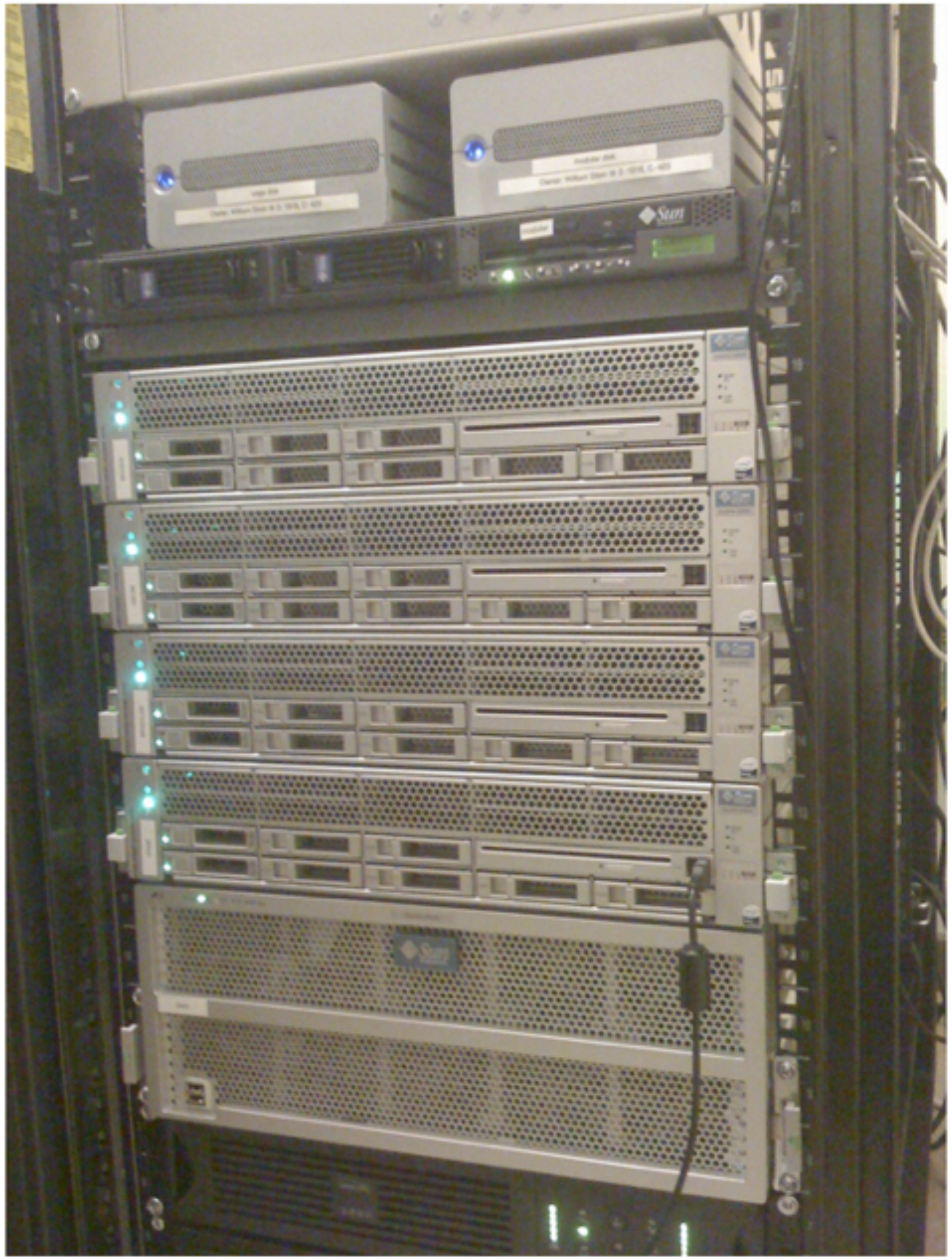


History

- *I started Sage* at Harvard in ***January 2005***.
- No existing math software ***good enough***.
- I got ***very annoyed*** that my students and colleagues had to pay a ***ridiculous amount*** to use the code I wrote in Magma for ***Arithmetic Geometry (modular forms, elliptic curves, etc.)***.
- Sage-1.0 released ***February 2006*** at Sage Days 1 (San Diego).
- ***Sage Days Workshops*** 1, 2, ..., 13, at UCLA, UW, Cambridge, Bristol, Austin, France, San Diego, Seattle, Georgia, etc.
- Sage ***won first prize*** in Trophees du Libre (November 2007)
- Funding from ***Microsoft, UW, NSF, DoD, Google, Sun,*** private donations, etc.

sagemath.org

Each day about 1500-2000 different real human visitors each day.



Sage Devel Headquarters: Four 24-core Sun X4450's with 128GB RAM each + 1 Sun X4540 with 24TB disk

sagenb.org

Mathematics Software: Welcome!

Sage is a different approach to mathematics software.

The Sage Notebook
With the Sage Notebook anyone can create, collaborate on, and publish interactive worksheets. In a worksheet, one can write code using Sage, Python, and other software included in Sage.

General and Advanced Pure and Applied Mathematics
Use Sage for studying calculus, elementary to very advanced number theory, cryptography, commutative algebra, group theory, graph theory, numerical and exact linear algebra, and more.

Use an Open Source Alternative
By using Sage you help to support a viable open source alternative to Magma, Maple, Mathematica, and MATLAB. Sage includes many high-quality open source math packages.

Use Most Mathematics Software from Within Sage
Sage makes it easy for you to use most mathematics software together. Sage includes GAP, GP/PARI, Maxima, and Singular, and dozens of other open packages.

Use a Mainstream Programming Language
You work with Sage using the highly regarded scripting language Python. You can write programs that combine serious mathematics with anything else.

Sign into the Sage Notebook

Username: wstein
Password: ••••••
Remember me
Sign In

Sign up for a new Sage Notebook account

Browse published Sage worksheets (no login required)

Some Advantages of Sage...

1. Python-- ***a general purpose*** language at core of Sage; huge user base compared to Matlab, Mathematica, Maple and Magma
2. Cython -- ***write fast compiled*** code in Sage
3. ***Free*** and ***Open source***
4. ***Peer review*** of Code: "*I do really and truly believe that the Sage refereeing model results in better code*." -- John Cremona

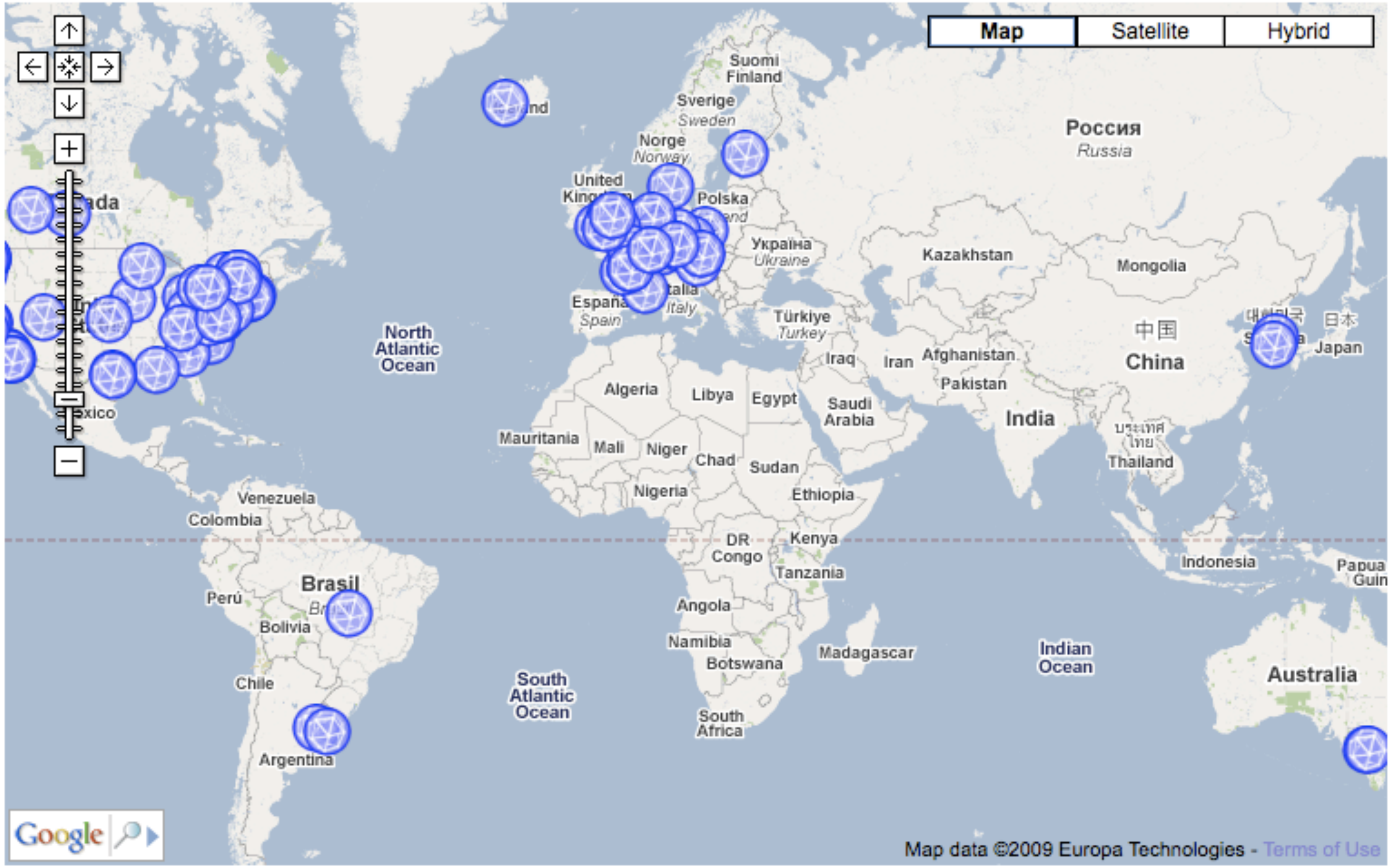
Sage Is...

- Over ***300,000 lines*** of new Python/Cython code
- ***Distribution*** of mathematical software; easy to build from source (over 5 million lines of code).
- About ***150 developers***: developer map
- ***Interfaces*** to most math software, including Magma, Macaulay 2, Singular, PHCpack, Polymake.
- Exact and numerical ***linear algebra*****,** optimization, ***statistics*** (numpy, scipy, R, and gsl all included), group theory, combinatorics, cryptography.
- **Number Theory:** elliptic curves, modular forms, L-functions -- this is *my research area*
- 2d and 3d ***plotting***

Sage developers around the world

This is a map of all contributors to the Sage project. There are currently 142 contributors in 85 different places from all around the world.

Map Zoom: Earth - USA (UW, West, East) - Europe - Asia - S. America - Australia



William Stein Tim Abbott Michael Abshoff Martin Albrecht Nick Alexander Maite Aranes Jennifer Balakrishnan Jason Bandlow Arnaud Bergeron Francois Bissey Jonathan Bober Tom Boothby Nicolas Borie Robert Bradshaw Michael Brickenstein Dan Bump Iftikhar Burhanuddin Ondrej Certik Wilson Cheung Craig Citro Francis Clarke Timothy Clemans Alex Clemesha John Cremona Karl-Dieter Crisman Doug Cutrell Tom Denton Didier Deshommes Dan Drake Alexander Dreyer Gabriel Ebner Burcin Erocal Gary Furnish Alex Ghitza Andrzej Giniewicz Amy Glen Daniel Gordon Chris Gorecki Jason Grout Carlo Hamalainen Marshall Hampton Jon Hanke Mike Hansen Bill Hart David Harvey Neal Holtz Sean Howe Alexander Hupfer Wilfried Huss Naqi Jaffery Peter Jipsen David Joyner Michael Kallweit Josh Kantor Kiran Kedlaya Simon King Emily Kirkman David Kohel Ted Kosan Sébastien Labbé Yann Laigle-Chapuy Kwankyu Lee David Loeffler Michael Mardaus Jason Martin Jason Merrill Matthias Meulien Robert Miller Kate Minola Joel Mohler Bobby Moretti Guillaume Moroz Gregg Musiker Tobias Nagel Brett Nakashima Pablo De Nápoli Minh Van Nguyen Andrey Novoseltsev Ronan Paixão Willem Jan Palenstijn John Palmieri David Perkinson Clement Pernet John Perry Pearu Peterson Yi Qiang Dorian Raymer R. Rishikesh David Roe Bjarke Hammersholt Roune Franco Saliola Kyle Schalm Anne Schilling Harald Schilly Jack Schmidt Dan Shumow Steven Sivek Nils-Peter Skoruppa Jaap Spies Blair Sutton Chris Swierczewski Philippe Theveny Nicolas Thiery Igor Tolkov Gonzalo Tornaria John Voight Justin Walker Mark Watkins Georg S. Weber Joe Wetherell Carl Witty Cristian Wuthrich Dal S. Yu Mike Zabrocki Bin Zhang Paul Zimmermann

Examples: Calculus

Symbolic expressions:

```
x, y = var('x,y')
type(x)
```

```
<class 'sage.calculus.calculus.SymbolicVariable'>
```

```
reset('e')
```

```
a = 1 + sqrt(7)^2 + pi^e + 2/3 + x^y
```

```
show(a)
```

$$x^y + \pi^e + \frac{26}{3}$$

```
show(expand(a^2))
```

$$x^{2y} + 2\pi^e x^y + \frac{52x^y}{3} + \pi^{2e} + \frac{52\pi^e}{3} + \frac{676}{9}$$

```
show(integrate(sin(x^2)))
```

$$\frac{\sqrt{\pi}\left(\left(\sqrt{2}i + \sqrt{2}\right)\mathrm{erf}\left(\frac{\left(\sqrt{2}i+\sqrt{2}\right)x}{2}\right) + \left(\sqrt{2}i - \sqrt{2}\right)\mathrm{erf}\left(\frac{\left(\sqrt{2}i-\sqrt{2}\right)x}{2}\right)\right)}{8}$$

Example: Some Rings

```
QQ[sqrt(2)]
```

```
Number Field in sqrt2 with defining polynomial x^2 - 2
```

```
RDF
```

```
Real Double Field
```

```
R = RealIntervalField(100); R
```

```
Real Interval Field with 100 bits of precision
```

```
R((1.01030,1.010332))
```

```
1.0103?
```

```
A = random_matrix(QQ, 500); v = random_matrix(QQ,500,1)
time x = A \ v
```

```
Time: CPU 1.56 s, Wall: 1.54 s
```

```
len(str(x[0]))
```

```
1484
```

Example: A Huge Integer Determinant

```
a = random_matrix(ZZ,200,200,x=-2^127,y=2^127)
time d = a.determinant()
len(str(d))
```

```
Time: CPU 3.78 s, Wall: 3.85 s
7786
```

We can also ***copy this matrix*** over to Maple and compute the same determinant there...

```
a[0,0]
```

```
-91532941219663566551217034654788002094
```

```
maple.with_package('LinearAlgebra')
B = maple(a)
t = maple.cputime()
time c = B.Determinant()
maple.cputime(t)
```

```
Time: CPU 0.00 s, Wall: 23.18 s
22.613
```

```
c == d
```

```
True
```

This ability to easily move objects between math software is ***unique to Sage***.

```
B = magma(a)
t = magma.cputime()
time e = B.Determinant()
magma.cputime(t)
```

```
Time: CPU 0.00 s, Wall: 10.74 s
10.51
```

Example: Plotting a Symbolic Expression

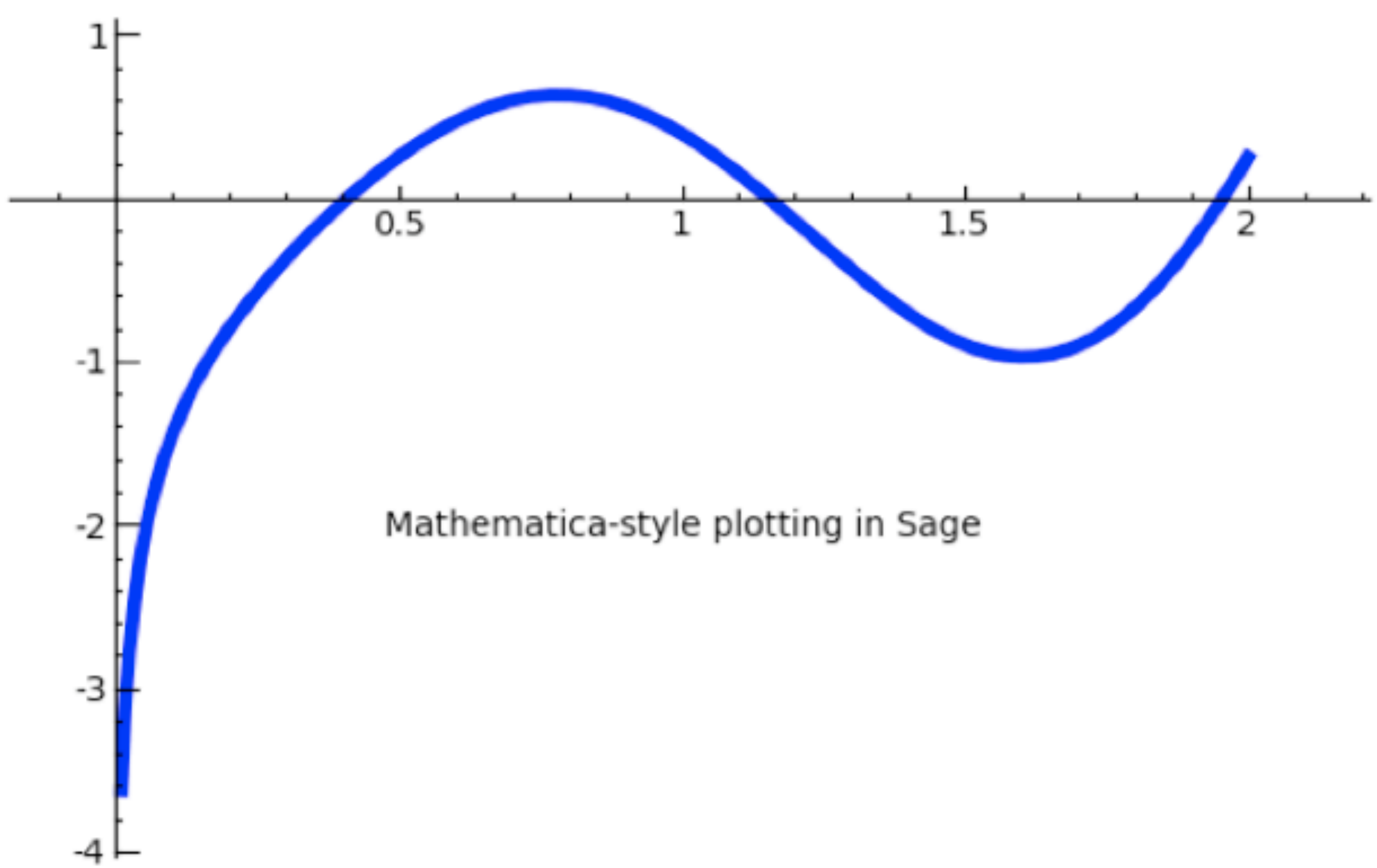
```
x = var('x')
f(x) = sin(3*x)*x+log(x) + 1/(x+1)^2
show(f)
```

evaluate

$$x \mapsto x\sin\left(3x\right) + \log\left(x\right) + \frac{1}{\left(x+1\right)^2}$$

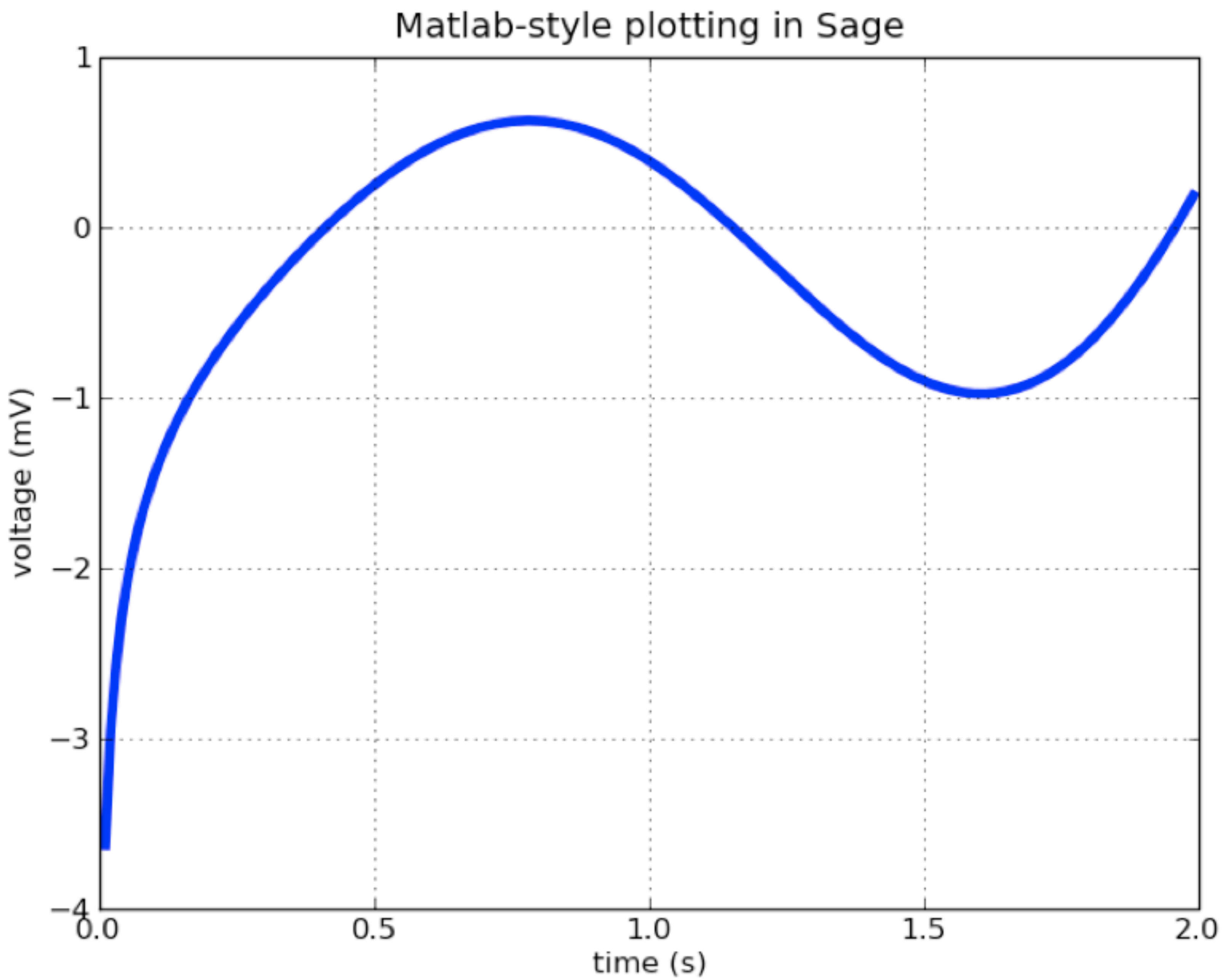
Plotting functions has similar syntax to Mathematica:

```
plot(f,(0.01,2), thickness=4) + text("Mathematica-style plotting in Sage", (1,-2),
rgbcolor='black')
```



NOTE: Sage also has 2d plotting that is almost *identical to MATLAB's 2d plotting*:

```
import pylab as p
p.figure()
t = p.arange(0.01, 2.0, 0.01)
s = p.sin(2 * p.pi * t)
s = p.array([float(f(x)) for x in t])
P = p.plot(t, s, linewidth=4)
p.xlabel('time (s)'); p.ylabel('voltage (mV)')
p.title('Matlab-style plotting in Sage')
p.grid(True)
p.savefig('sage.png')
```



Example: Fast Evaluation

```
f(x,y,z) = sin(3*x)*x + log(x*y*z)^3 + 1/(1+y)^2
```

```
g = f._fast_float_(x,y,z)
timeit('g(4.5r,3.2r,5.7r)')
```

```
    625 loops, best of 3: 669 ns per loop
```

```
%python
import math
def g(x,y,z): return math.sin(3*x)*x + log(x*y*z)**3 + 1/(1+y)**2
```

```
timeit('g(4.5r,3.2r,5.7r)')
```

```
    625 loops, best of 3: 7.15 Âµs per loop
```

```
7.15/.669
```

```
    10.6875934230194
```

Example: Interface with Maple and Mathematica

```
var('x')
f = sin(3*x)*x+log(x) + 1/(x+1)^2
show(integrate(f))
```

$$\frac{\sin(3x) - 3x\cos(3x)}{9} + x\log(x) - \frac{1}{x+1} - x$$

The command maple(...) fires up Maple (if you have it!), and creates a reference to a live object:

```
m = maple(f); m
```

```
sin(3*x)*x+ln(x)+1/(x+1)^2
```

```
type(m)
```

```
<class 'sage.interfaces.maple.MapleElement'>
```

```
m.parent()
```

```
Maple
```

```
m.parent().pid()
```

```
67798
```

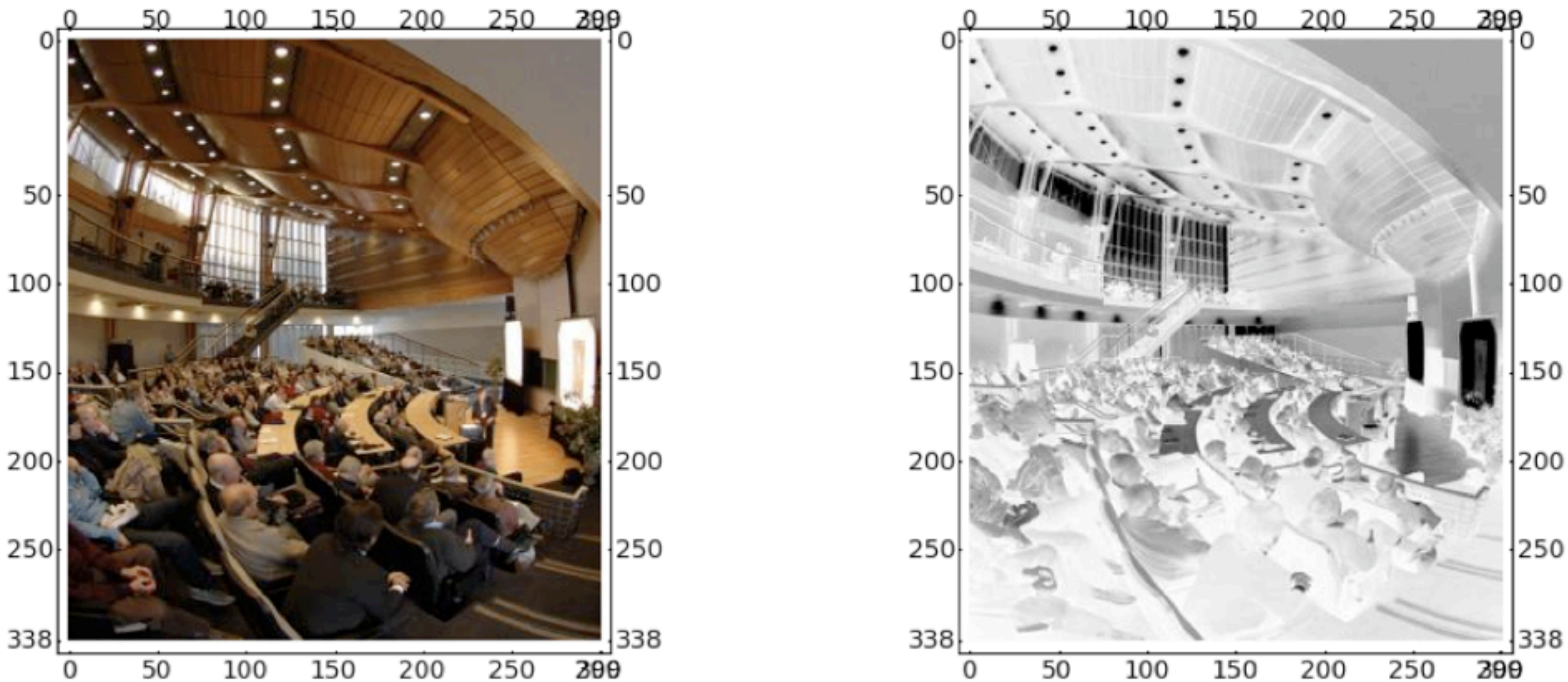
```
os.system('ps ax |grep 24038')
```

```
67818 s001  S+     0:00.00 sh -c ps ax |grep 24038
67820 s001  R+     0:00.00 grep 24038
0
```

Example: Interactive Image Compression

This illustrates pylab (matplotlib + numpy), Sage plotting, html output, and @interact.

```
# first just play
import pylab
A = pylab.imread(DATA + 'simons.png')
graphics_array([matrix_plot(A), matrix_plot(1-A[0:,0:,1])]).show(figsize=[10,4])
```




```
A[0,0,]
```

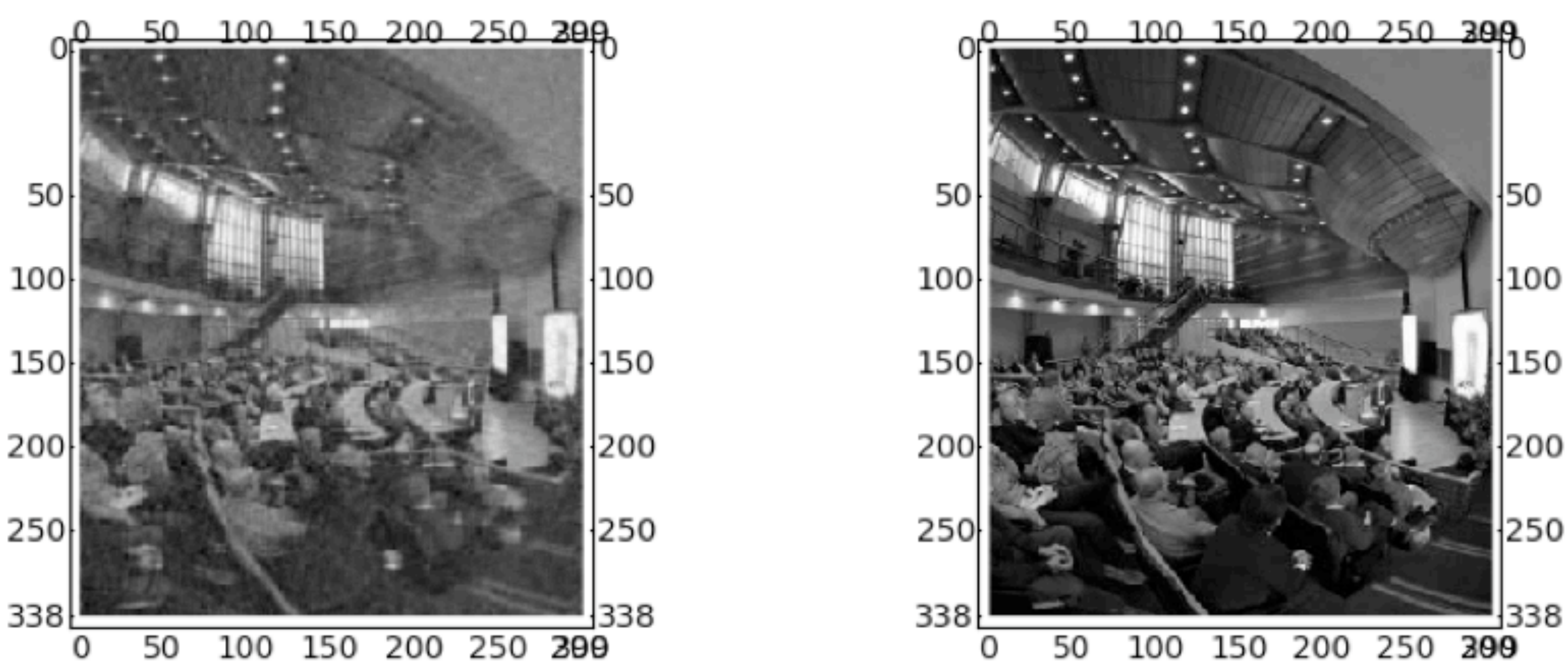
```
array([ 0.03529412,  0.04705882,  0.01960784,  1.        ],
dtype=float32)
```

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'simons.png'), 2)
@interact
def svd_image(i=(20,(1..100)), display_axes=True):
    u,s,v = pylab.linalg.svd(A_image)
    A     = sum(s[j]*pylab.outer(u[0:,j], v[j,0:]) for j in range(i))
    g = graphics_array([matrix_plot(A),matrix_plot(A_image)])
    show(g, axes=display_axes, figsize=(8,3))
    html('<h2>Compressed using %s eigenvalues</h2>'%i)
```

i 38

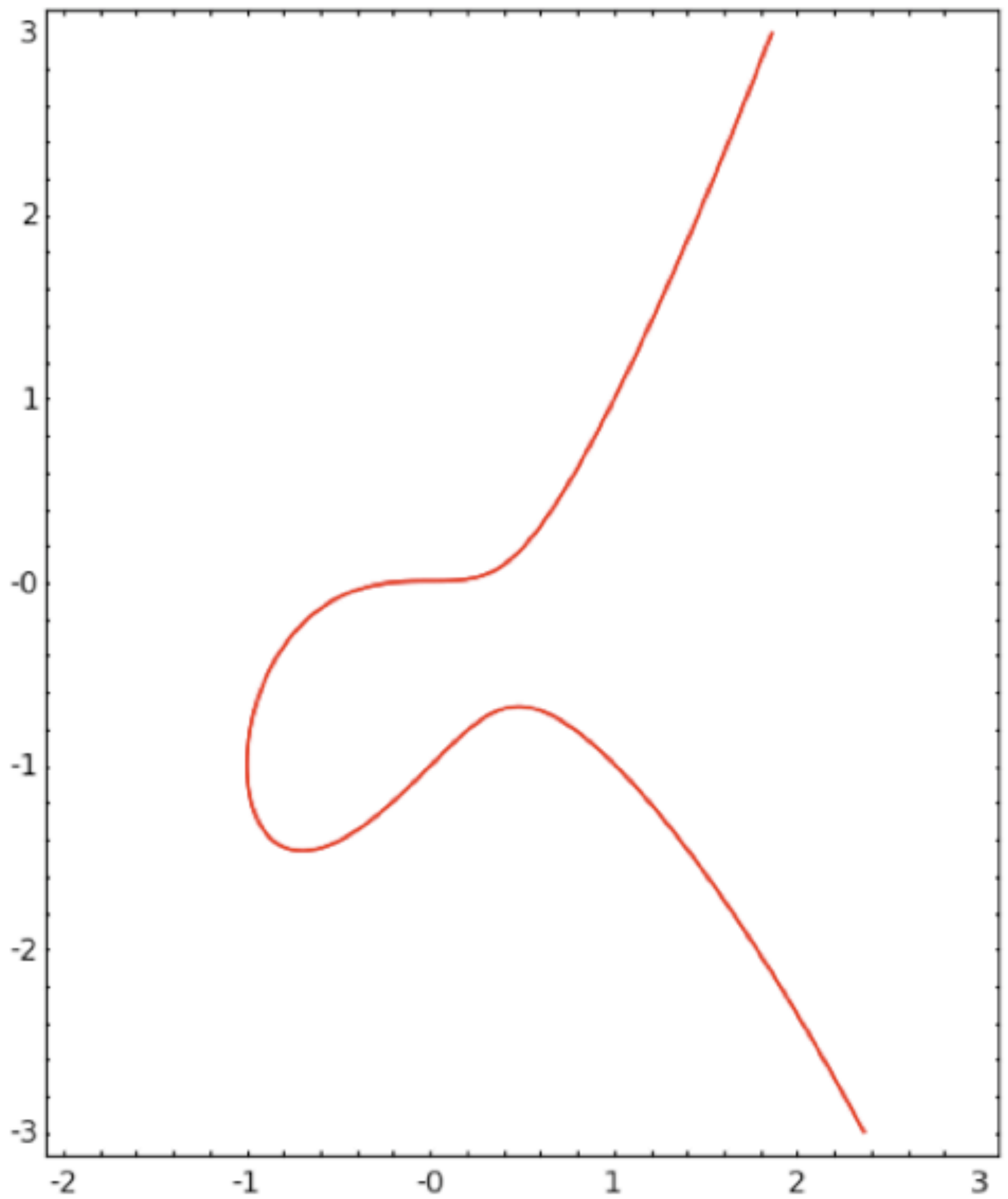
display_axes ☑

Compressed using 38 eigenvalues



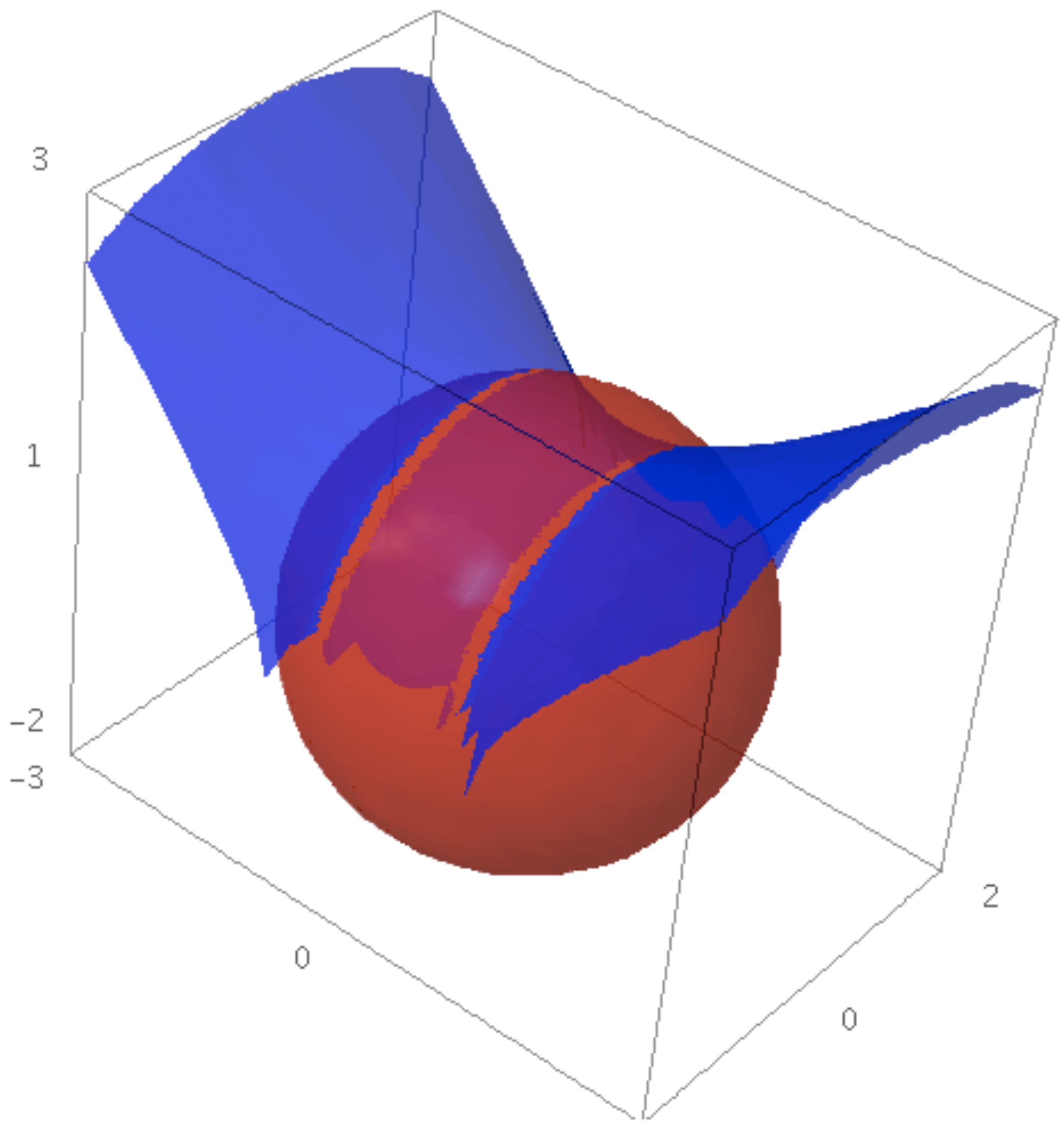
Example: Implicit 2d Plot

```
var("x y")
implicit_plot(y^3 + x*y - x^2 - x, (-2,3), (-3,3), plot_points=400, cmap='hsv').show(aspect_ratio=1)
```

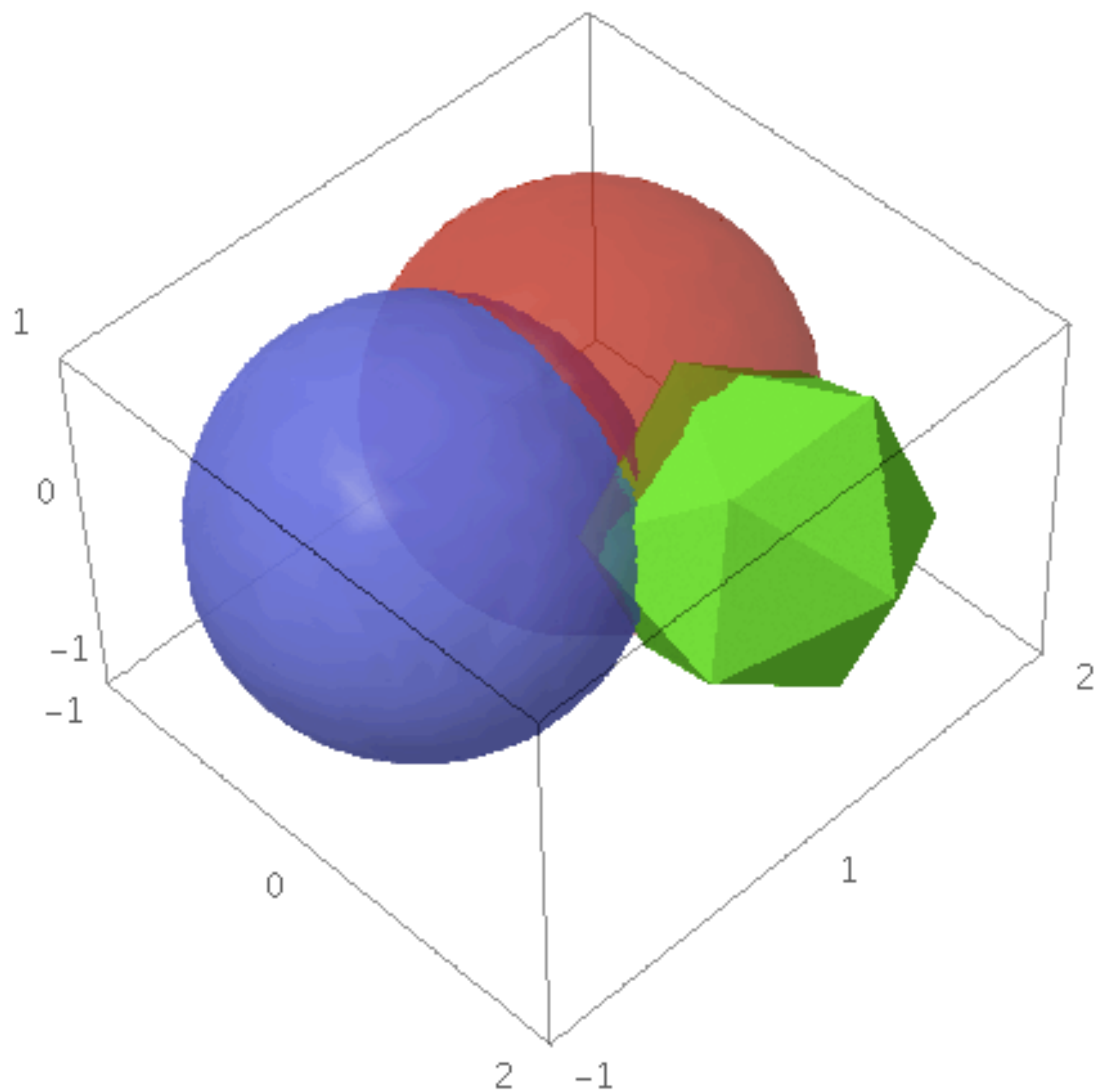


Example: 3d Plots

```
var('x,y')
P = plot3d(sqrt(x^2-y^2+3), (x,-3,3), (y,-2,2), color='blue', opacity=0.7)
P + sphere((0,0,0),2,color='red', aspect_ratio=[1,1,1], opacity=0.6)
```

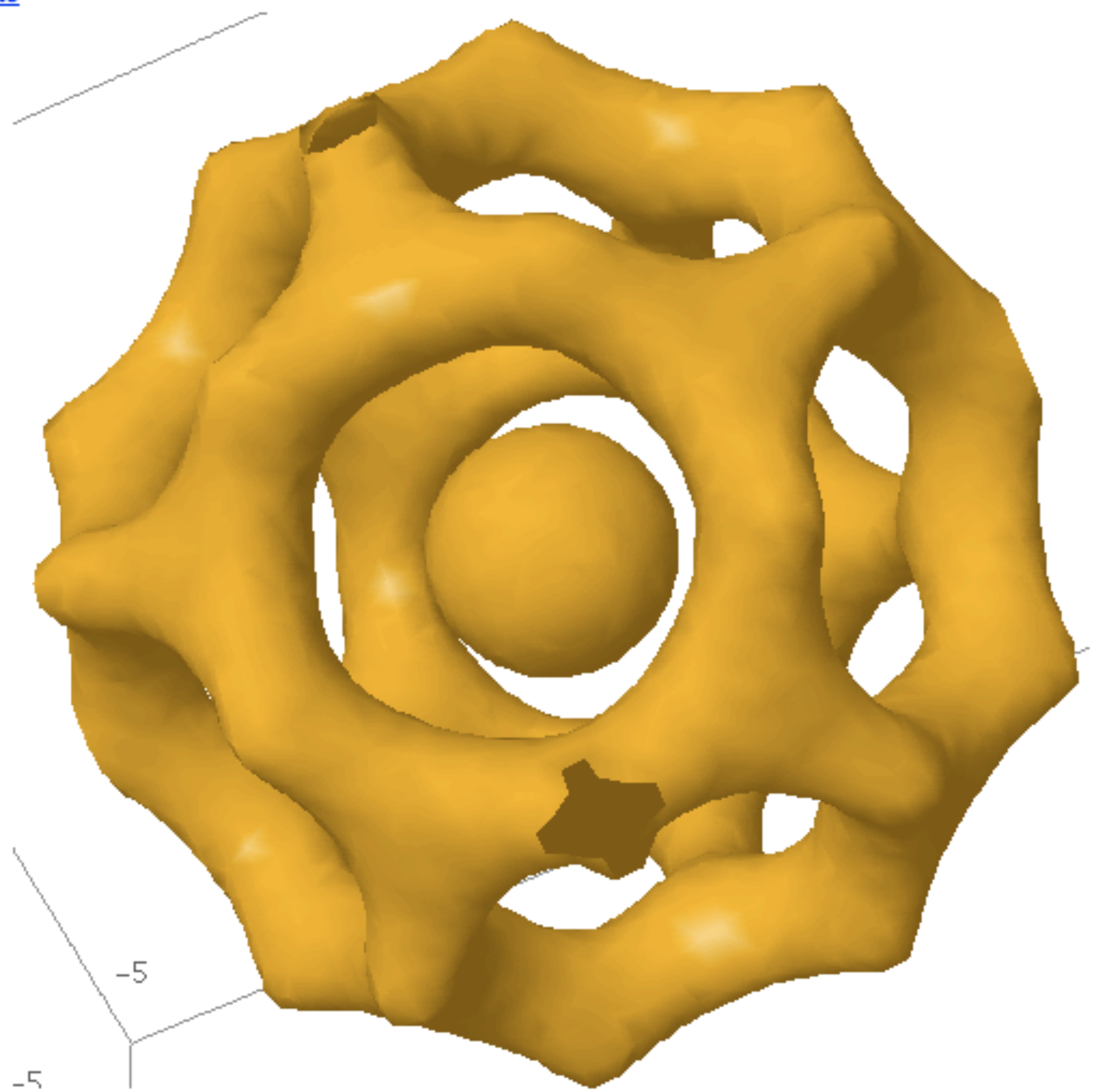


```
( sphere((0,0,0), opacity=0.7) + sphere((0,1,0), color='red', opacity=0.5)
       + icosahedron((1,1,0), color='green') )
```



```
# implicit_plot3d -- not yet released
# (see http://trac.sagemath.org/sage_trac/ticket/5249)!!
var('x,y,z')
T = RDF(golden_ratio)
p = (2 - (cos(x + T*y) + cos(x - T*y) + cos(y + T*z)
        + cos(y - T*z) + cos(z - T*x) + cos(z + T*x)))
r = 4.77
implicit_plot3d(p, (-r, r), (-r, r), (-r, r), plot_points=40)
```

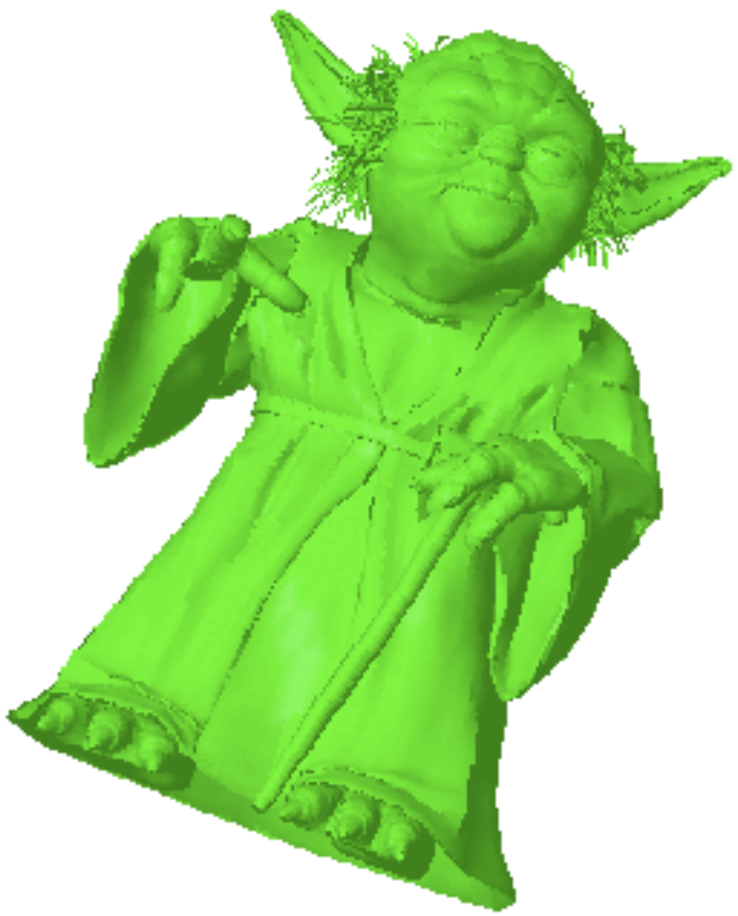
evaluate



3d plotting (using jmol) is fast even though it does **not** use Java3d or OpenGL or require any special signed code or drivers.

```
# Yoda! -- over 50,000 triangles.
from scipy import io
X = io.loadmat(DATA + 'yodapose.mat')
from sage.plot.plot3d.index_face_set import IndexFaceSet
V = X['V']; F3=X['F3']-1; F4=X['F4']-1
Y = IndexFaceSet(F3,V,color='green') + IndexFaceSet(F4,V,color='green')
Y = Y.rotateX(-1)
Y.show(aspect_ratio=[1,1,1], frame=False, figsize=4)
html('"Use the source, Luke..."')
```

"Use the source, Luke..."



Example: Write Fast Code (using Cython)

```
to      sage-support
date    Sat, Jan 31, 2009 at 11:15 AM
Hi,

I received first a MemoryError, and later on Sage reported:

uitkomst1=[]
uitkomst2=[]
eind=int((10^9+2)/(2*sqrt(3)))
print eind
for y in srange(1,eind):
 test1=is_square(3*y^2+1,True)
 test2=is_square(48*y^2+1,True)
 if test1[0] and test1[1]%3==2: uitkomst1.append((y,(2*test1[1]-1)/3))
 if test2[0] and test2[1]%3==1: uitkomst2.append((y,(2*test2[1]+1)/3))
print uitkomst1
een=sum([3*x-1 for (y,x) in uitkomst1 if 3*x-1<10^9])
print uitkomst2
twee=sum([3*x+1 for (y,x) in uitkomst2 if 3*x+1<10^9])
print een+twee

If you replace 10^9 with 10^6, the above listing works properly.

Maybe I made a mistake?

Rolandb
```

I rewrote Roland's code slightly so it wouldn't waste memory by constructing big lists... but the result was slow.

I rewrote Roland's code slightly so it wouldn't waste memory by constructing big lists... but the result was slow.

```
def f_python(n):
    uitkomst1=[]
    uitkomst2=[]
    eind=int((n+2)/(2*sqrt(3)))
    print eind
    for y in (1..eind):
        test1=is_square(3*y^2+1,True)
        test2=is_square(48*y^2+1,True)
        if test1[0] and test1[1]%3==2:
            uitkomst1.append((y,(2*test1[1]-1)/3))
        if test2[0] and test2[1]%3==1:
            uitkomst2.append((y,(2*test2[1]+1)/3))
    print uitkomst1
    een=sum(3*x-1 for (y,x) in uitkomst1 if 3*x-1<10^9)
    print uitkomst2
    twee=sum(3*x+1 for (y,x) in uitkomst2 if 3*x+1<10^9)
    print een+twee
```

```
time f_python(10^5)
```

```
28868
[(1, 1), (15, 17), (209, 241), (2911, 3361)]
[(1, 5), (14, 65), (195, 901), (2716, 12545)]
51408
Time: CPU 0.72 s, Wall: 0.77 s
```

```
time f_python(10^6)
```

```
288675
[(1, 1), (15, 17), (209, 241), (2911, 3361), (40545, 46817)]
[(1, 5), (14, 65), (195, 901), (2716, 12545), (37829, 174725)]
716034
Time: CPU 7.14 s, Wall: 7.65 s
```

Rewrite using *CYTHON*

```
%cython
from sage.all import is_square
cdef extern from "math.h":
    long double sqrtl(long double)
def f(n):
    uitkomst1=[]
    uitkomst2=[]
    cdef long long eind=int((n+2)/(2*sqrt(3)))
    cdef long long y, t
    for y in range(1,eind):
        t = <long long>sqrtl(<long long> (3*y*y + 1))
        if t * t == 3*y*y + 1:
            uitkomst1.append((y, (2*t-1)/3))
        t = <long long>sqrtl(<long long> (48*y*y + 1))
        if t * t == 48*y*y + 1:
            uitkomst2.append((y, (2*t+1)/3))
    print uitkomst1
    een=sum([3*x-1 for (y,x) in uitkomst1 if 3*x-1<10^9])
    print uitkomst2
    twee=sum([3*x+1 for (y,x) in uitkomst2 if 3*x+1<10^9])
    print een+twee
```

_Users_ws..._code_sage104_spyx.c   _Users_ws...de_sage104_spyx.html

```
time f(10^6)
```

```
[(1L, 1L), (4L, 4L), (15L, 17L), (56L, 64L), (209L, 241L), (780L, 900L), (2911L, 3361L), (10864L, 12
[(1L, 5L), (14L, 65L), (195L, 901L), (2716L, 12545L), (37829L, 174725L)]
2
Time: CPU 0.03 s, Wall: 0.03 s
```

A speedup by a factor of 238!!

```
7.14/0.03
```

evaluate

```
238.000000000000
```

```
time f(10^9)
```

```
288675135
[(1L, 1L), (4L, 4L), (15L, 17L), (56L, 64L), (209L, 241L), (780L, 900L), (2911L, 3361L), (10864L, 12
[(1L, 5L), (14L, 65L), (195L, 901L), (2716L, 12545L), (37829L, 174725L), (526890L, 2433601L), (73386
2
Time: CPU 25.60 s, Wall: 26.50 s
```

Example: Multiply Polynomials

```
R.<x,y,z> = GF(997)[]; R
```

```
Multivariate Polynomial Ring in x, y, z over Finite Field of size 997
```

```
show((x+y+z+3)^2)
```

$$x^2 + 2xy + y^2 + 2xz + 2yz + z^2 + 6x + 6y + 6z + 9$$

```
f = (x+y+z+1)^20
time g = f*(f+1)
```

```
Time: CPU 0.11 s, Wall: 0.12 s
```

```
%magma
R<x,y,z> := PolynomialRing(GF(997),3);
f := (x+y+z+1)^20;
time g := f*(f+1);
```

```
Time: 0.280
```

```
mf = magma(f)
time g = mf*(mf+1)
```

```
Time: CPU 0.00 s, Wall: 0.30 s
```

Example: Compute a Groebner Bases

```
P.<a,b,c> = PolynomialRing(QQ,3, order='lex'); P
```

```
Multivariate Polynomial Ring in a, b, c over Rational Field
```

```
I = sage.rings.ideal.Katsura(P,3); show(I.gens())
```

$$\left(a + 2b + 2c - 1, a^2 - a + 2b^2 + 2c^2, 2ab + 2bc - b\right)$$

```
I.groebner_basis()
```

```
[a - 60*c^3 + 158/7*c^2 + 8/7*c - 1, b + 30*c^3 - 79/7*c^2 + 3/7*c, c^4 - 10/21*c^3 + 1/84*c^2 + 1/84*c]
```

NOTE: Sage includes Singular and is very tightly integrated with it.

```
sage.rings.ideal.Katsura(P,3).groebner_basis('singular:std')
```

```
[a - 60*c^3 + 158/7*c^2 + 8/7*c - 1, b + 30*c^3 - 79/7*c^2 + 3/7*c, c^4 - 10/21*c^3 + 1/84*c^2 + 1/84*c]
```

```
sage.rings.ideal.Katsura(P,3).groebner_basis('singular:slimgb')
```

```
[a - 60*c^3 + 158/7*c^2 + 8/7*c - 1, b + 30*c^3 - 79/7*c^2 + 3/7*c, c^4 - 10/21*c^3 + 1/84*c^2 + 1/84*c]
```

NOTE: You must have Macaulay 2 installed for this to work...

```
sage.rings.ideal.Katsura(P,3).groebner_basis('macaulay2')
```

```
[a - 60*c^3 + 158/7*c^2 + 8/7*c - 1, b + 30*c^3 - 79/7*c^2 + 3/7*c, c^4
- 10/21*c^3 + 1/84*c^2 + 1/84*c]
```

```
sage.rings.ideal.Katsura(P,3).groebner_basis('magma')
```

```
[a - 60*c^3 + 158/7*c^2 + 8/7*c - 1, b + 30*c^3 - 79/7*c^2 + 3/7*c, c^4
- 10/21*c^3 + 1/84*c^2 + 1/84*c]
```

Example: Compute a Groebner Fan

```
P.<a,b,c> = PolynomialRing(QQ,3, order='lex'); P
I = sage.rings.ideal.Katsura(P,3)
```

```
F = I.groebner_fan(); F
```

```
Groebner fan of the ideal:
Ideal (a + 2*b + 2*c - 1, a^2 - a + 2*b^2 + 2*c^2, 2*a*b + 2*b*c - b) of
Multivariate Polynomial Ring in a, b, c over Rational Field
```

```
F.weight_vectors()
```

```
[(4, 4, 1), (3, 2, 1), (5, 2, 3), (2, 1, 3), (2, 3, 5), (2, 3, 1), (2,
5, 3), (1, 4, 4)]
```

```
F.render().show(axes=False)
```