MAA talk on Sage - 20min

MAA Talk on Sage

William Stein

February 26, 2011, Santa Rosa

Factor

Factoring an integer:

```
factor(2012)
```

$2^2 \cdot 503$

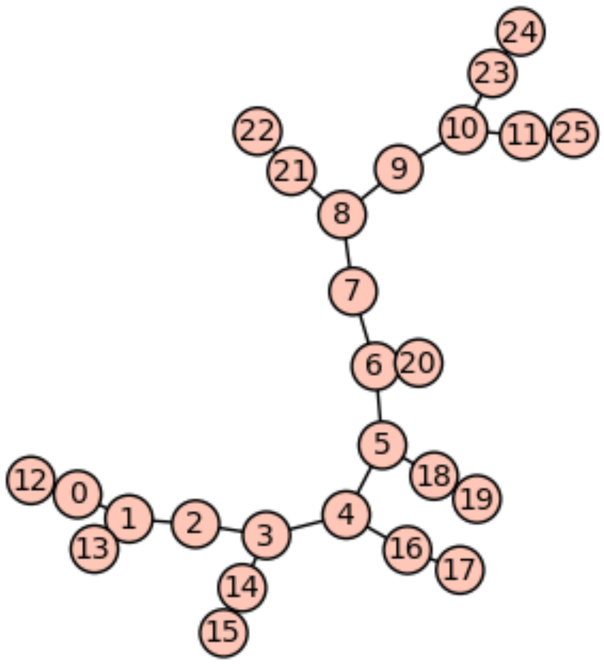
Factoring a symbolic expression:

```
var('x,y')
factor(x^8 - y^2*e^(2*x))
```

$(x^4 - ye^x)(x^4 + ye^x)$

Graph Theory

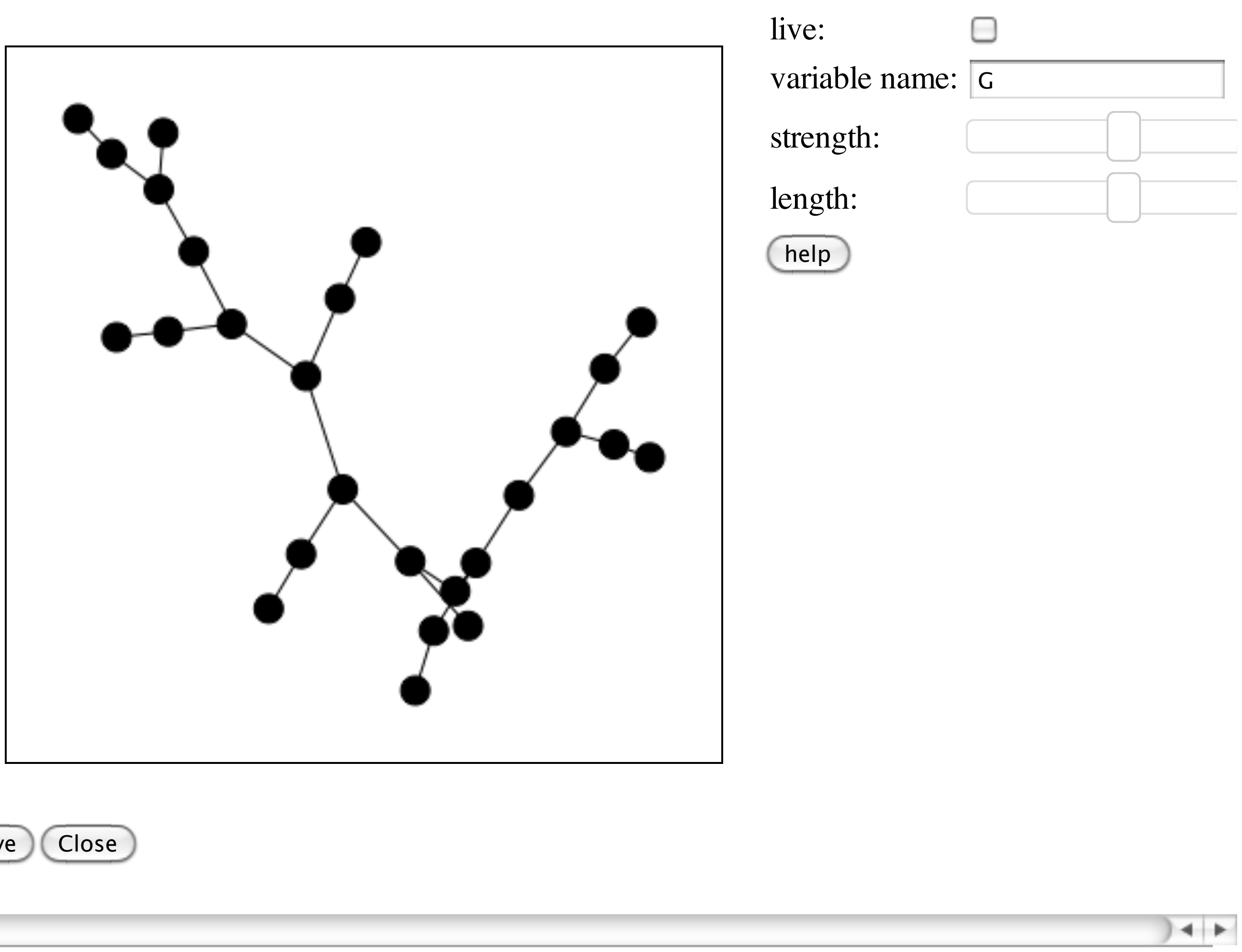
```
set_random_seed(1); G = graphs.RandomLobster(7, .6, .3); show(G)
```



```
G.automorphism_group()
```

$\langle (11, 23)(24, 25) \rangle$

```
graph_editor(G)
```



```
G.is_planar()
```

True

Symbolic Integrals

```
integrate(sin(x)*tan(x), x)
```

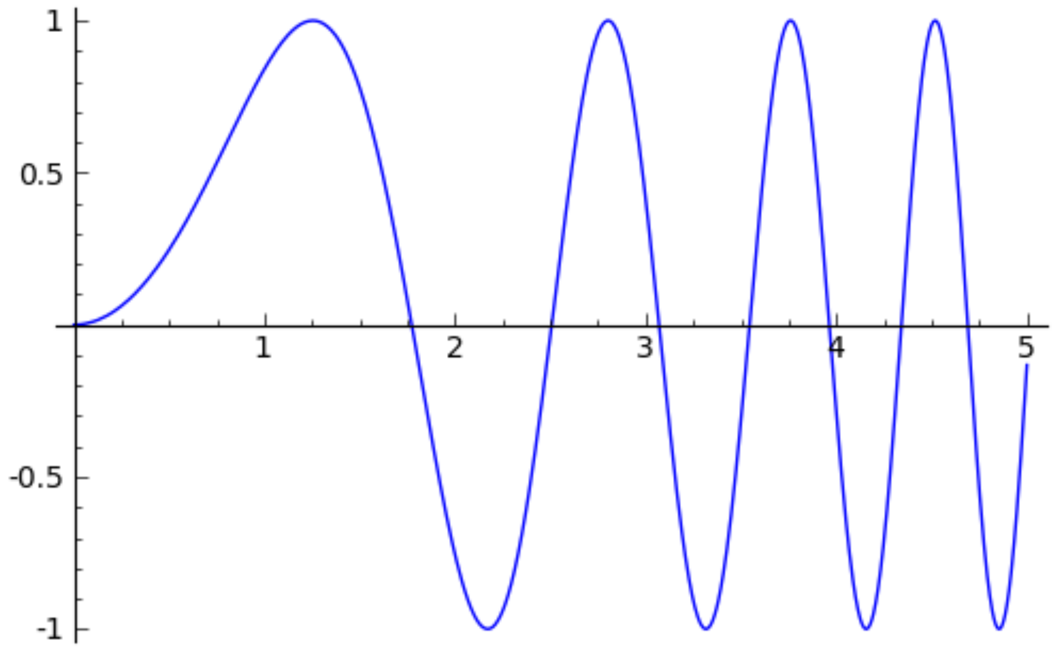
$-\frac{1}{2}\log\left(\sin\left(x\right) - 1\right) + \frac{1}{2}\log\left(\sin\left(x\right) + 1\right) - \sin\left(x\right)$

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

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

Plotting Functions

```
plot(sin(x^2), (x,0,5))
```

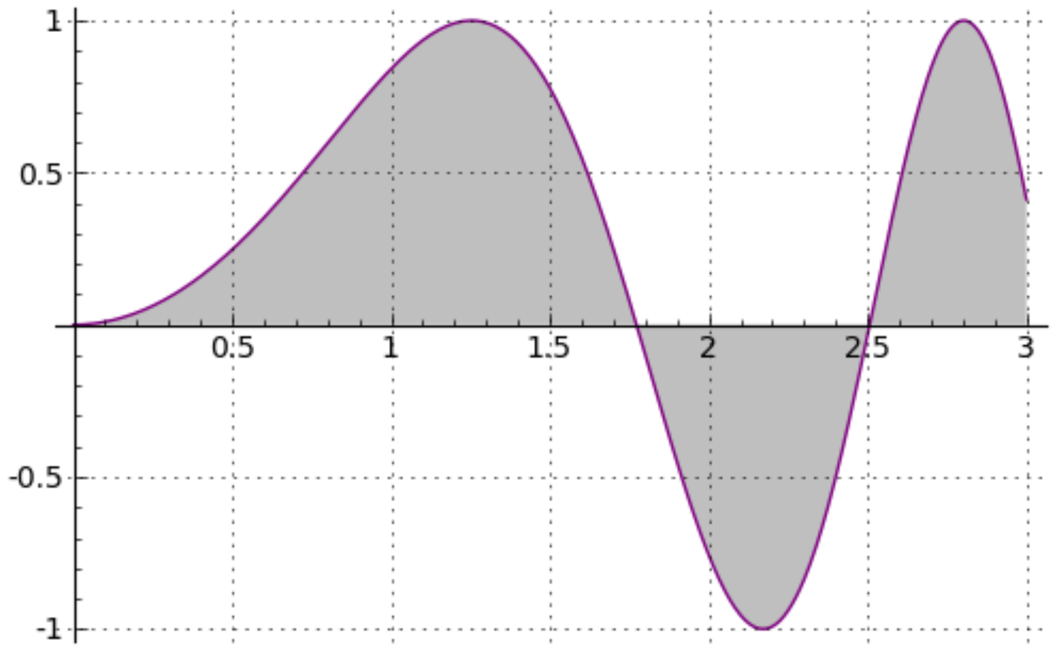


```
var('x')
@interact
def h(f=sin(x^2), grid=True, t=(1..20)):
    show(plot(f, (x, 0, 3),
                    thickness=t, color='purple', fill=True,
gridlines=grid))
```

f sin(x^2)

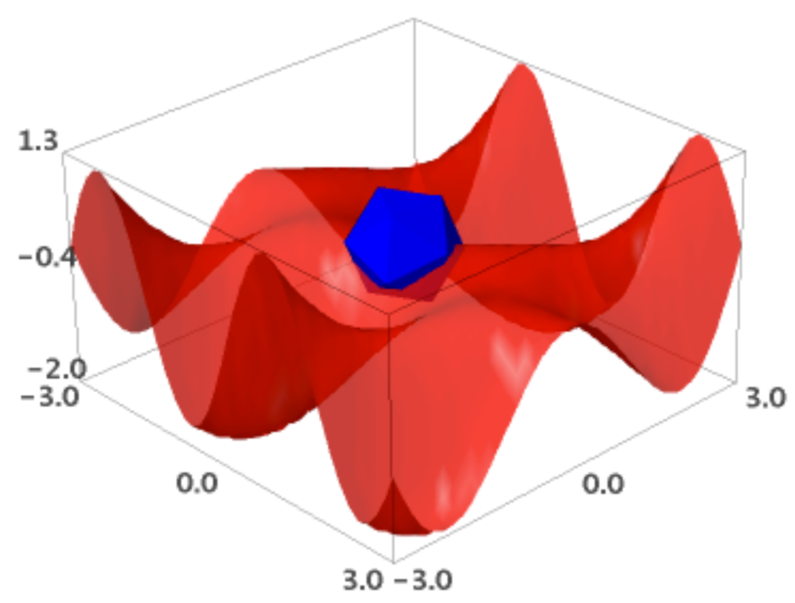
grid ✓

t 1



Plotting a Function in 3D

```
f(x,y) = sin(x - y)*y*cos(x)
plot3d(f, (x,-3,3), (y,-3,3), opacity=.7, color='red') +
icosahedron(color='blue')
```



[Wake this 3-D view]
[Click to Sleep this 3-D view]
[Help for Jmol 3-D viewer]

Applet

Functions

State

3-D display size:

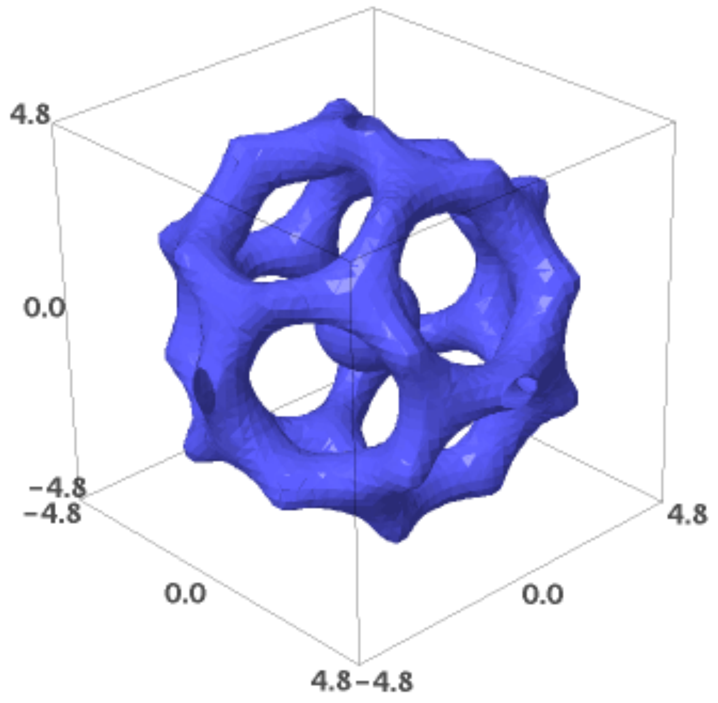
Medium (400px)

- [Arbitrarily resizable in own window]
- [Get static image to save]

☐ Spin on

An implicit 3D plot:

```
T = RDF(golden_ratio)
var('x,y,z')
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, (x, -r, r), (y, -r, r), (z, -r, r),
plot_points=40)
```



Wake this 3-D view
Click to Sleep this 3-D view
Help for Jmol 3-D viewer

Applet

Functions

State

3-D display size:

Medium (400px)

- Arbitrarily resizable in own window
- Get static image to save

☐ Spin on

Image Compression

```
import pylab; import numpy
A_image = numpy.mean(pylab.imread(DATA + 'santarosa.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]//2)))):
    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=(10,6))
    html("Compressed to %.1f%% of size using %s eigenvalues."%(
                        100*(2.0*i*n+i)/(n*n), i))
```

Eigenvalues (quality) 20

```
Compressed to 14.1% of size using 20 eigenvalues.
```