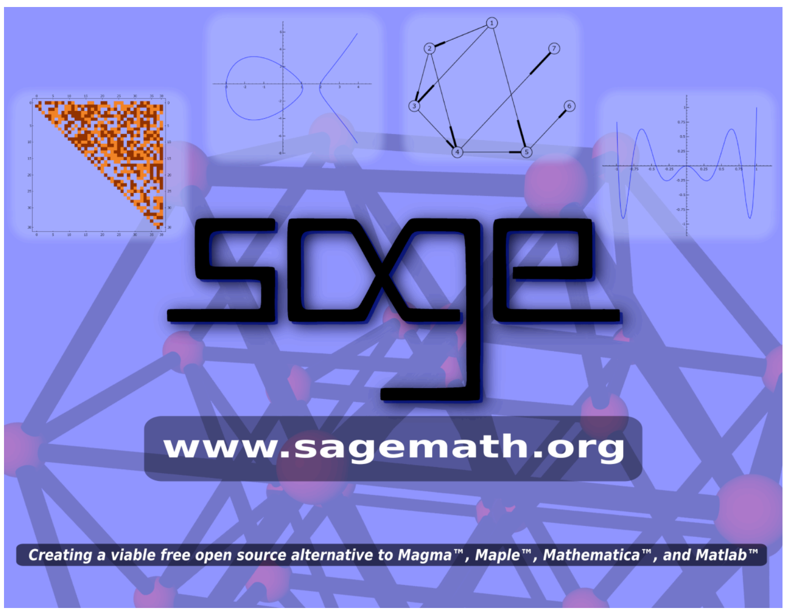
SeaPIG talk -- 20090131

SAGE

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

William Stein, Associate Professor, University of Washington



History

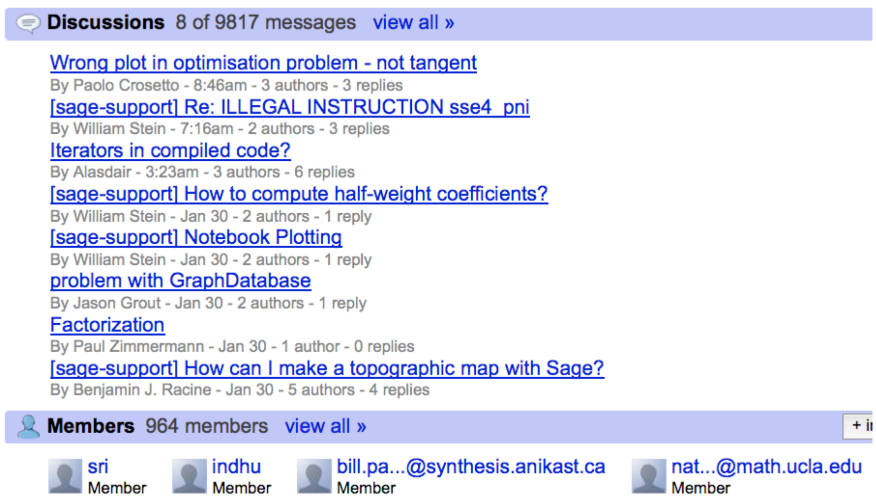
- *I started Sage* at Harvard in *January 2005*.
- IMHO, no existing math software is good enough (*free or commercial*).
- Sage-1.0 released *February 2006* at Sage Days 1 (San Diego).
- *Sage Days Workshops* 1, 2, ..., 11, at UCLA, UW, Cambridge, Bristol, Austin, France, San Diego, ***Seattle***, etc.

- Funding from **UW, NSF, DoD, Microsoft, Google, Sun**, private donors, etc.

Sage is 100% Free and Open Source Software

Active user community; **964** members of the sage-support mailing list.

Free webapp -- **sagenb.org** -- has about 3000 users (and there are other servers at universities around the world..)


Discussions 8 of 9817 messages view all »

Wrong plot in optimisation problem - not tangent
By Paolo Crosetto - 8:46am - 3 authors - 3 replies
[sage-support] Re: ILLEGAL INSTRUCTION sse4_pni
By William Stein - 7:16am - 2 authors - 3 replies
Iterators in compiled code?
By Alasdair - 3:23am - 3 authors - 6 replies
[sage-support] How to compute half-weight coefficients?
By William Stein - Jan 30 - 2 authors - 1 reply
[sage-support] Notebook Plotting
By William Stein - Jan 30 - 2 authors - 1 reply
problem with GraphDatabase
By Jason Grout - Jan 30 - 2 authors - 1 reply
Factorization
By Paul Zimmermann - Jan 30 - 1 author - 0 replies
[sage-support] How can I make a topographic map with Sage?
By Benjamin J. Racine - Jan 30 - 5 authors - 4 replies

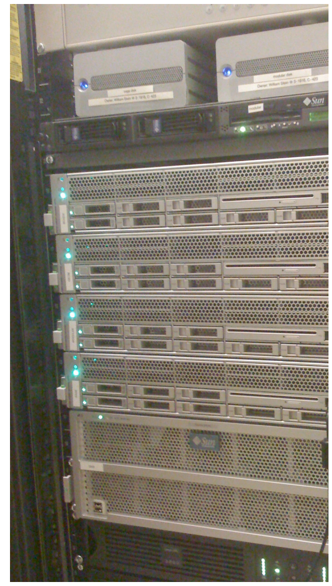
Members 964 members view all »

sri Member
indhu Member
bill.pa...@synthesis.anikast.ca Member
nat...@math.ucla.edu Member

---

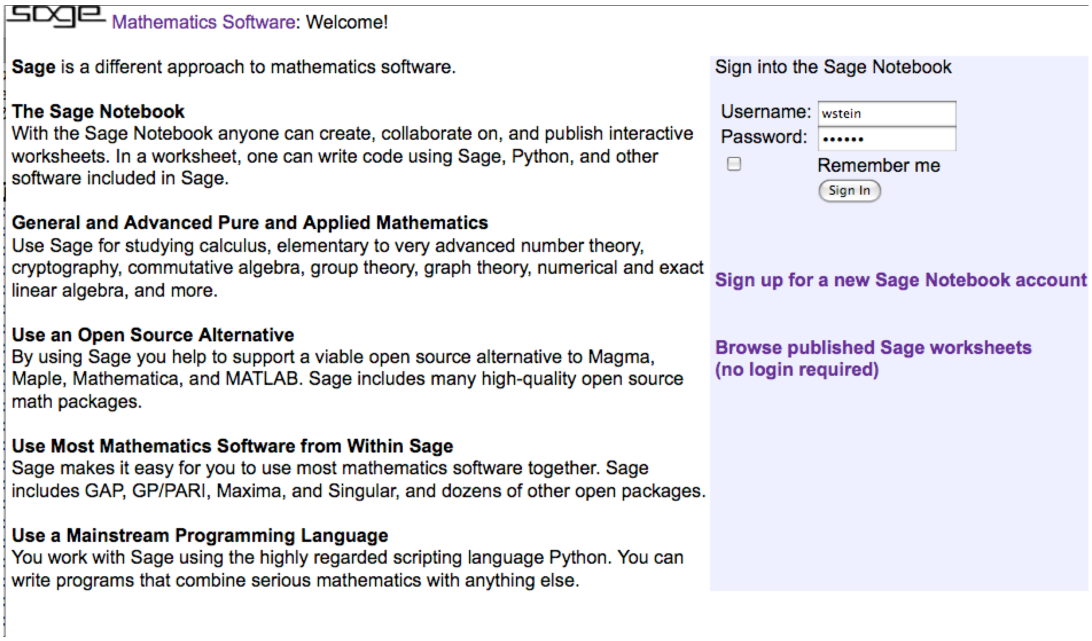
sagemath.org


sage
open source mathematics software · v3.2.3 (2009-01-08)
RSS · Blog · Trac · Wiki · Search:
Sage online · Milnix.org · KAIST · Download Sage
Intro About Help Download Search Development Links
Sage is a free open-source mathematics software system licensed under the GPL. It combines the power of many existing open-source packages into a common Python-based interface.
Mission: Creating a viable free open source alternative to Magma, Maple, Mathematica and Matlab.
Donate · Acknowledgments · Browse the Code · Questions?
Download 3.2.3
Binary · Source · Packages
Sage Via the Web
Milnix.org · KAIST
Help
Documentation · Support · Tutorial
Feature Tour
Quickstart · Research · Education
Library
Testimonials · Books · Publications
Search
Random Link: Quickstart into Sage
Sign In



---

sagenb.org



Sage = Python + Local Web Interface + Tons of Work

The first example from the Python tutorial:

```
the_world_is_flat = 1
if the_world_is_flat:
  print "Be careful not to fall off!"
```

```
Be careful not to fall off!
```

The Sage preparser (%python turns it off temporarily):

```
%python
3/5 + 2/3 + 1/3
```

```
0
```

```
3/5 + 2/3 + 1/3
```

```
8/5
```

```
preparse('3/5 + 2/3 + 1/3')
```

```
'Integer(3)/Integer(5) + Integer(2)/Integer(3) +
Integer(1)/Integer(3)'
```

Symbolic expressions:

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

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

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

```
x^y + pi + sqrt(2) + 5/3
```

```
show(a)
```

$x^{y} + \pi + \sqrt{2} + \frac{5}{3}$

What is Sage?

- Well over *300,000 lines* of new Python/Cython code
- *A Distribution* of mathematical software (over 60 third-party packages); builds from source without dependency (over 5 million lines of code)
- *Exact and numerical linear algebra*, optimization (*numpy, scipy, R, and gsl* all **included**)
- Group theory, *number theory*, combinatorics, graph theory
- Symbolic *calculus*
- Coding theory, *cryptography* and cryptanalysis
- 2d and 3d *plotting*
- *Statistics* (Sage includes R)
- Overall range of *functionality rivals* that of Maple, Matlab, and Mathematica
- Sage is ***frickin' huge***

Reference Manual (over 3000 pages, all new)

Example: An Integer Determinant

```
a = random_matrix(ZZ,200)
print a[0]
time d = a.determinant(); d
```

```
(1, 1, 0, 0, -1, 1, 2, 1, 2, 0, 1, 0, -1, 1, 1, 2, -1, 0, -1, 1,
-16, 0, -2, 1, 0, 6, 1, 4, -1, 3, 27, 1, -1, 0, 6, 1, 1, 4, 1, -1,
1, 0, 0, -1, -29, 4, -2, 1, 2, -1, -1, -1, 3, 3, 0, -1, 1, 6, 1, 1,
6, 6, 1, 21, 0, 11, -7, 2, -1, 1, -1, 2, 1, 2, 0, 0, -4, -13, -1, 1,
1, 2, 6, -6, 0, -2, -1, 2, 2, -1, 0, 2, -1, -3, 3, -1, 1, 1, -8, -1,
-1, 1, 10, -1, 1, 1, -22, 0, -6, -2, -1, -2, -2, 2, 1, 1, -2, -1,
-1, -1, -1, 0, -9, 2, 1, 1, 1, 2, -2, 0, -2, 0, 1, -1, 11, -1, -2,
3, 1, 1, 63, -93, -2, 0, -11, 1, -7, -1, 5, 2, -41, 3, 2, 3, 1, 1,
0, -1, 0, 0, -2, 0, -1, 1, -4, 1, -5, 5, 1, 0, -11, -1, 1, -1, -1,
2, -1, 0, -3, 1, -1, 2, -1, -263, 1, -1, -9, 1, -2, 0, -189, 3, -1,
0, 0, 9, 1, 1, -5, 1)
776202539430700979639122404230900551465828150681295623078860880096 80\
674796509730802411143904875441043332111663406380408161978989764987 67\
644536680416275127598583370904889281457205854173143126657832407977 30\
186585355860136581872347588955471801118678315977027963154480326691 25\
491789060269216192026359520309398079473857586502893864648273675879 99\
```

```
15903082082090053138222324019794480340114468887728350216313449573010\
9397200563570831479911910377761727280293793558438
Time: CPU 0.29 s, Wall: 0.31 s
```

```
maple.with_package('linalg')
B = maple(a)
t = maple.cputime()
time c = B.det()
maple.cputime(t)
```

```
Time: CPU 0.00 s, Wall: 29.30 s
26.890999999999998
```

```
c == d
```

```
True
```

"The speedup of LinBox [what Sage incorporates] against LinearAlgebra Maple's module is tremendous. It allows for instance the computation of an integer determinant of 400x400 dense matrix with entries lying in [1..100] in only 1.7s on PIV-3.2Ghz while Maple computation turns out to need 536s." (from [linalg.org)](linalg.org)

```
a = random_matrix(ZZ, 400, x=1,y=100)
time d = a.det()
```

```
Time: CPU 2.26 s, Wall: 2.26 s
```

Example: A Symbolic Expression

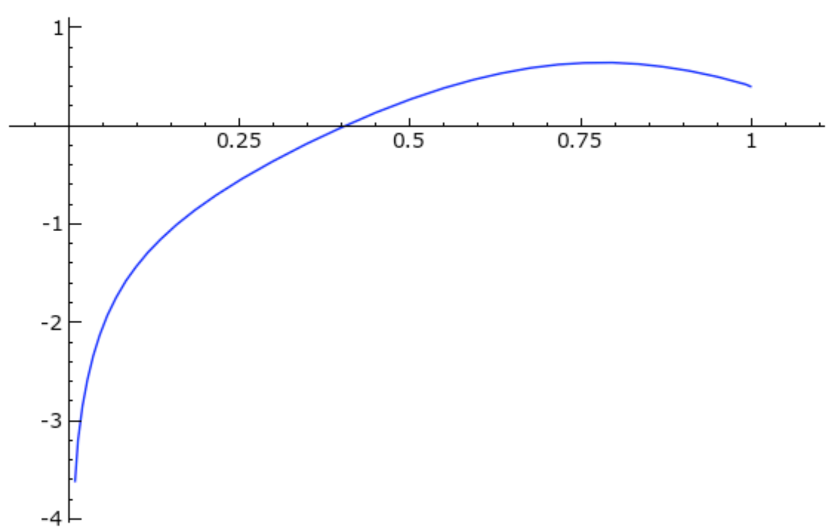
```
x = var('x')
```

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

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

Plotting functions has same syntax as Mathematica:

```
plot(f,(0.01,1))
```



_fast_float_ yields super-fast evaluation of Sage symbolic expressions -- e.g., here it is 10 times faster than native Python!

```
g = f._fast_float_(x)
timeit('g(4.5r)')
```

```
625 loops, best of 3: 515 ns per loop
```

```
%python
# %python, so no preparsing so uses pure python
import math
def g(x): return math.sin(3*x)*x + log(x) + 1/(1+x)**2
```

```
timeit('g(4.5r)')
```

```
625 loops, best of 3: 7.03 Âµs per loop
```

Example: Compare Answers with Maple

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

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

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()
```

```
24038
```

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

```
24038 s007  Ss+    0:00.01 /bin/sh /Users/wstein/bin/maple -t
24233 s015  S+     0:00.00 sh -c ps ax |grep 24038
24235 s015  R+     0:00.00 grep 24038
0
```

Use Maple objects via a Pythonic notation:

```
show(m.integrate('x'))
```

$$1/9\,\sin\left(3\,x\right) - 1/3\,\cos\left(3\,x\right)x + \ln\left(x\right)x - x - \left(x+1\right)^{-1}$$

```
mathematica(f).Integrate(x)
```

```
    -x - (1 + x)^(-1) - (x*Cos[3*x])/3 + x*Log[x] + Sin[3*x]/9
```

Example: Interactive Image Compression

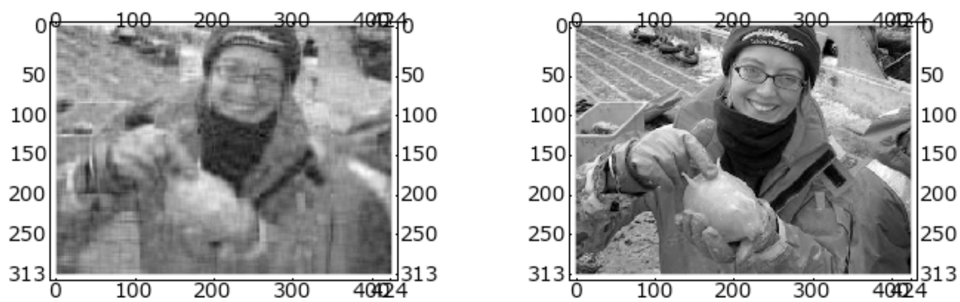
This illustrates pylab (matplotlib + numpy), Sage plotting, html output, and @interact.

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'seapig.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=(7,2))
   html('<h2>A Marine Biologist holding a SEAPIG,<br>compressed using %s
eigenvalues</h2>'%i)
```

i

display_axes

A Marine Biologist holding a SEAPIG,
compressed using 20 eigenvalues



Example: Python Class Heierarchy

```
def class_hierarchy(cls, v):
   v.append(str(cls))
   for supercls in cls.__bases__:
       class_hierarchy(supercls, v)
@interact
def foo(object=1):
   print object
   print '<html><h2>Inheritance hierarchy of\n%r</h2>'%(
        str(type(object)).replace('<','').replace('>',''))
   print '<font color="#333333"><pre>'
   v = []; class_hierarchy(object.__class__, v)
```

```
    print '\n'.join(['.'*(3*i)+w for i, w in
              enumerate(reversed(v))]).replace('<','').replace('>','')
    print '</pre></font></html>'
```

object 1

1

Inheritance hierarchy of "type 'sage.rings.integer.Integer'"

```
type 'object'
...type 'sage.structure.sage_object.SageObject'
......type 'sage.structure.element.Element'
.........type 'sage.structure.element.ModuleElement'
............type 'sage.structure.element.RingElement'
...............type 'sage.structure.element.CommutativeRingElement'
..................type 'sage.structure.element.IntegralDomainElement'
.....................type 'sage.structure.element.DedekindDomainElement'
........................type 'sage.structure.element.PrincipalIdealDomainElement'
...........................type 'sage.structure.element.EuclideanDomainElement'
..............................type 'sage.rings.integer.Integer'
```

Example: 3d Plots

```
var('x,y')
plot3d(sin(x*y^2), (x,-2,2), (y,-2,2))
```

```
%hide
var('u,v')
plots = ['Two Interlinked Tori', 'Star of David', 'Double Heart',
         'Heart', 'Green bowtie', "Boy's Surface", "Maeder's Owl",
         'Cross cap']

@interact
def _(example=selector(plots, buttons=True, nrows=2),
      tachyon=("Raytrace", False), frame = ('Frame', False),
      opacity=(1,(0.1,1))):
   url = ''
   if example == 'Two Interlinked Tori':
       f1 = (4+(3+cos(v))*sin(u), 4+(3+cos(v))*cos(u), 4+sin(v))
       f2 = (8+(3+cos(v))*cos(u), 3+sin(v), 4+(3+cos(v))*sin(u))
       p1 = parametric_plot3d(f1, (u,0,2*pi), (v,0,2*pi), color="red", opacity=opacity)
       p2 = parametric_plot3d(f2, (u,0,2*pi), (v,0,2*pi), color="blue",opacity=opacity)
       P = p1 + p2
   elif example == 'Star of David':
       f_x = cos(u)*cos(v)*(abs(cos(3*v/4))^500 + abs(sin(3*v/4))^500)^(-1/260)*
(abs(cos(4*u/4))^200 + abs(sin(4*u/4))^200)^(-1/200)
       f_y = cos(u)*sin(v)*(abs(cos(3*v/4))^500 + abs(sin(3*v/4))^500)^(-1/260)*
(abs(cos(4*u/4))^200 + abs(sin(4*u/4))^200)^(-1/200)
       f_z = sin(u)*(abs(cos(4*u/4))^200 + abs(sin(4*u/4))^200)^(-1/200)
       P = parametric_plot3d([f_x, f_y, f_z], (u, -pi, pi), (v, 0, 2*pi),opacity=opacity)
   elif example == 'Double Heart':
       f_x = ( abs(v) - abs(u) - abs(tanh((1/sqrt(2))*u)/(1/sqrt(2))) +
abs(tanh((1/sqrt(2))*v)/(1/sqrt(2))) )*sin(v)
       f_y = ( abs(v) - abs(u) - abs(tanh((1/sqrt(2))*u)/(1/sqrt(2))) -
abs(tanh((1/sqrt(2))*v)/(1/sqrt(2))) )*cos(v)
```

```
        f_z = sin(u)*(abs(cos(4*u/4))^1 + abs(sin(4*u/4))^1)^(-1/1)
        P = parametric_plot3d([f_x, f_y, f_z], (u, 0, pi), (v, -pi, pi),opacity=opacity)
    elif example == 'Heart':
        f_x = cos(u)*(4*sqrt(1-v^2)*sin(abs(u))^abs(u))
        f_y = sin(u) *(4*sqrt(1-v^2)*sin(abs(u))^abs(u))
        f_z = v
        P = parametric_plot3d([f_x, f_y, f_z], (u, -pi, pi), (v, -1, 1), frame=False,
color="red",opacity=opacity)
    elif example == 'Green bowtie':
        f_x = sin(u) / (sqrt(2) + sin(v))
        f_y = sin(u) / (sqrt(2) + cos(v))
        f_z = cos(u) / (1 + sqrt(2))
        P = parametric_plot3d([f_x, f_y, f_z], (u, -pi, pi), (v, -pi, pi), frame=False,
color="green",opacity=opacity)
    elif example == "Boy's Surface":
        url = "http://en.wikipedia.org/wiki/Boy's_surface"
        fx = 2/3* (cos(u)* cos(2*v) + sqrt(2)* sin(u)* cos(v))* cos(u) / (sqrt(2) -
sin(2*u)* sin(3*v))
        fy = 2/3* (cos(u)* sin(2*v) - sqrt(2)* sin(u)* sin(v))* cos(u) / (sqrt(2) -
sin(2*u)* sin(3*v))
        fz = sqrt(2)* cos(u)* cos(u) / (sqrt(2) - sin(2*u)* sin(3*v))
        P = parametric_plot3d([fx, fy, fz], (u, -2*pi, 2*pi), (v, 0, pi), plot_points =
[90,90], frame=False, color="orange",opacity=opacity)
    elif example == "Maeder's Owl":
        fx = v *cos(u) - 0.5* v^2 * cos(2* u)
        fy = -v *sin(u) - 0.5* v^2 * sin(2* u)
        fz = 4 *v^1.5 * cos(3 *u / 2) / 3
        P = parametric_plot3d([fx, fy, fz], (u, -2*pi, 2*pi), (v, 0, 1),plot_points =
[90,90], frame=False, color="purple",opacity=opacity)
    elif example =='Cross cap':
        url = 'http://en.wikipedia.org/wiki/Cross-cap'
        fx = (1+cos(v))*cos(u)
        fy = (1+cos(v))*sin(u)
        fz = -tanh((2/3)*(u-pi))*sin(v)
        P = parametric_plot3d([fx, fy, fz], (u, 0, 2*pi), (v, 0, 2*pi), frame=False,
color="red",opacity=opacity)
    else:
        print "Bug selecting plot?"
        return


    html('<h2>%s</h2>'%example)
    if url:
        html('<h3><a target="_new" href="%s">%s</a></h3>'%(url,url))
    show(P, viewer='tachyon' if tachyon else 'jmol', frame=frame)
```

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..."
```

Questions?