TACC -- 1. Introduction to Sage

An Introduction to Sage

William Stein, Professor of Mathematics, University of Washington

Mission Statement

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

Firefox <--> Internet Explorer, Opera

Open Office, Latex <--> Microsoft Office

Linux <--> Microsoft Windows

PostgreSQL, MySQL <--> Oracle, Microsoft SQLserver

GIMP <--> Photoshop

Sage <--> Magma, Maple, Mathematica, Matlab

History of Sage

- *I started Sage* at Harvard in **January 2005**.
- Sage-1.0 released **February 2006** at Sage Days 1 (UC San Diego).
- **Nearly 30 Sage Days Workshops** (!) at UCLA, UW, Cambridge, Bristol, Austin, France, San Diego, Seattle, MSRI, ..., Barcelona, ...
- Sage **won first prize** in the Trophees du Libre (November 2007)
- Funding from **Microsoft, Univ of Washington, UC San Diego, NSF, DoD, Google, Sun,** private donations, etc.
- **Hundreds** of other people subsequently got involved in Sage's

development, and the scope of the project has widened to cover ***all mathematical computation***. There is interest from all areas of mathematics, physics, engineering, etc. (Developer mailing list has 1184 subscribers and about 50 messages/day... and sage-support has 1718 subscribers.)

What is Sage?

- Sage = Python + Math
- A unified self-contained **distribution** of open source ***free*** mathematical software.
- Nearly a **half million lines of new Python (and Cython) code** that implements new capabilities and algorithms.
- A "**cloud" application** like GMail or Google Docs: http://sagenb.org (currently about 30,000 users); Of course Sage also runs on ***your*** desktop and supercomputer.

Tour of the http://sagemath.org website

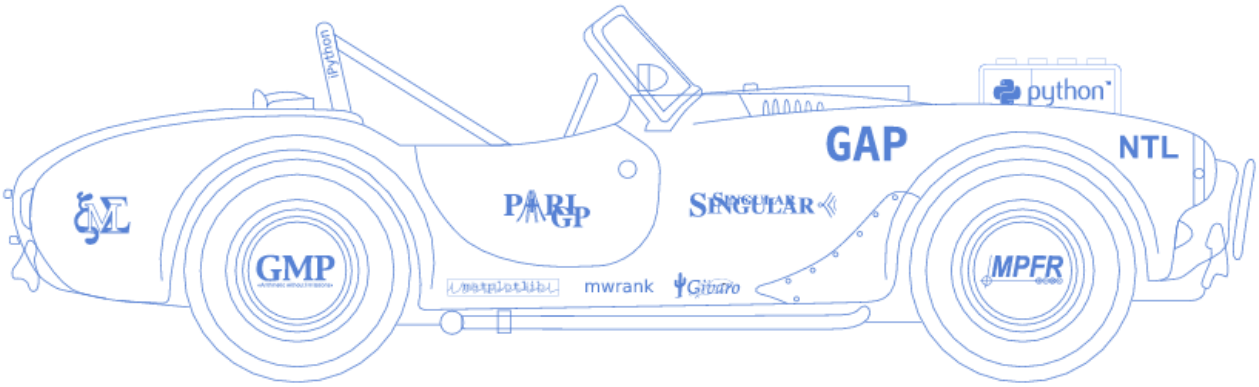
- Google page search
- Documentation
- Live Chat: http://sagemath.org/help-irc.html
- Downloading and Installing Sage
- Developer map

Summary:
Sage is about building the car instead of reinventing the wheel

1. Sage uses Python, a *mainstream programming language*, instead of inventing a custom mathematics language
2. Use straightforward method to link programs together -- *C library*

and pseudotty's, instead of XML servers/OpenMath. **We** implement all conversion routines, instead of expecting upstream to do it: we ***make them*** communicate with Sage, whether they want to or not.

3. *Give copious credit to contributors* and be very developer friendly (easily build from source).
4. Reuse, improve, and *contribute to existing libraries and projects* (e.g., Scipy, Numpy, R, ATLAS, CVXopt, GSL), instead of starting over and competing with them.
5. Make the GUI using a web browser: the world of java and javascript is available and Sage *integrates with the web*.



»Every free computer algebra system I've tried has reinvented many times the wheel without being able to build the car.«

And now a demo...

Very Big Numbers

Very big numbers: In less than a second, Sage exactly computes $(10^6)!$, which has over 5 million digits.

```
time n = factorial(10^6)
```

```
Time: CPU 0.68 s, Wall: 0.79 s
```

```
n.ndigits()
```

```
5565709
```

```
time m = n * (n+1)
```

```
Time: CPU 0.38 s, Wall: 0.39 s
```

and a **matrix** with a big determinant:

```
a = random_matrix(ZZ,200, x=2^128); a
```

```
200 x 200 dense matrix over Integer Ring (type 'print a.str()' to
see all of the entries)
```

```
a = random_matrix(ZZ,200, x=2^128); a
```

```
200 x 200 dense matrix over Integer Ring (type 'print a.str()' to
see all of the entries)
```

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

```
50110420315670456383223454274407788191
```

```
time a.determinant()
```

1121810520523080974656631619019872334140792747676595779920319979921
7035950126846293332383280971863257042078310032291276787887552961323
7468592341142460886190515509450222101477918147750547051743861165817
3768367676017813073097278497499249820091789408821736982881442638152
6885352831263837872621655396985941434034760470474317858708414313396
5899823418060606351060595348606580507982277525447817776456629742788
2358381146313695911111549170516368353009467901530938286666573131552
6166700506575420273478237005973112235762192331017991880478031663685
6346484642689366265149406543140717265621618149134812073698237363997
1827120178369108712926572879981864792842769012194518912959817574725
6956797435102431590434381351874066892787102840758266606997294517607
9538176327755629532401174627191322417176387908815565452056556065853
8590327972151723574448835501809768823645149034233570371983355300203
0410405277233100923890701970933947692628646843234266954221970817404
9445714114735549193400919169917066050555345309267128927803869233377
7486530182174514608688251245679394971005199417960649327672176832407
6207004421123731164865126688300864815575949209091475064071778855370
6647236022333825913105571797564132067148418354302852936982767159543
4120517762987174671329235846672238801597691730728310974368663993766
5906768535810426279962360175453862328734790267932709059632621613093
7673897798027036836663035257833812737315617990728234078234073667898
8228489475891446853693603765777674324913760600762182078147155339012
5983771352045053467791792005738002051106325901228319486500444738435
0709296192627890557705528815055345642742046508326499486537673210799
6686020817193328182973026699651937055848142590085223091613916745016
2506945295557969312435101081151542062804701451671374063373179721272
4937699640879285304093854511591269050029370930299731337801812579525
6119277223352234965362937023214626021076825946403121117212913285674
5537497210413732508523048519500318226538669037743905442701188546878
9050306503433784204145360149229374849248362487882759140506584691478
6953442572369920870110170994921270430674705015155178174156156684173
0206496850193538181984086673862827912169832623326215886404840607241
9188986857204529434792968649459368354238874081537890087303720022981
9894229090250272062505697552613007445738727525580461834891686994564
2886812977393967826767080836045931271060332836992538805068337689575
0139775388829132231775970652018474762047685741020477397064598308635
3106615522544416063308798745930642426775462861036263579667558460329
8303645171898999797096250589035496934402282408462209923236082585023
1577773506354022881793685659977176811363016962518236264428107264768
1612014470268062647275891678655197871675825687207580624049587717210
8046455130424038088456781748367770829099431571597744726213961923976
8884135542471439835938124172713633537608106701264778386863204761899
6670596288079307164849542965100570860721849500391439936814861494439
5013243797083171747371190036229296930436562859848198843798210556063
9566124987431410843693631248288603858746496696281112465700475117355
1567620027836321250875784236955150430029951339677664097838573513408
0839198644335782062277151790639641509614166659702542327889305800347
1337176494122843497349052838762846193304315803500730282856066253799
3030849786305610241126137166324266964730081426444967385265679377608
4918475541010007669523927860942321488018268875236583476282681478600
9622751178838747891907713841008977454119680045974014697538706531251
9228280725009584676954195731049565401171906104661121788536779462404
4068698517576708671292669674415631198689531441706120774600393775128
6963798446503494009323285791272162784174447771492728678150778580604

```
```

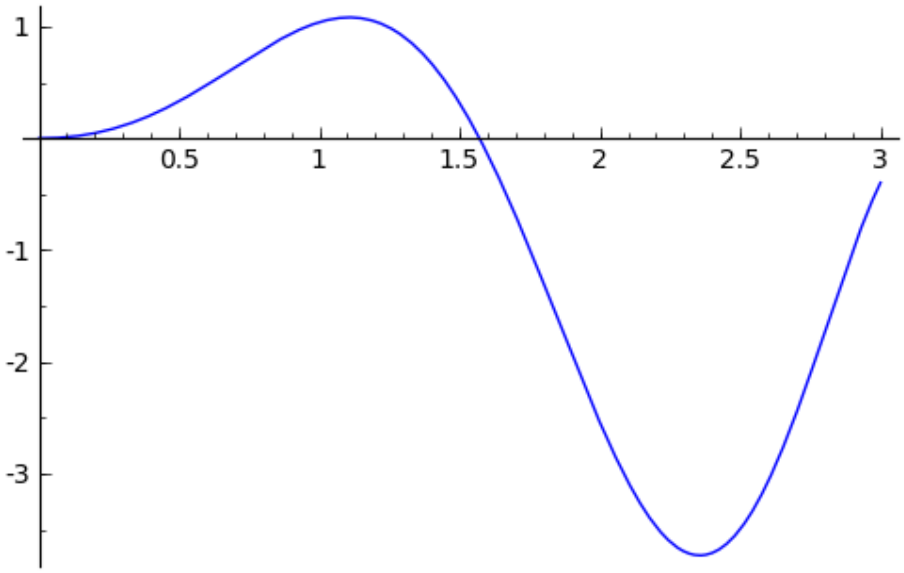
Calculus

Sage does **Calculus**:

```
f(x) = sin(x)^2*cos(x)*exp(x)
show(f)
```

$$x \mapsto e^{x} \sin\left(x\right)^{2} \cos\left(x\right)$$

```
plot(f, 0, 3)
```




```
show(integrate(f, x))
```

$$x \mapsto -\frac{3}{40} e^{x} \sin\left(3\,x\right) + \frac{1}{8} e^{x} \sin\left(x\right) - \frac{1}{40} e^{x} \cos\left(3\,x\right) + \frac{1}{8} e^{x} \cos\left(x\right)$$

```
```

```
```

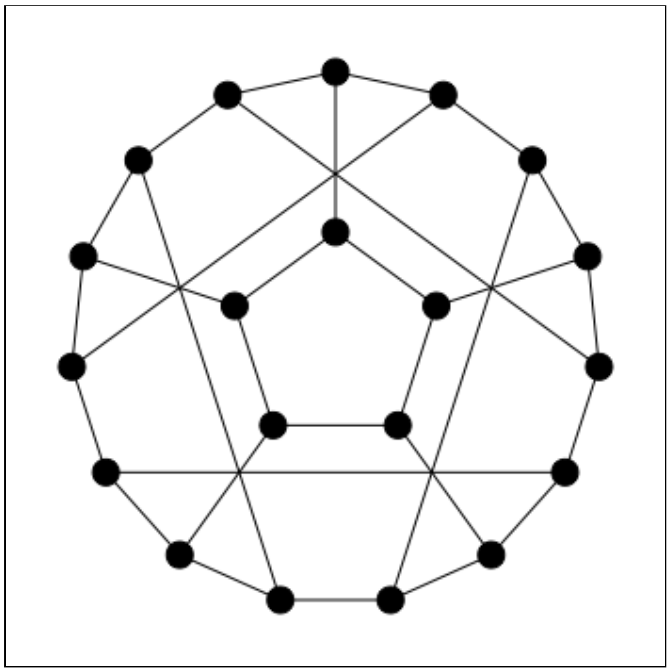
Graph Theory

Sage can do **graph theory**:

```
g = graphs.FlowerSnark(); g
```

```
    Flower Snark: Graph on 20 vertices
```

```
graph_editor(g)
```



Sage contains many ***unique and deep algorithms***:

```
g.automorphism_group()
```

```
    Permutation Group with generators
    [(2,11)(3,12)(4,13)(5,8)(6,9)(7,10)(16,19)(17,18),
    (1,4,7,10,13)(2,5,8,11,14)(3,6,9,12,20)(15,16,17,18,19),
    (1,14)(2,4)(5,7)(8,10)(11,13)]
```

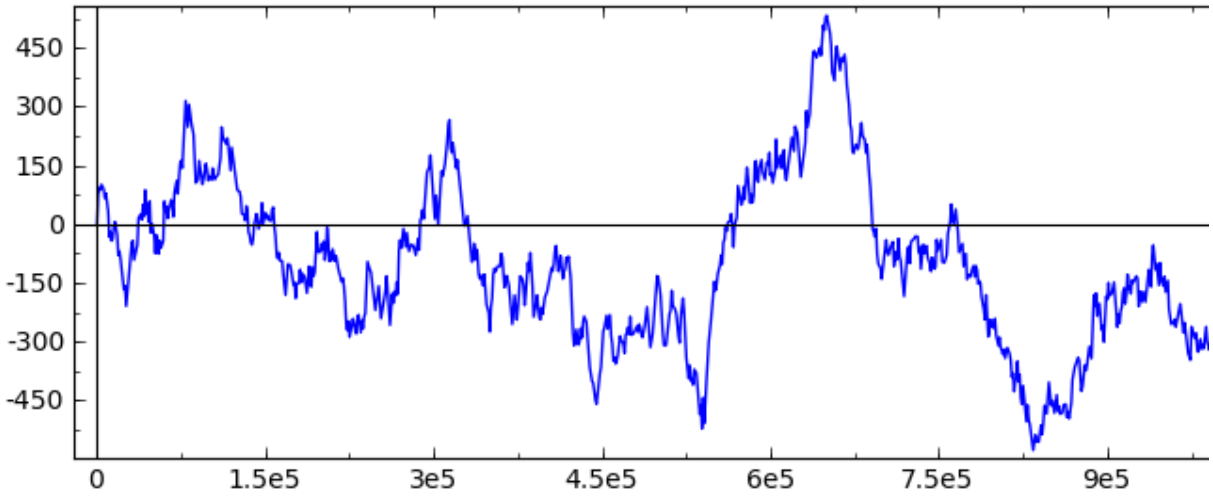
```

```

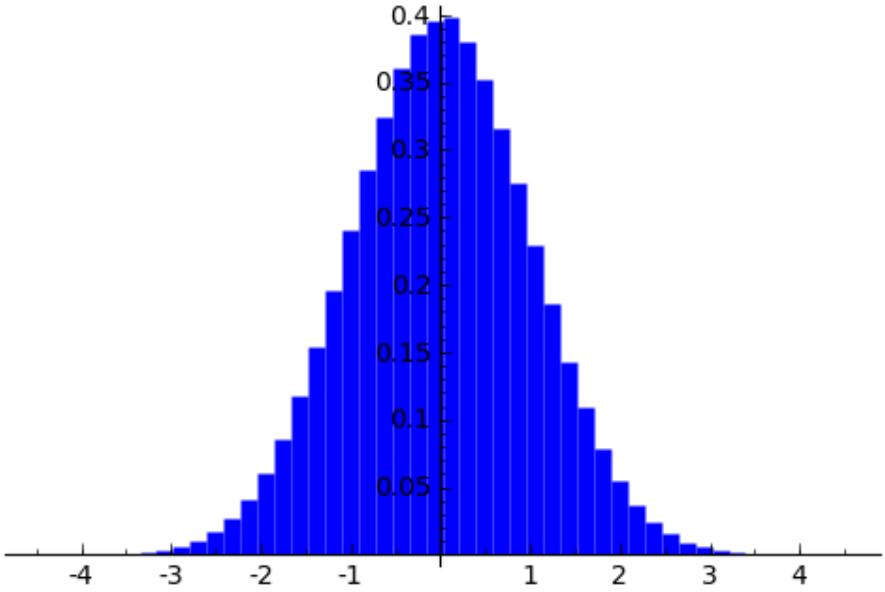
Statistics

Sage includes R, scipy.stats, and GSL (=Gnu Scientific Library).

```
t = stats.TimeSeries(10^6).randomize('normal')
plot(t.sums(), frame=True, figsize=[8,3])
```




```
t.plot_histogram()
```




```
mean(t)
```

```
-0.00033107089746577014
```

```
std(t)
```

```
1.0004268203066153
```

```
show(std([1,pi,e]))
```

$$\sqrt{\frac{1}{18}\left(\pi - 2\,e + 1\right)^2 + \frac{1}{18}\left(\pi + e - 2\right)^2 + \frac{1}{18}\left(2\,\pi - e - 1\right)^2}$$

```

```

Matrices

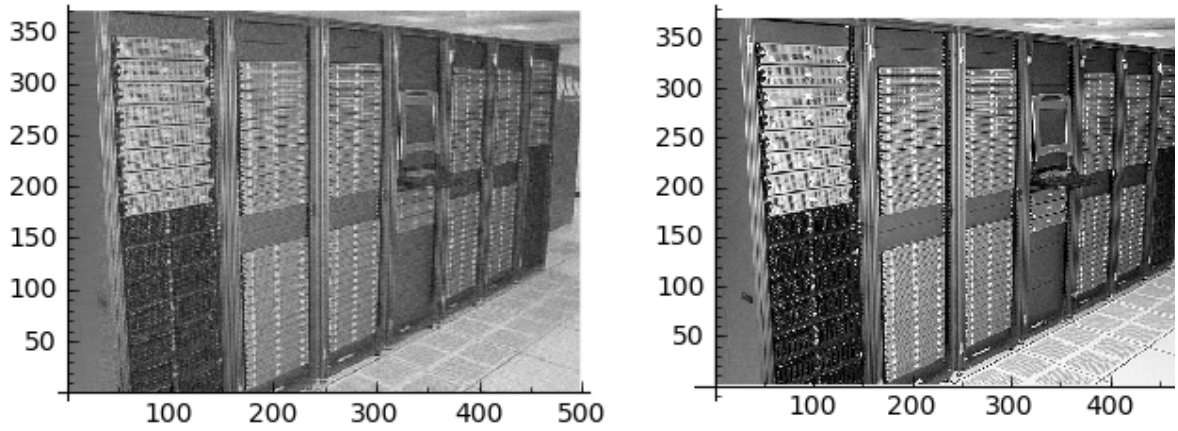
And **numerical linear algebra**:

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'lonestar.png'), 2)
@interact
def svd_image(i = ("Eigenvalues (quality)",(20,(1..100))),
              display_axes = ("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>Lonestar: compressed using %s eigenvalues</h2>'%i)
```

Eigenvalues (quality) 20

Display Axes

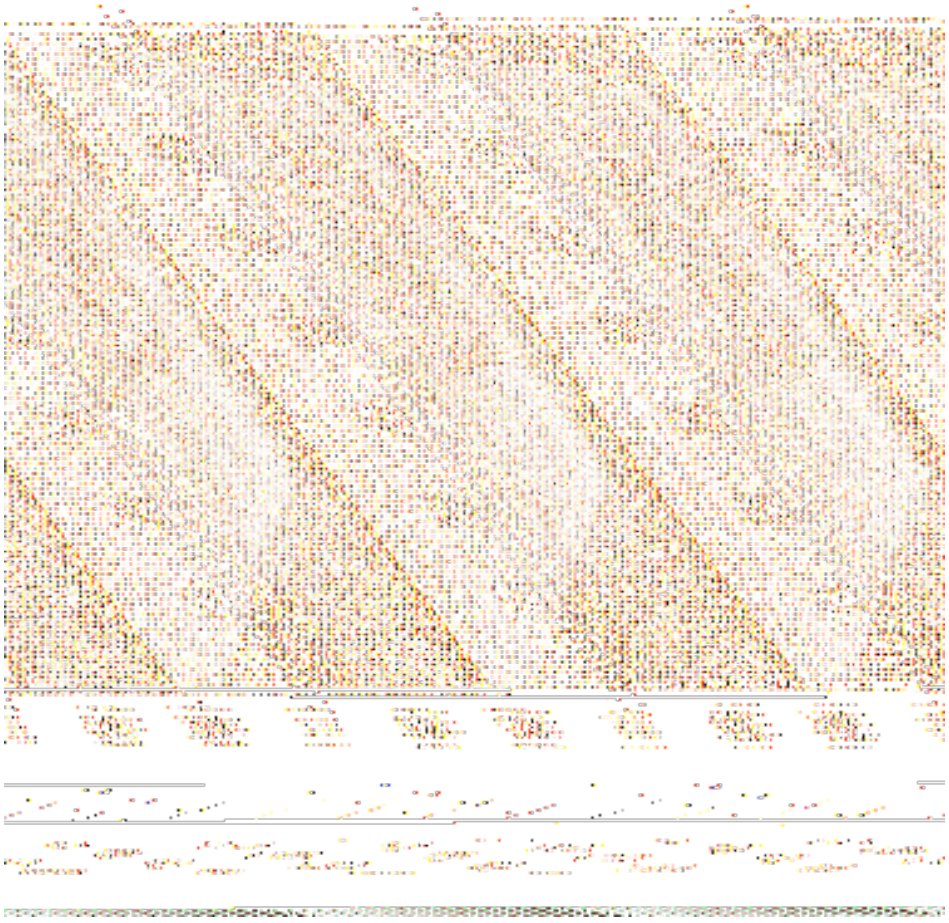
Lonestar: compressed using 91 eigenvalues



3-Dimensional Plots

Sage can **draw 3d plots**:

```
var('x,y')
b = 2.2
(plot3d(sin(x^2-y^2),(x,-b,b),(y,-b,b), opacity=.9) +
    plot3d(0, (x,-b,b), (y,-b,b), color='red'))
```

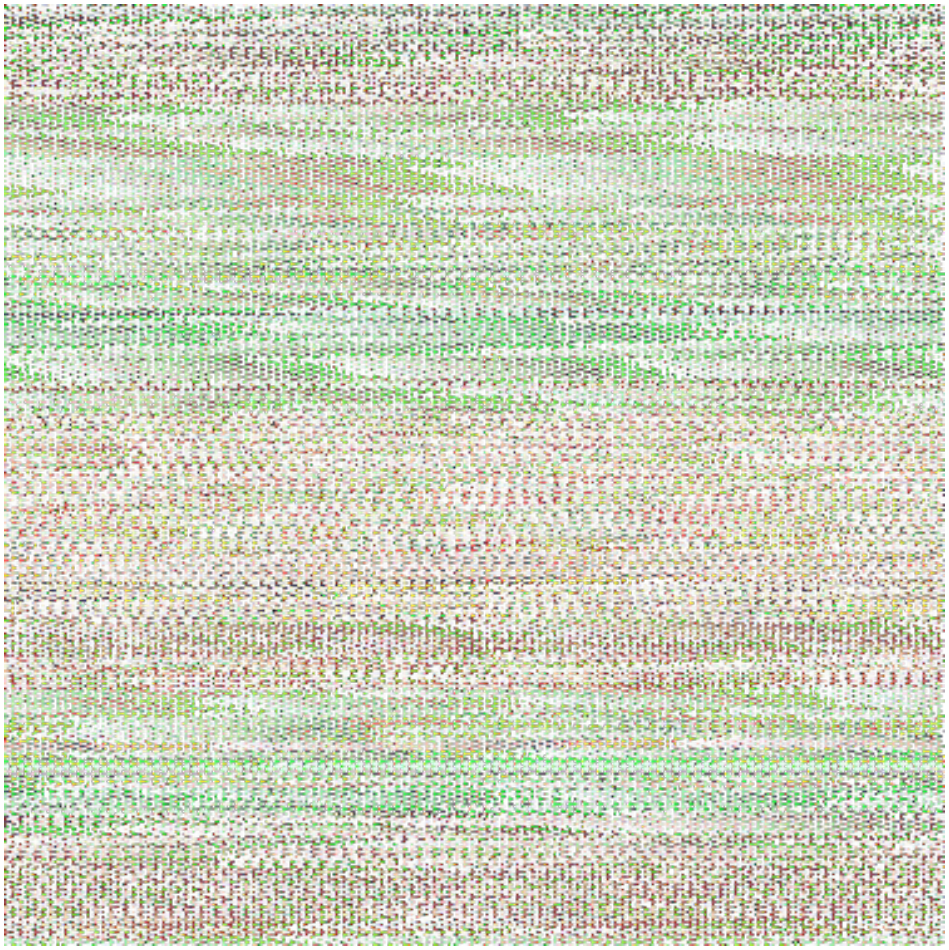


Get Image

```
var('x,y,z')
f = y^2*z^2-x^2*y^3-x*z^3+x^3*y*z
show(f)
```

$$x^3yz - x^2y^3 - xz^3 + y^2z^2$$

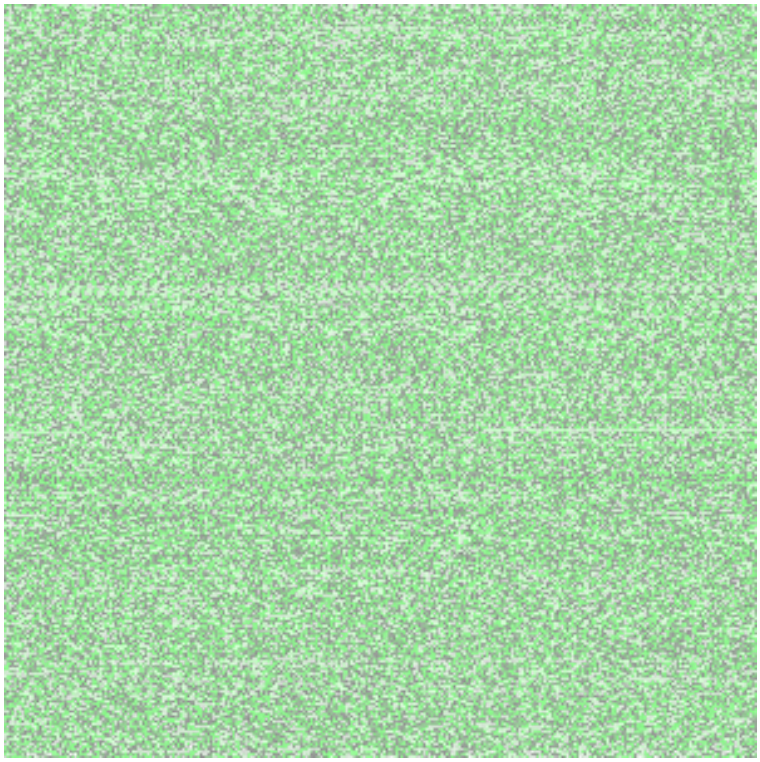
```
h=1; implicit_plot3d(f, (x,-h,h), (y,-h,h), (z,-h,h),
                plot_points=50, opacity=0.7, color='purple')
```



Get Image

Sage can plot Yoda:

```
from scipy import io
x = io.loadmat(DATA + 'yodapose.mat', struct_as_record=True)
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 = Color('#00aa00')) +
    IndexFaceSet(F4, V, color = Color('#00aa00')))
Y = Y.rotateX(-1)
Y.show(aspect_ratio = [1,1,1], frame = False, figsize = 4)
```



Get Image

And if you're really serious, Sage ***is*** Python, so you can also use:

- **Mayavi:** Python interface to VTK
- **Chaco:** interact 2d graphics system developed at Enthought (local Austin company)
- **VisIt**: https://wci.llnl.gov/codes/visit/manuals.html (python bindings)
- **ParaView:** http://www.cmake.org/Wiki/ParaView/Python_Scripting
- **VPython**: http://vpython.org/
- **PyGame:** http://www.pygame.org/news.html

Key take-away points:

- **FOSS:** 100% free open source software: good for clusters, sharing, research, collaboration
- **Python:** very mainstream, scientific-computing friendly programming language
- **Cython:** optimized Python to C/C++ compiler we develop with support for C/C++ datatypes
- **Interfaces:** to control Matlab, Octave, Mathematica, etc., from Sage
- **Self contained:** can have many copies of Sage at once, so no "dependency hell"
- **Peer reviewed:** get your code refereed
- **Web based:** notebook interface

Questions?

Now, take a break, then start with the tutorials...