SACNAS

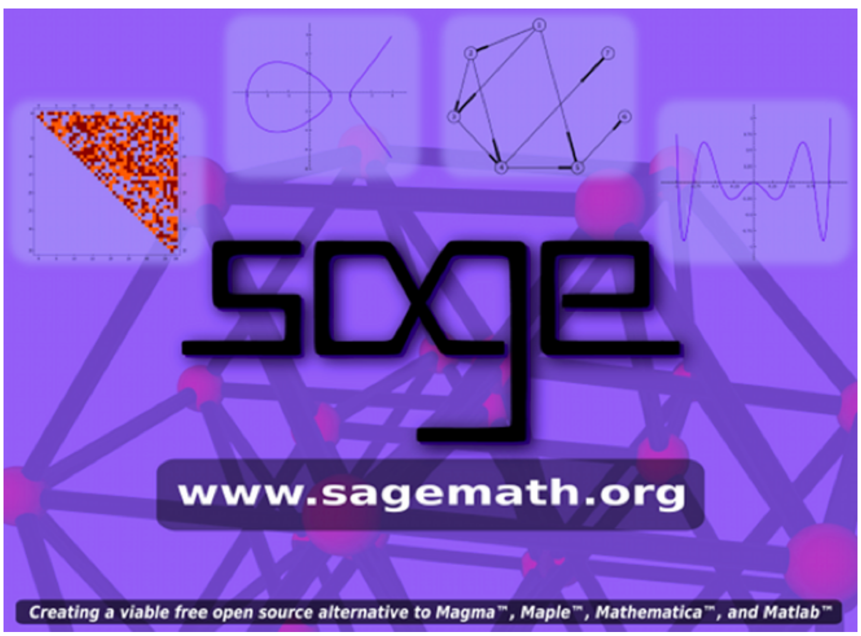
```
1+2+3+4 # testing
```

    10

Sage: Creating a Viable Open Source Alternative to Magma, Maple, Matlab, and Mathematica

SACNAS 2012, October 10, 2012

William Stein (Professor of Mathematics, University of Washington)



Sage: Mission Statement

"Create A Viable Free Open Source Alternative to Magma, Maple, Mathematica, and Matlab"

- **Mathematical features:** Of Magma, Maple, Mathematica, and Matlab, with comparable speed
- **Graphics:** 2d and 3d
- **Notebook:** Interactive graphical user interface
- **Documentation:** Books, papers, curriculum, etc.

Sage is not a drop-in replacement: does not run programs written in the custom languages of the Ma's.

Sage is not like Octave (versus Matlab).

Sage's culture, architecture, programming language, and feel are different than the Ma's.

Why *not* Magma, Maple, Matlab, Mathematica?

1. **Commercial:** Expensive for my collaborators and students. Not community owned.
2. **Closed:** Implementation of algorithms often secret
3. **Frustrating:** Tight control of development
4. **Copy protection:** Hard to run on supercomputer or my new laptop or after my 1-year license expires.
5. **Programming language:** All use a special math-only language
6. **Bugs:** Bug tracking is secret
7. **Compiler:** Lack of compilers for their math-only languages

1. **Python:** Sage uses a mainstream general purpose programming language (with a compiler: Cython)
2. **Distribution:** about 100 open source packages (**written by you** and your colleagues!)
3. **Interfaces:** smoothly tie together all these libraries and packages
4. **New library:** implements novel algorithms; over a half million lines; worldwide community of several hundred people.

```
4/6
```

    2/3

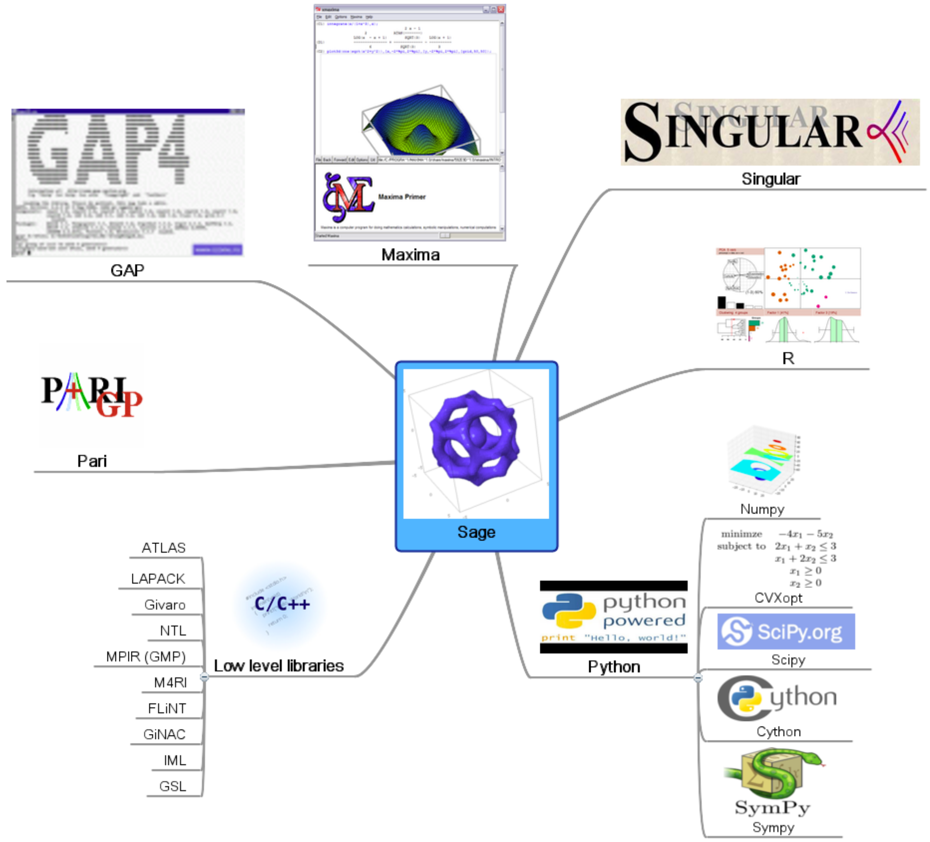
```
# Sage uses Python code

def foo(n, m):
    if n > m:
        print("%s is bigger!"%n)

foo(10, 5)
```

    10 is bigger!

```
foo('william', 'jon')
```

    william is bigger!

Distribution



... each with their own rich history!

Hundreds of Sage Developers

(There are at least 247 contributors in 167 different places from all around the world.)

William Stein, Tim Abbott, Michael Abshoff, Antti Ajanki, Martin Albrecht, Nick Alexander, Bill Allombert, Ethan Van Andel, Ivan Andrus, Pablo Angulo, Benjamin Antieau, André Apitzsch, Maite Aranes, Oscar Gerardo Lazo Arjona, Eviatar Bach, Jennifer Balakrishnan, Jason Bandlow, Gregory Bard, Sébastien Barthélemy, Rob Beezer, Karim Belabas, Arnaud Bergeron, Luis Berlioz, Erin Beyerstedt, Francois Bissey, Jonathan Bober, Tom Boothby, Nicolas Borie, Johan Bosman, Robert Bradshaw, Michael Brickenstein, Nils Bruin, André-Patrick Bubel, Stanislav Bulygin, Dan Bump, Iftikhar Burhanuddin, Paul Butler, Oriol Castejón, Ondrej Certik, Wilson Cheung, Dan Christensen, Craig Citro, Anders Claesson, Francis Clarke, Timothy Clemans, Alex Clemesha, Nathann Cohen, Jenny Cooley, John Cremona, Karl-Dieter Crisman, Fidel Barrera Cruz, Doug Cutrell, Alyson Deines, Vincent Delecroix, Jeroen Demeyer, Tom Denton, Maarten Derickx, Didier Deshommes, Ryan Dingman, Dan Drake, Tom Draper, Alexander Dreyer, Tim Dumol, Nathan Dunfield, Gabriel Ebner, Ben Edwards, Dana Ernst, Burcin Erocal, Ron Evans, Richard J. Fateman, Lars Fischer, Jean-Pierre Flori, Evan Fosmark, Laurent Fousse, Gary Furnish, Alex Ghitza, Andrzej Giniewicz, Alain Giorgetti, Samuele Giraudo, Amy Glen, Daniel Gordon, Chris Gorecki, Jan Groenewald, Rob Gross, Jason Grout, Ryan Grout, Mathieu Guay-Paquet, Alexey U. Gudchenko, Harold Gutch, Jonathan Gutow, Jose Guzman, Anna Haensch, Carlo Hamalainen, Marshall Hampton, Jon Hanke, David Møllerller Hansen, Mike Hansen, Bill Hart, David Harvey, Leif Hille, Florent Hivert, Ryan Hinton, Neal Holtz, Golam Mortuza Hossain, Sean Howe, Alexander Hupfer, Wilfried Huss, Hamish Ivey-Law, Naqi Jaffery, Peter Jeremy, Peter Jipsen, Fredrik Johansson, Niles Johnson, Timo Jolivet, Benjamin Jones, David Joyner, Michael Kallweit, Josh Kantor, Kiran Kedlaya, Lloyd Kilford, Simon King, Keshav Kini, David Kirkby, Emily Kirkman, David Kohel, Ted Kosan, Ross Kyprianou, Sébastien Labbé, Yann Laigle-Chapuy, Kwankyu Lee, Julien Leroy, Richard Lindner, David Loeffler, Miguel Marco, Michael Mardaus, Robert Mařík, Jason Martin, Alexandre Blondin Massé, Peter McNamara, Gregory McWhirter, Jason Merrill, Matthias Meulien, Robert Miller, Kate Minola, Moritz Minzlaff, Joel Mohler, Thierry Monteil, Peter Mora, Bobby Moretti, Rich Morin, Guillaume Moroz, Gregg Musiker, Tobias Nagel, Brett Nakashima, Pablo De Nápoli, Johan Sebastian Rosenkilde Nielsen, Minh Van Nguyen, Andrey Novoseltsev, Christopher Olah, Johan Oudinet, Bill Page, Ronan Paixão, Willem Jan Palenstijn, John Palmieri, Dmitrii Pasechnik, Javier López Peña, Paulo César Pereira de Andrade, David Perkinson, Clement Pernet, John Perry, Pearu Peterson, David Poetzsch-Heffter, Viviane Pons, Bill Purvis, Julien Puydt, Yi Qiang, Jordi Quer, Gustavo Rama, Jens Rasch, Martin Raum, Dorian Raymer, Stefan Reiterer, R. Rishikesh, David Roe, Bjarke Hammersholt Roune, Gordon Royle, Serge A. Salamanka, Franco Saliola, Leonardo Sampaio, Kyle Schalm, Ed Scheinerman, Anne Schilling, Harald Schilly, Jack Schmidt, Michael Schneider, Christopher Schwan, Dag Sverre Seljebotn, Dan Shumow, Denis Simone, Steven Sivek, Nils-Peter Skoruppa, Jaap Spies, Jonathan Spreer, Armin Straub, Marco Streng, Kevin Stueve, Christian Stump, Blair Sutton, Chris Swierczewski, Luis Felipe Tabera Alonso, Glenn Tarbox, Philippe Theveny, Nicolas Thiery, Griffen Thoma, Emmanuel Thomé, John Thurber, Igor Tolkov, Gonzalo Tornaria, Kiminori Tsukazaki, Charlie Turner, Michel Vandenbergh, Joris Vankerschaver, Soledad Villar, John Voight, Felipe Voloch, Steve Vonn, Justin Walker, Mark Watkins, Georg S. Weber, Eric Webster, Ralf-Philipp Weinmann, Joe Wetherell, Carl Witty, Cristian Wuthrich, Soroosh Yazdani, Dal S. Yu, Gary Zablackis, Mike Zabrocki, Bin Zhang, Paul Zimmermann, Mao Ziyang

---

History

- **2005:**SAGE-0.1 released February 1, 2005; SAGE=Software for Arithmetic Geometry Experimentation. Why did I name it "Sage"? claritalb.org
- **2006:** (2 Sage Days workshops); Sage is not just for number theory
- **2007:** (4 Sage Days) 100% test requirements; peer review of all new code (see trac); industry funding; NSF; Trophees du Libre
- **2008:** (7 Sage Days) Release managers besides me.

- **2009:** (8 Sage Days) Better foundations; 3d graphics; more developers (e.g., sage-combinat)
- **2010:** (13 Sage Days) More devs and users; nontrivial NSF grants
- **2011:** (12 Sage Days) Much faster http://sagenb.org with >50,000 accounts; very stable releases; undergrad curriculum development
- **2012:** (12 Sage Days)
- **2013:** Release Salvus...

See this article for more details about the (pre-)history of Sage.

Random Question Break

????

Some random examples involving Sage:

```
time n = factorial(10^7)
```

```
Time: CPU 11.42 s, Wall: 11.42 s
```

```
%cython
import numpy as np
cimport numpy as np

def mandelbrot_cython(float x0,float  x1,float  y0,float  y1,
                      int N=200, int L=50, float R=3):
    '''returns an array NxN to be plotted with matrix_plot
```

```
    '''
    cdef double complex c, z, I
    cdef float deltax, deltay, R2 = R*R
    cdef int h, j, k
    cdef np.ndarray[np.uint16_t, ndim=2] m
    m = np.zeros((N,N), dtype=np.uint16)
    I = complex(0,1)
    deltax = (x1-x0)/N
    deltay = (y1-y0)/N
    for j in range(N):
        for k in range(N):
            c = (x0+j*deltax)+ I*(y0+k*deltay)
            z=0
            h=0
            while (h<L and
                   z.real**2 + z.imag**2 < R2):
                z=z*z+c
                h+=1
            m[j,k]=h
    return m
```

__mnt_sage...1_code_sage12_spyx.c __mnt_sage...ode_sage12_spy:

```
import pylab
x0_default = -2
y0_default = -1.5
side_default = 3.0
side = side_default
x0 = x0_default
y0 = y0_default
options = ['Reset','Upper Left', 'Upper Right', 'Stay', 'Lower
Left', 'Lower Right']

@interact
def show_mandelbrot(option = selector(options, nrows = 2,
width=8),
                    N = slider(100, 1000,100, 300),
                    L = slider(20, 300, 20, 60),
                    plot_size = slider(2,10,1,6),
                    auto_update = False):
    global x0, y0, side
    if option == 'Lower Right':
        x0 += side/2
        y0 += side/2
    elif option == 'Upper Right':
        y0 += side/2
    elif option == 'Lower Left':
        x0 += side/2
    if option=='Reset':
```

```
        side = side_default
        x0 = x0_default
        y0 = y0_default
    elif option != 'Stay':
        side = side/2

    time m=mandelbrot_cython(x0 ,x0 + side ,y0 ,y0 + side , N, L
)
#    p = (matrix_plot(m) +
#             line2d([(N/2,0),(N/2,N)], color='red', zorder=2) +
#             line2d([(0,N/2),(N,N/2)], color='red', zorder=2))
#    time show(p, figsize = (plot_size, plot_size))
    pylab.clf()
    pylab.imshow(m, cmap = pylab.cm.gray)
    time pylab.savefig('mandelbrot.png')
```

```
random_matrix?
```

```
matrix?
```

```
a = matrix(ZZ, 10, 10, range(100), sparse=True); a
```

```
[ 0  1  2  3  4  5  6  7  8  9]
[10 11 12 13 14 15 16 17 18 19]
[20 21 22 23 24 25 26 27 28 29]
[30 31 32 33 34 35 36 37 38 39]
[40 41 42 43 44 45 46 47 48 49]
[50 51 52 53 54 55 56 57 58 59]
[60 61 62 63 64 65 66 67 68 69]
[70 71 72 73 74 75 76 77 78 79]
[80 81 82 83 84 85 86 87 88 89]
[90 91 92 93 94 95 96 97 98 99]
```

```
dir(a)
```

```
['C', 'H', 'I', 'LU', 'N', 'QR', 'T', '__abs__', '__add__',
'__array__', '__call__', '__class__', '__cmp__', '__copy__',
'__delattr__', '__delitem__', '__dict__', '__dir__', '__div__',
'__doc__', '__eq__', '__format__', '__ge__', '__getattribute__'
'__getitem__', '__getstate__', '__gt__', '__hash__', '__iadd__'
'__idiv__', '__imul__', '__init__', '__invert__', '__isub__',
'__iter__', '__le__', '__lt__', '__mod__', '__module__', '__mul_
'__ne__', '__neg__', '__new__', '__nonzero__', '__pos__', '__pow
'__pyx_vtable__', '__radd__', '__rdiv__', '__reduce__',
'__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__rpow__'
'__rsub__', '__rtruediv__', '__rxor__', '__setattr__',
'__setitem__', '__setstate__', '__sizeof__', '__str__', '__sub_
```

```
'__subclasshook__', '__truediv__', '__weakref__', '__xor__',
'_act_on_', '_acted_upon_', '_add_', '_add_parent', '_adjoint',
'_axiom_', '_axiom_init_', '_backslash_', '_base_ring', '_cache
'_charpoly_df', '_charpoly_hessenberg',
'_charpoly_over_number_field', '_check_symmetrizability',
'_cholesky_decomposition_', '_clear_cache', '_cmp_', '_coeff_re
'_column_ambient_module', '_decomposition_spin_generic',
'_decomposition_using_kernels', '_derivative', '_dict', '_div_'
'_dummy_attribute', '_echelon_classical', '_echelon_form_PID',
'_echelon_in_place_classical', '_echelon_strassen',
'_echelonize_ring', '_eigenspace_format', '_elementwise_product
'_fricas_', '_fricas_init_', '_gap_', '_gap_init_', '_get_cache
'_giac_', '_giac_init_', '_gp_', '_gp_init_',
'_gram_schmidt_noscale', '_iadd_', '_idiv_', '_ilmul_', '_im_ge
'_imul_', '_indefinite_factorization', '_interface_',
'_interface_init_', '_interface_is_cached_', '_is_atomic', '_is
'_kash_', '_kash_init_', '_latex_', '_latex_coeff_repr',
'_linbox_sparse', '_list', '_lmul_', '_macaulay2_',
'_macaulay2_init_', '_magma_init_', '_make_new_with_parent_c',
'_maple_', '_maple_init_', '_mathematica_', '_mathematica_init_
'_matrix_', '_maxima_', '_maxima_init_', '_maxima_lib_',
'_maxima_lib_init_', '_mod_int', '_mpmath_', '_mul_', '_mul_par
'_multiply_classical', '_multiply_classical_with_cache',
'_multiply_strassen', '_neg_', '_nonzero_positions_by_column',
'_nonzero_positions_by_row', '_numerical_approx', '_octave_',
'_octave_init_', '_pari_', '_pari_init_', '_pickle', '_pow_',
'_pow_naive', '_r_init_', '_reduction', '_repr_', '_richcmp_',
'_right_kernel_matrix', '_right_kernel_matrix_over_domain',
'_right_kernel_matrix_over_field',
'_right_kernel_matrix_over_number_field', '_rmul_',
'_row_ambient_module', '_sage_', '_sage_input_', '_sage_src_lin
'_scilab_', '_scilab_init_', '_set_parent',
'_set_row_to_negative_of_row_of_A_using_subset_of_columns',
'_singular_', '_singular_init_', '_solve_right_general',
'_solve_right_nonsingular_square', '_sub_', '_subdivide_on_augm
'_subdivide_on_stack', '_subdivisions', '_test_category',
'_test_change_ring', '_test_eq', '_test_not_implemented_methods
'_test_pickling', '_test_reduce', '_tester', '_travel_column',
'_unpickle_generic', '_zigzag_form', 'abs', 'act_on_polynomial'
'add_multiple_of_column', 'add_multiple_of_row', 'additive_orde
'adjoint', 'antitranspose', 'apply_map', 'apply_morphism',
'as_sum_of_permutations', 'augment', 'base_extend', 'base_ring'
'block_sum', 'cartesian_product', 'category', 'change_ring',
'characteristic_polynomial', 'charpoly', 'cholesky',
'cholesky_decomposition', 'column', 'column_module', 'column_sp
'columns', 'commutator', 'conjugate', 'conjugate_transpose', 'c
'db', 'decomposition', 'decomposition_of_subspace',
'delete_columns', 'delete_rows', 'denominator', 'dense_columns'
'dense_matrix', 'dense_rows', 'density', 'derivative', 'det',
'determinant', 'diagonal', 'dict', 'dimensions', 'dump', 'dumps
'echelon_form', 'echelonize', 'eigenmatrix_left',
```

```
'eigenmatrix_right', 'eigenspaces_left', 'eigenspaces_right',
'eigenvalues', 'eigenvectors_left', 'eigenvectors_right',
'elementary_divisors', 'elementwise_product', 'exp',
'extended_echelon_form', 'fcp', 'find', 'get_subdivisions',
'gram_schmidt', 'hadamard_bound', 'hermite_form', 'hessenberg_fo
'hessenbergize', 'image', 'indefinite_factorization',
'integer_kernel', 'inverse', 'is_bistochastic', 'is_dense',
'is_diagonalizable', 'is_hermitian', 'is_idempotent',
'is_immutable', 'is_invertible', 'is_mutable', 'is_nilpotent',
'is_normal', 'is_one', 'is_positive_definite', 'is_scalar',
'is_similar', 'is_singular', 'is_skew_symmetric',
'is_skew_symmetrizable', 'is_sparse', 'is_square', 'is_symmetric
'is_symmetrizable', 'is_unit', 'is_unitary', 'is_zero', 'iterate
'jordan_form', 'kernel', 'kernel_on', 'left_eigenmatrix',
'left_eigenspaces', 'left_eigenvectors', 'left_kernel',
'left_nullity', 'lift', 'linear_combination_of_columns',
'linear_combination_of_rows', 'list', 'matrix_from_columns',
'matrix_from_rows', 'matrix_from_rows_and_columns',
'matrix_over_field', 'matrix_space', 'matrix_window', 'maxspin',
'minimal_polynomial', 'minors', 'minpoly', 'mod',
'multiplicative_order', 'mutate', 'n', 'ncols', 'new_matrix',
'nonpivots', 'nonzero_positions', 'nonzero_positions_in_column',
'nonzero_positions_in_row', 'norm', 'nrows', 'nullity',
'numerical_approx', 'numpy', 'order', 'parent', 'permanent',
'permanental_minor', 'pivot_rows', 'pivots', 'plot',
'prod_of_row_sums', 'randomize', 'rank', 'rational_form',
'rational_reconstruction', 'rename', 'rescale_col', 'rescale_row
'reset_name', 'restrict', 'restrict_codomain', 'restrict_domain
'right_eigenmatrix', 'right_eigenspaces', 'right_eigenvectors',
'right_kernel', 'right_kernel_matrix', 'right_nullity',
'rook_vector', 'row', 'row_module', 'row_space', 'rows', 'rref',
'save', 'set_block', 'set_col_to_multiple_of_col', 'set_column',
'set_immutable', 'set_row', 'set_row_to_multiple_of_row',
'smith_form', 'solve_left', 'solve_right', 'sparse_columns',
'sparse_matrix', 'sparse_rows', 'stack', 'str', 'subdivide',
'subdivision', 'subdivision_entry', 'subdivisions', 'submatrix',
'subs', 'substitute', 'swap_columns', 'swap_rows',
'symplectic_form', 'tensor_product', 'trace', 'trace_of_product
'transpose', 'version', 'visualize_structure', 'weak_popov_form
'wiedemann', 'with_added_multiple_of_column',
'with_added_multiple_of_row', 'with_col_set_to_multiple_of_col',
'with_rescaled_col', 'with_rescaled_row',
'with_row_set_to_multiple_of_row', 'with_swapped_columns',
'with_swapped_rows', 'zigzag_form']
```

```
a.rank()
```

2

```
a.rank?
```

File: /home/salvus/sage-5.3/devel/sage/sage/matrix/matrix0.pyx

Type: <type 'builtin_function_or_method'>

Definition: a.rank()

Docstring:

TESTS:

We should be able to compute the rank of a matrix whose entries are polynomials over a finite field (trac #5014):

```
sage: P.<x> = PolynomialRing(GF(17))
sage: m = matrix(P, [ [ 6*x^2 + 8*x + 12, 10*x^2 + 4*x + 11],
...                  [8*x^2 + 12*x + 15,  8*x^2 + 9*x + 16] ])
sage: m.rank()
2
```

```
a.rank(
```

```
matrix(
```

```
A = random_matrix(QQ, 4); b = A^(-1); b
```

```
[   -1  1/4  1/4 -3/4]
[-1/2  1/8  1/8  1/8]
[   -1  3/4 -1/4 -1/4]
[-1/2 -1/8 -1/8 -1/8]
```

```
show(b)
```

$$\left(\begin{array}{rrrr}
1 & 0 & 0 & 0 \\
-\frac{27}{34} & \frac{5}{17} & -\frac{8}{17} & \frac{6}{17} \\
\frac{8}{17} & -\frac{8}{17} & \frac{6}{17} & \frac{4}{17} \\
\frac{11}{34} & \frac{3}{17} & \frac{2}{17} & \frac{7}{17}
\end{array}\right)$$

```
latex(b)
```

```
\left(\begin{array}{rrrr}
1 & 0 & 0 & 0 \\
-\frac{27}{34} & \frac{5}{17} & -\frac{8}{17} &
\frac{6}{17} \\
\frac{8}{17} & -\frac{8}{17} & \frac{6}{17} &
\frac{4}{17} \\
\frac{11}{34} & \frac{3}{17} & \frac{2}{17} &
\frac{7}{17}
```

```
\end{array}\right)
```

```
show(integrate(sin(x)*cos(x), x))
```

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

```
f(x) = sin(x^2) * cos(sin(x)) + tan(x) - log(x); f
```

```
x |--> sin(x^2)*cos(sin(x)) - log(x) + tan(x)
```

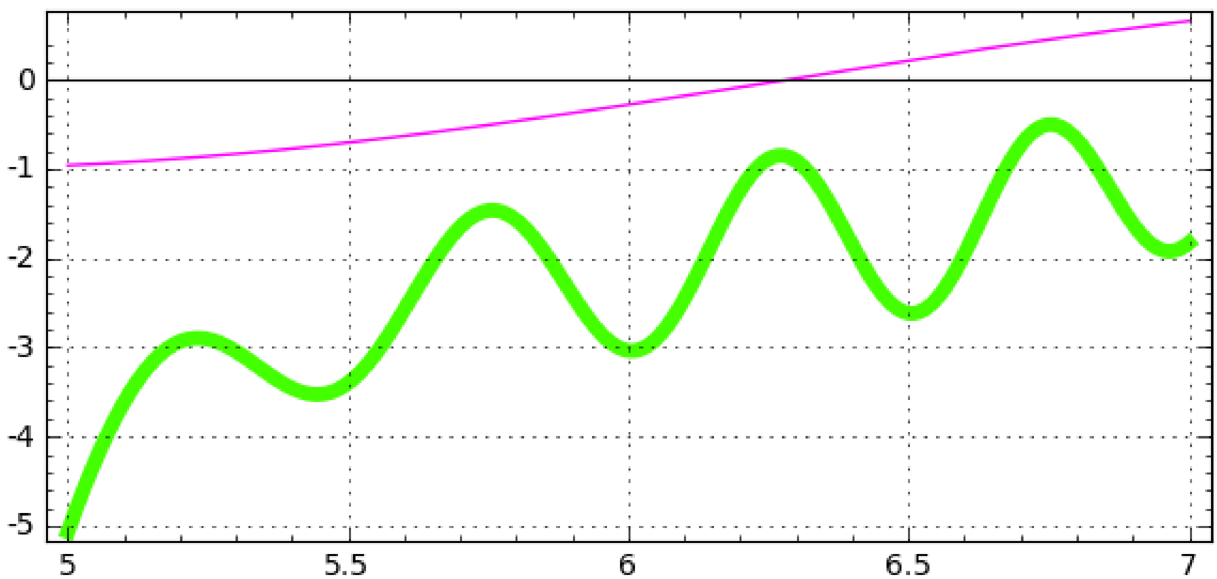
```
show(f)
```

$$x \mapsto \sin\left(x^{2}\right)\cos\left(\sin\left(x\right)\right) - \log\left(x\right) + \tan\left(x\right)$$

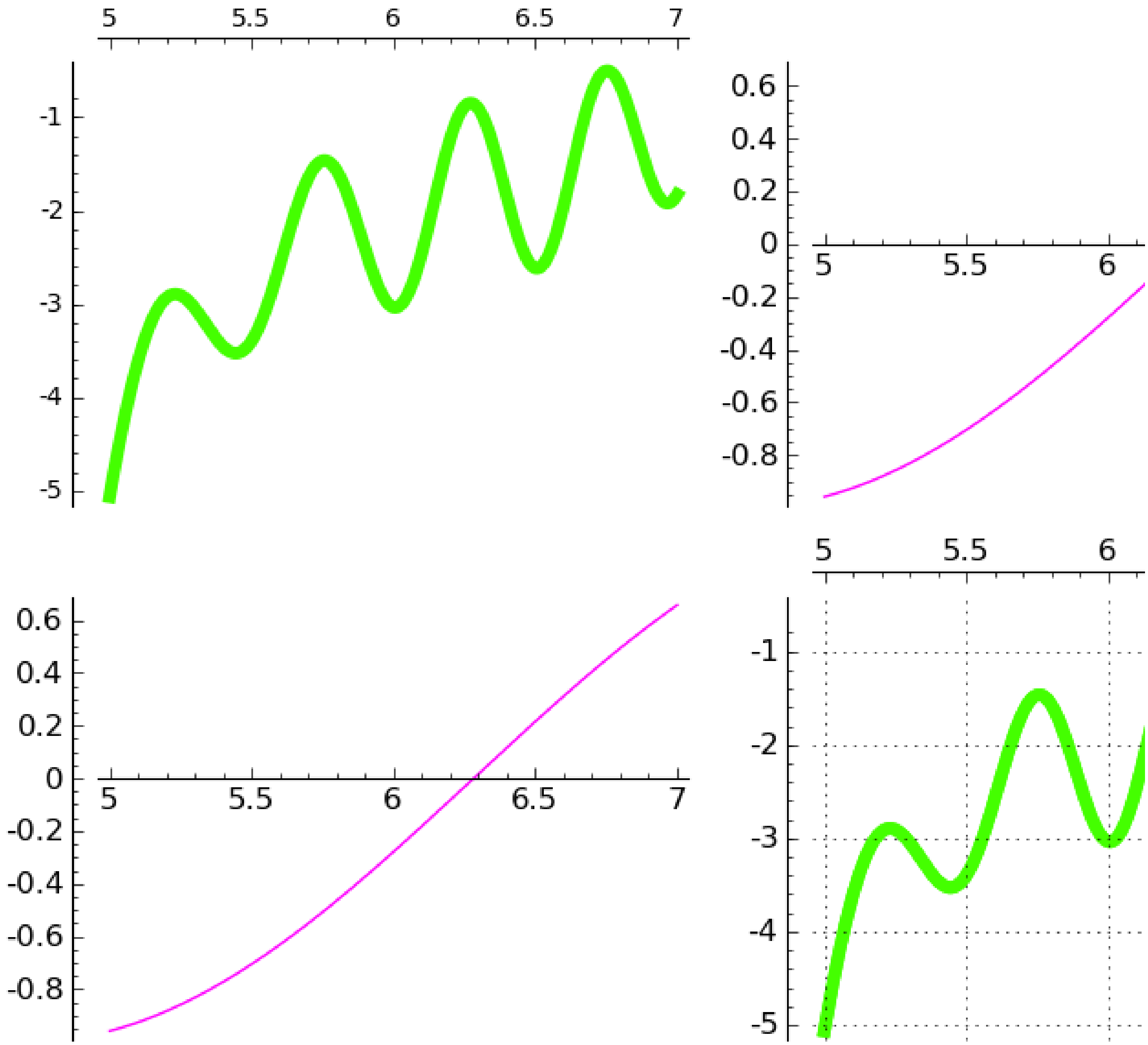
```
latex(f)
```

```
x \ {\mapsto}\ \sin\left(x^{2}\right)
\cos\left(\sin\left(x\right)\right) - \log\left(x\right) +
\tan\left(x\right)
```

```
G = plot(f, (5, 7), color='#44ff00', thickness=5,
    figsize=[6,3], gridlines=True, frame=True)
H = plot(sin(x), 5,7, color='magenta')
G + H
```




```
graphics_array([[G, H], [H, G]])
```

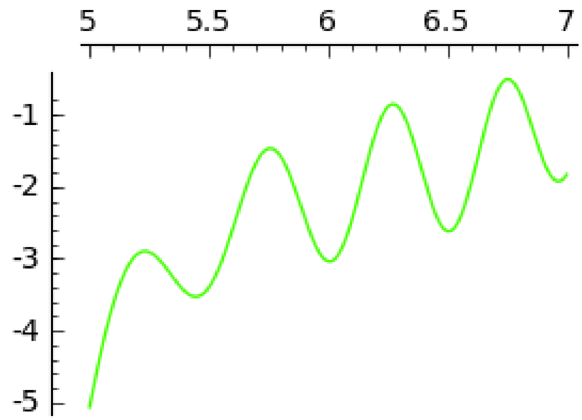


```
help(interact)
```

[Click to open help window]

```
@interact
def g(xmax=(6, 20), color=Color('red')):
    G = plot(f, (5, xmax), figsize=[6,3], color=color)
    G.show(ymin=-10, ymax=10)
```

```
g(7)
```




```
G = plot(sin, 0, 10)
G.save('a.pdf')
```

a.pdf

```
sage.interfaces.
```

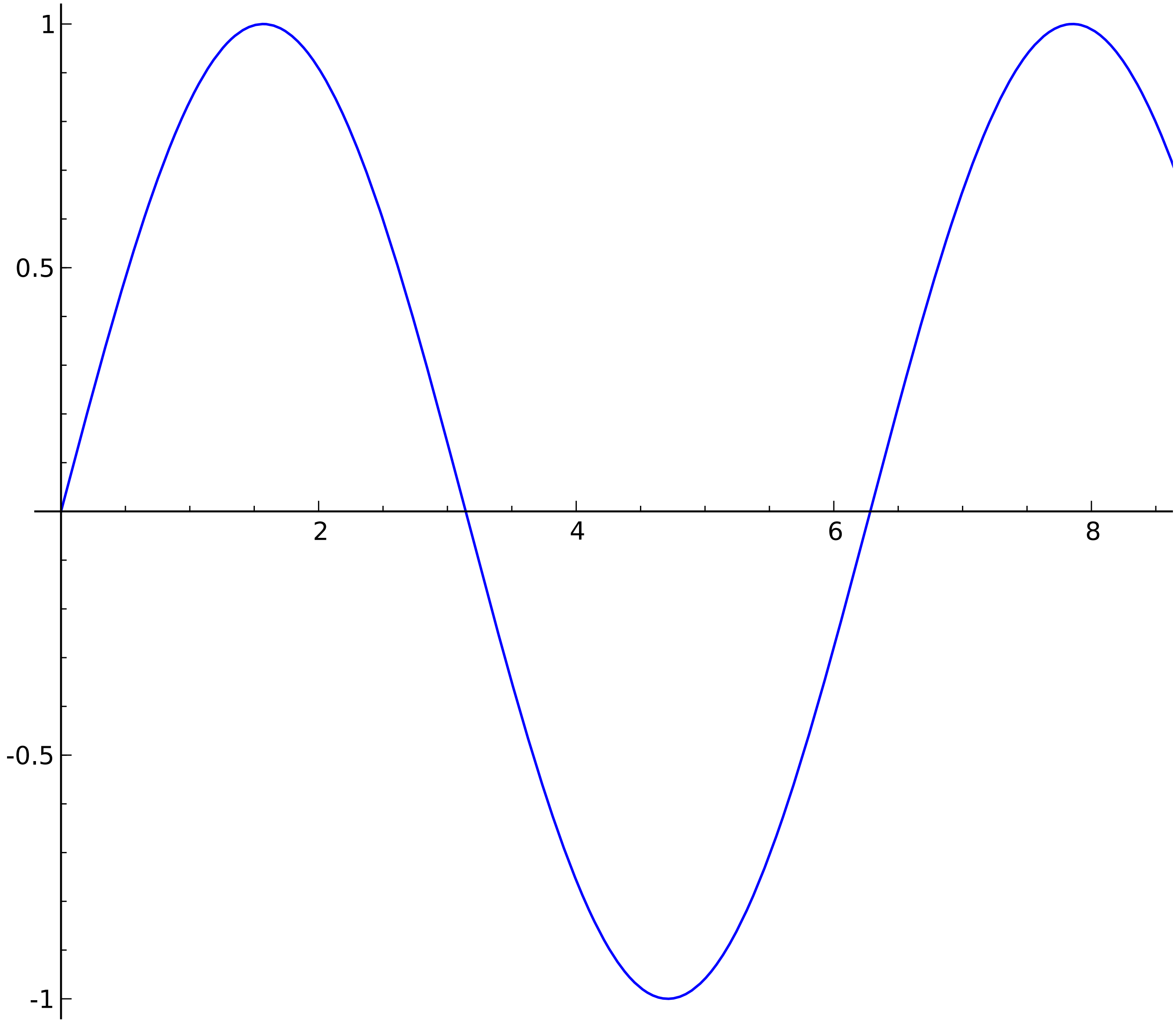
```
G.save('a.ps')
```

```
/home/salvus/sage-5.3/local/lib/python2.7/site-packages/matplot:
ight_bbox.py:55: UserWarning: bbox_inches option for ps backend
not implemented yet.
  warnings.warn("bbox_inches option for %s backend is not
implemented yet." % (format))
```

a.ps

```
G.save(
```

```
G.save('a.svg')
```




```
a = octave('rand(5)'); a
```

```
0.549036 0.512252 0.604208 0.0767369 0.747327
0.266268 0.756133 0.682624 0.645004 0.31979
0.914937 0.88774 0.630625 0.973085 0.791813
0.9205 0.500435 0.365157 0.392919 0.994673
0.651773 0.531602 0.127743 0.911378 0.685484
```

```
a.eig()
```

```
(3.02114,0)
(-0.192864,0.125433)
(-0.192864,-0.125433)
(0.377322,0)
(0.00146815,0)
```

```
%octave

x = rand(5)
eig(x)
```

```
x =
```

```
 0.343843 0.589597 0.408991 0.989844 0.0934817
 0.626751 0.00267151 0.797916 0.978089 0.465727
 0.20643 0.0241775 0.966753 0.660207 0.475048
 0.0305844 0.97238 0.381771 0.431595 0.556873
 0.758349 0.885778 0.676631 0.431916 0.627955


ans =

 (2.64135,0)
 (-0.644357,0)
 (-0.0211313,0.46375)
 (-0.0211313,-0.46375)
 (0.418085,0)
```

How to Use Sage

- Type Python code into the boxes and press shift-enter or click "evaluate". More about the notebook... (read the bottom of the page that appears)
- There is extensive documentation, including a massive reference manual.
- Get help at: live chat, ask.sagemath.org, and the sage-support google group.

Interactive Image Compression

(using numpy)

```
import pylab, numpy

X = pylab.imread(DATA + 'sacnas.png')
A_image = numpy.mean(X, 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=8)
    html("%sx%s image compressed to %.1f%% of size using %s
eigenvalues."%(
            A_image.shape[0], A_image.shape[1],100*
(2.0*i*n+i)/(n*n), i))
```

Number Theory

```
factor(2009201020112012)      # makes use of PARI
```

```
2^2 * 43 * 2269 * 5148259709
```

```
# Jon Bober - Rademacher's formula
time number_of_partitions(10^6)
```

```
1471684986358223398631004760609895943484030484439142125334612747
661174189186182763301488739835975558420153741306002880959293873
232270327849578001932784396072064228659048713020170971840761025
986084690814282935670692978599129051989944549067221999782345287
740222882298501367675662947818874946878790038246999881977292006
668735996662273816798266213482417208446631027428001918132198177
651123454259502672842445259229678119344813999466473010574256435
949891814852853513705513994767199816914590220155991019596014174
715430750022184895815209339012481734469448319323280150665384042
417958775176129491624814247999880293650719525707448504757166277
033914424951138232981952630083364898260458377122024553049963821
028531832004519046591968302787537418118486000612016852593542741
504626724547323732184583342751252422746539913017407694128084740
422179992860711083363033162982891024446496968053954167918754800
636774022023128467646919775022348562520747741843343657801534130
197553037516970799928704028567784161934747236817177215404666430
15630003467104673818
Time: CPU 0.03 s, Wall: 0.03 s
```

```
@interact
def _(n=(25..10000)):
    plot(prime_pi, 0, n, gridlines='minor').show(figsize=[8,3])
```

Graph Theory

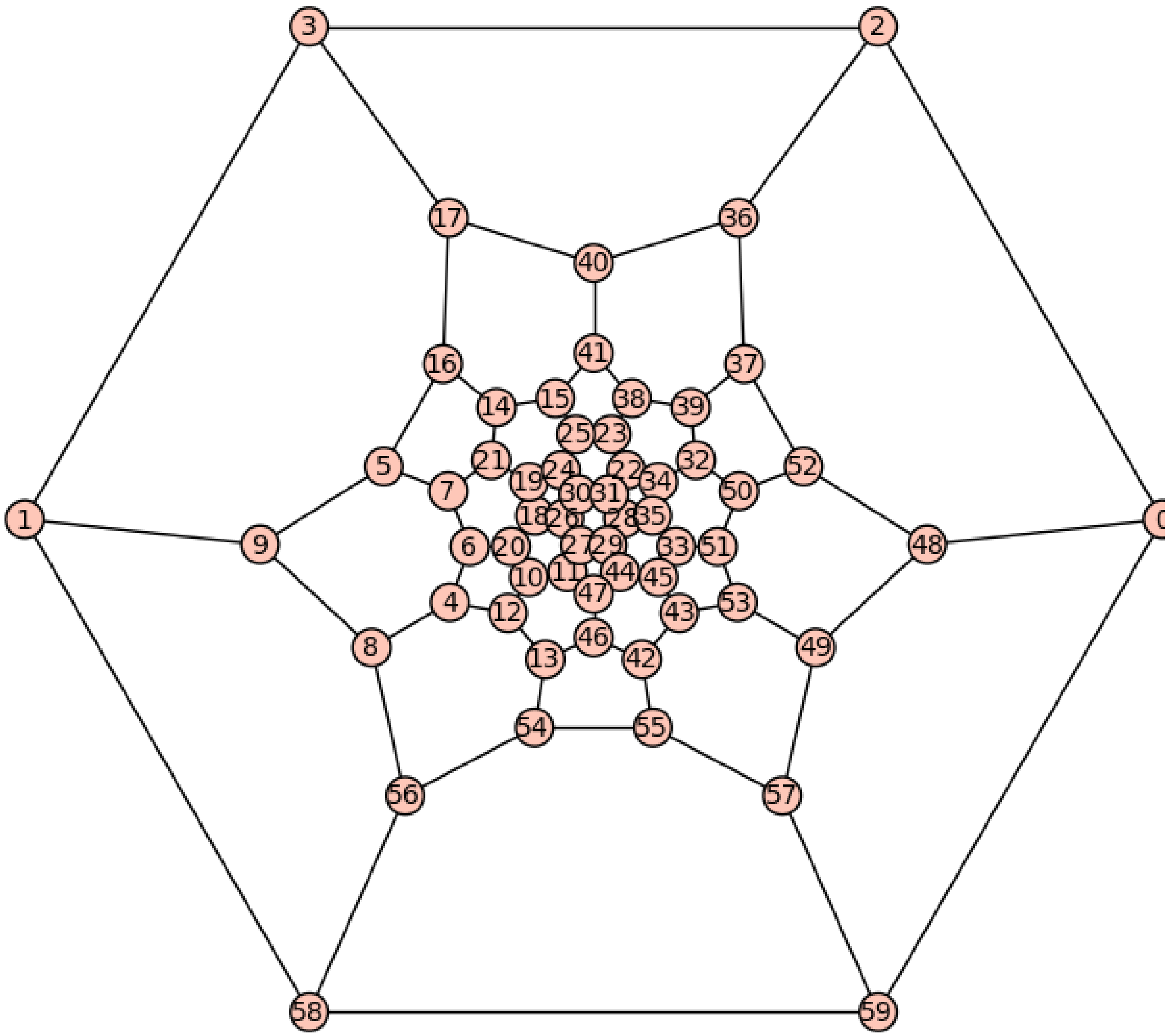
"Sage's graph theory crushes anything I have met from the point of view of methods implemented. I would say: 'if you found a proprietary graph library and you are convinced that it is better for your needs than Sage, your license is on me.' But those licenses are really expensive :-D"

```
graphs.
```

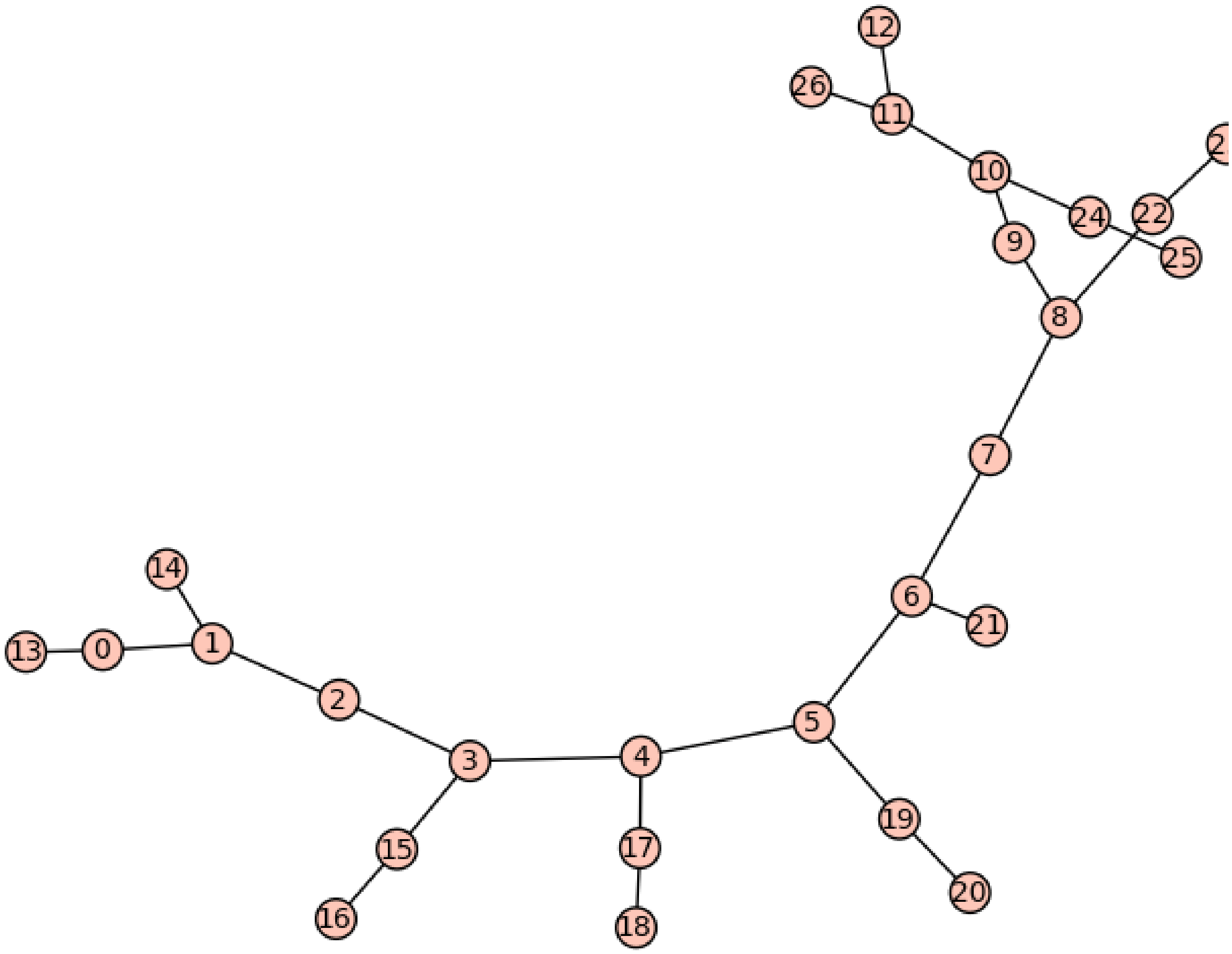
```
Traceback (click to the left of this block for traceback)
...
```

SyntaxError: invalid syntax

```
G = graphs.BuckyBall()
G.plot().show(figsize=8)
```



```
set_random_seed(1)
G = graphs.RandomLobster(8, .6, .3)
show(G, figsize=7)
```




```
G.automorphism_group()
```

    Permutation Group with generators [(12,26)]

```
G.chromatic_number()
```

    2

```
G.shortest_path(13,20)
```

    [13, 0, 1, 2, 3, 4, 5, 19, 20]

Cython

Cython C-Extensions for Python

- Smooth transition between Python and compiled C code.
- Make code that involves lots of manipulation of C-level data structures optimally fast
- Heavily used in scientific computing using Python.

```
def python_sum(n):
    s = int(0)
    for i in xrange(1, n+1):
        s += i*i
    return s
```

```
python_sum(3)
```

```
14
```

```
time python_sum(2*10^6)
```

```
2666668666667000000
Time: CPU 0.18 s, Wall: 0.18 s
```

```
timeit('python_sum(2*10^6)')
```

```
5 loops, best of 3: 173 ms per loop
```

```
def python_sum2(n):
    return sum(i*i for i in xrange(1,n+1))
```

```
time python_sum2(2*10^6)
```

```
2666668666667000000
Time: CPU 0.21 s, Wall: 0.21 s
```

```
%cython
def cython_sum(long n):
    cdef long long i, s = 0
    for i in range(1, n+1):
        s += i*i
    return s
```

__mnt_sage...1_code_sage91_spyx.c __mnt_sage...ode_sage91_spy

```
cython_sum(3)
```

```
14L
```

```
time cython_sum(2*10^6)
```

```
2666668666667000000L
Time: CPU 0.00 s, Wall: 0.00 s
```

```
timeit('cython_sum(2*10^6)')
```

```
125 loops, best of 3: 2.79 ms per loop
```

```
165/.663
```

```
248.868778280543
```

```
%cython
def cython_sum2(long n):
    cdef long long i
    return sum(i*i for i in range(1,n+1))
```

__mnt_sage...1_code_sage97_spyx.c __mnt_sage...ode_sage97_spy:

```
time cython_sum2(2*10^6)
```

```
2666668666667000000
Time: CPU 0.22 s, Wall: 0.22 s
```

Of course, it is better to choose a different algorithm:

```
var('k, n')
factor(sum(k^2, k, 1, n))
```

```
1/6*(n + 1)*(2*n + 1)*n
```

```
def sum2(n):
    return n*(2*n+1)*(n+1)/6
```

```
sum2(2*10^6)
```

```
2666668666667000000
```

Even then, Cython provides a speedup:

```
%cython
def c_sum2(long long n):
    return n*(2*n+1)*(n+1)/6
```

__mnt_sage..._code_sage104_spyx.c __mnt_sage...de_sage104_spy:

```
c_sum2(3)
```

```
14L
```

```
c_sum2(2*10^6)
```

```
-4077886789512586603L
```

```
n = 2*10^6
timeit('sum2(n)')
```

```
625 loops, best of 3: 2.29 µs per loop
```

```
timeit('c_sum2(n)')
```

    625 loops, best of 3: 145 ns per loop

```
2.01/.218
```

    9.22018348623853

But at a cost!

```
c_sum2(2*10^6)          # WARNING: overflow -- it's just like C...
```

    -4077886789512586O3L

```
n=2*10^6; n*(2*n+1)*(n+1) > 2^63
```

    True

Solving Equations

Solve a ***cubic equation***:

```
x = var('x'); show(solve(x^3 + x - 1==0, x)[0])
```

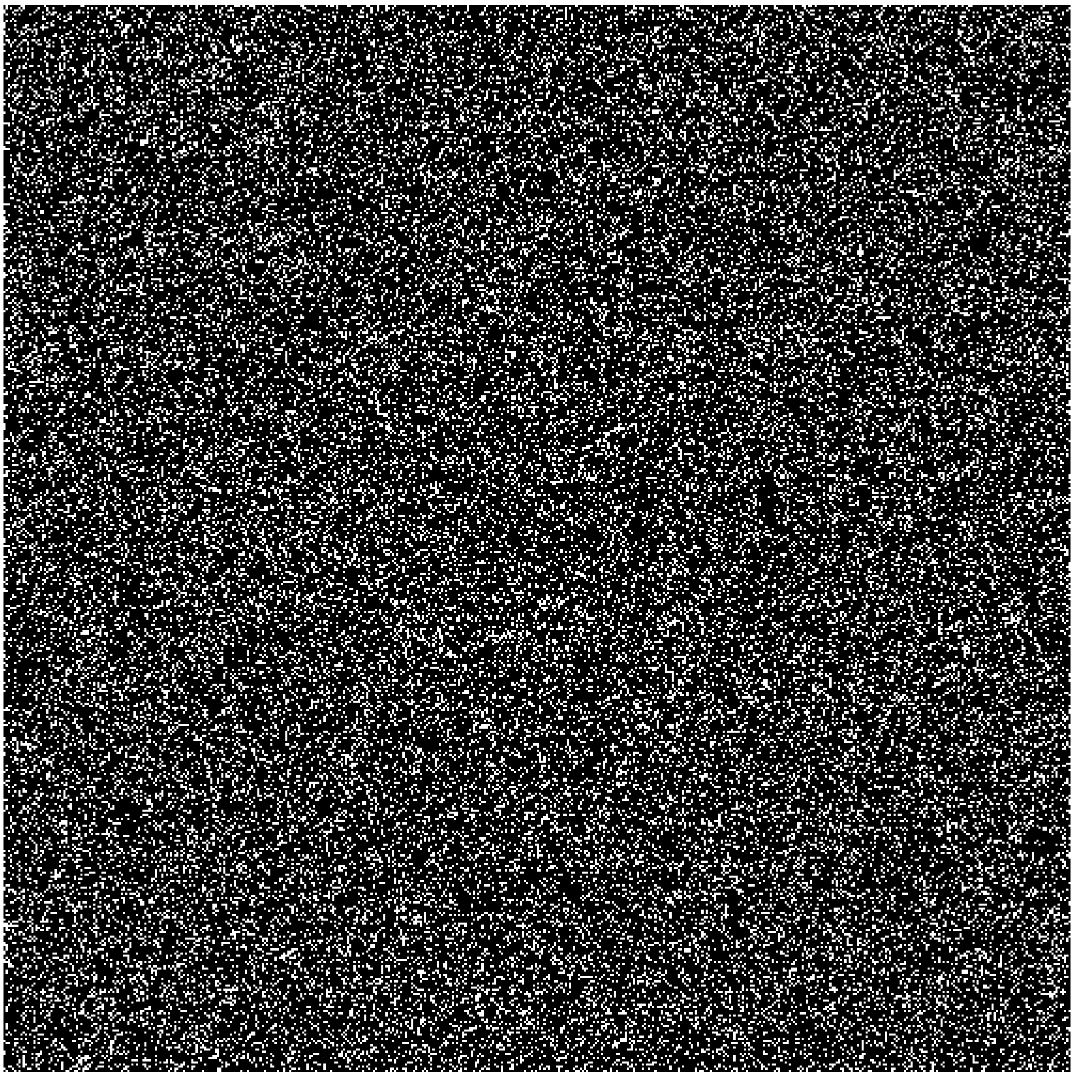
$$x = -\frac{1}{2}\left(i\sqrt{3}+1\right)\left(\frac{1}{18}\sqrt{3}\sqrt{31}+\frac{1}{2}\right)^{\left(\frac{1}{3}\right)} + \frac{-i\sqrt{3}+1}{6\left(\frac{1}{18}\sqrt{3}\sqrt{31}+\frac{1}{2}\right)^{\left(\frac{1}{3}\right)}}$$

Solve a system of ***two linear equations*** with one unknown coefficient α:

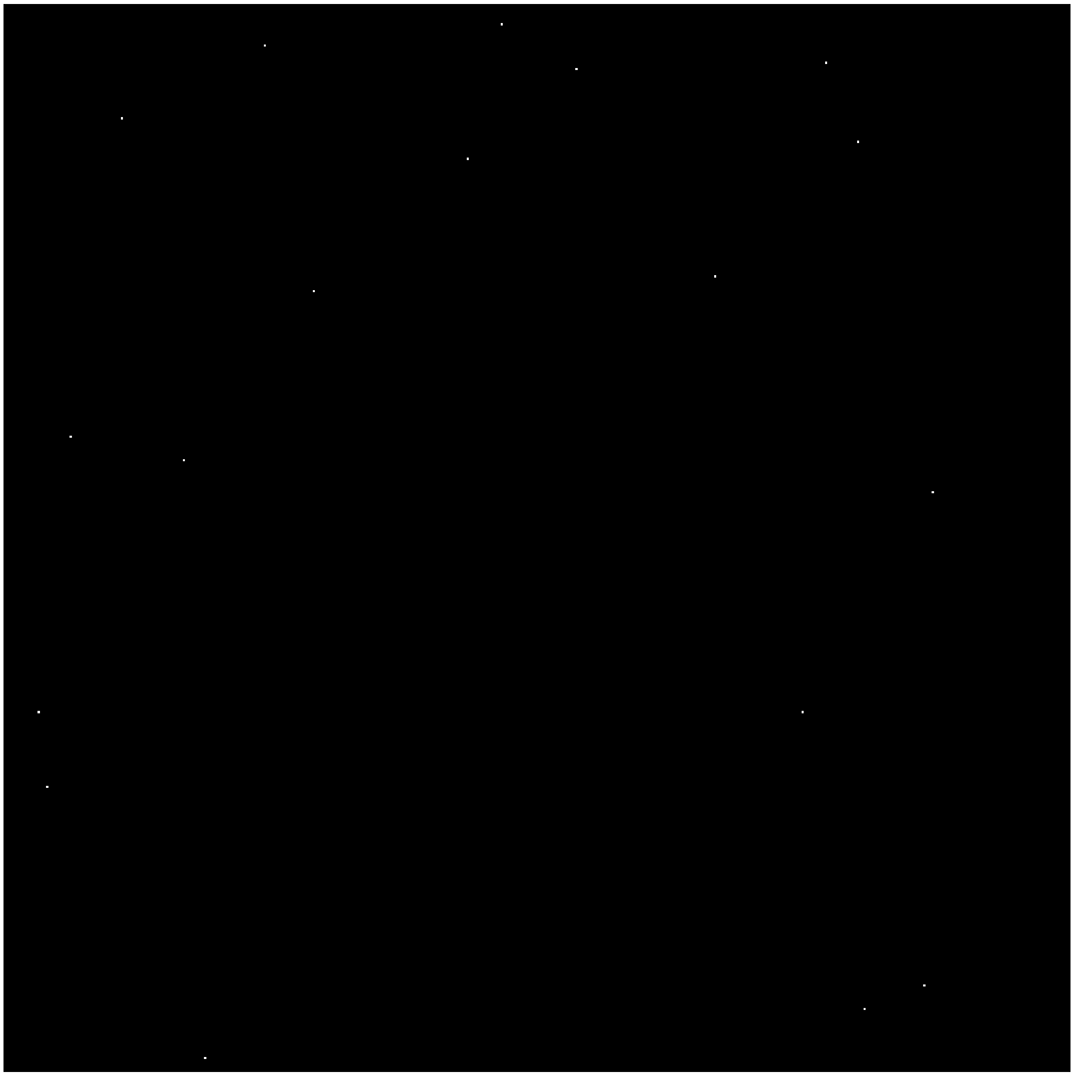
```
var('alpha, y')
show(solve([3*x + 7*y == 2, alpha*x + 3*y == 8], x,y)[0])
```

Solve a ***system of 500 linear equations*** exactly over the rational numbers:

```
n = 500
A = random_matrix(ZZ, n)
A.visualize_structure()
```



```
(A*A).visualize_structure()
```



```
import scipy.optimize
```

```
scipy.optimize.bisect(
```

```
# IML is used -- http://www.cs.uwaterloo.ca/~astorjoh/iml.html
time w = A \ v
```

```
Time: CPU 1.84 s, Wall: 1.85 s
```

```
print w.str()
```

```
WARNING: Output truncated!
```

full_output.txt

```
[
```

-351702876068802906437552420686055850326076457904676962275100083
401420955261319182505472655403446871133375351213855350226815464
541678542779937822420479756487655377896847799471633977576723782
752013079903081977572402942342855631847634917334375567725200949
981494332871024651254158873412391649819482714094296169547438994
166187839273830840430242204133808650932888101185375789831810538
749302374084369069440792237427868339146115514095572829020266899
134355557855057198503380254232298641549695798842795869295699735
920100244120646553550505898020495083583179450653843216750029252
795806490140294417796752695156539524124805975616026523319266984
226942792117185857783063808851623050866691080434331157140193997
656405532372637424988128302930037221573841204291852109644817711
003582668463237232745438803646481729385618653719290569550431763
479911712917380077097901887716153891517164989985540840321675523
530360179417770975752951610003948983037774750319911117156732832
778539539244149073095868925297296767522098873014512743624494378
988592956277583816139788377093357552496933776948081224203156447
737484105107310982124866738518101800129697107516438736576977941
437927029350938265391137496664227427915828658742276488248475686
797308951586142050635510616384914735047939100560830346367036281
909909358743617446014742435439813694562291652176283751189754664
361180619892765924138532602101463250422678650464862058688845651
378935711191095339677561251599756523993225002326911920613591375
810713117303327205420957785613280646524680178422545699670542002
185206878446429324207574638947936431387935201327903186688186655
889749881958253860610185138652298499093125007922510871073642155
160192682022566179204513802804002649348052390662029135279718239
571118081333104312614968911252810197919260114909352478204286878
399735896224340600057359854531576146926251200089897481071278163
047295612770389853217738467738241538454558869964299842463856929
637878107844843582575044052722770755619169229082572645352435097
865291508953639079652724169694694026175524388621142937766905589
623925909303675286193416452595298612780914411142216266018732516
619208172280955826296174872463796580072215353832080758361490836
805804530940354095353962448272508728028384915074672326844898676
080066542916543081427459267666550278740733523244436200829487939
3811194688142700374028190883369685908216732077/40677828754842123
331161732867083284451766898265331336067813926311992369988856628
673485573566711552909564301747518750953602511057982067427351271
165067814139492190954219009946130255863282988325207830875101159
000462483853695254032218916775883778046276732513287003348820832
733915165992021611860243478289208286874090823481776402174489759
612915621713638267535188696384621721767869207503970464394197565
331329906325922062334126886845307227747533166221063380004741504
008722003300796042035953934256491705990739861502852318631779763
996046241653617055408115480273443479448433183797541958284471397
075956823953574961456072422821379853471265957417978467907648030
104168048866621341878285450037144900446462813715169327824861140
191054416345143797859434383214986994913744299749414119161883006
447066207671860293291735706044903379459163329722215462487004764
395022789804832840107300683573385442349998224750211775548096956

```
12088131965551967459900917790184504128348366688399796247714348З
1162765197083703777482201078935003185060800078772950446193913338
8344042981270732923845847840661087831919093000529797399599904218
9229258550852534919556375386828966901377215773658334869709938404
1443445476354882729114398329220660753134328170369143687044568081
4539273223910518127512694139126508695831281678860513712181333729
2590436549601884994184519097657397061550328468140875020630728462
3564230141908136118998207860731109237365075922240988708341841893
1093842374187901309709249334375710994985735557672788610080629408
1201517155444973479370259713482031022520308410577644943311952468
1238661819696470413701424802298593548776978832108656234910264109
2205911593318386075749335920440095933187160459299339643209960990
9544012564454057364306563040653021522038450273701898168250233560
1542331594468641179158138428277349731703737219963018417632363299
1494386783276795049901144295223363777310171682399431489512674283
8722166638871500680396177707298901665698448682925479970938945488
3015207286373345737063021706179180185939364399949813105084680637
5909826757786706357829130980704081337842853100001277249846738997
6878582385201810484324518892638117928226308648220890311998167122
9766497072354298239001901713874648443916904411649363916870517743
6627139496012306231401783236358878349778653017133170776393229190
3962899031016050228081413485309678808196586762986239037632308148
081447209515623489102340]
[
7126109666514325766865816281323217854027936420028241233111018294
6957296194203263097154402938277462517133906629031143528461511612
0441097120169029621264201559926948837404463199725065392914261993
6363726571652597606910738417927277709146849430771949030118855670
3903408161566172523537564191278182816453577246135823555431848608
6036241632315995877050056198788230497329665256989181826524968540
8071379878700091072896777058914549680319691201740287717442921639
8454332964408736836608492509718496299166559932718284968262327950
8938573116045910787352245863991905838467324633559360554430624817
9591321929375547989733588081213978213880427816067055456996112221
9081364887351926806664833447256645215110538011580734213103731303
4512364445006119180576475961863594091165497083388721381453096455
0227859038467274943342912925623941767234969615689598240188986260
0521194373576446372251042914604915703153885476062853490230778617
2731395396147714122001657791717438876674684606713884558865151857
1759929442542118633823647352595428481329724477625678899030742979
5337583158223898524823649696421317905019268225575949674635727905
7926087684404625778663873834394082223883152404741743301223911129
9762538824905584485729938365647613520107089894172673404553763545
2154423840684598978509653865006260018636577638045534236784825990
2944057837796311113705969598918970167771490316260084254566579197
7137040461551339446476404041985684490374430677637646489311742060
6829367769730234685664104514410692047813979570039090331988105104
7339572230363816338924666227886503338037720379727395038207464330
6688667113373134986705023460093715301789246640264384562907374162
1206459579307745005379571026886128547970958807051253992658517350
7049028737268865920757894719932155963464160432598594917437126150
```

```
9807074423539828526047627467728358304222164472474880598327544348
7736828906201635260905498818075844120226872577149844466426036871
5040670465420788840440266441899434832132557088303837838676997019
3475657889853649812233805955825366263349716294788744845256356435
7947849234549156401250286511306821379510598707319657116203529292
2091755395562745097297150142145693347197839779225002496499973787
7582566251807629976297590706791775078663740690207585833545702990
8049817906399126226217253274418040438810779328875787606446771723
0896802045553120040221747735815047265560792085411289595812221132
39135952750960631854645917841141921884756787/8135565750968424653
2323465734166568903533796530662672135627852623984739977713256198
6971147133423105819128603495037501907205022115964134854702542081
0135628278984381908438019892260511726565976650415661750202319904
0924967707390508064437833551767556092553465026574006697641664573
7830331984043223720486956578416573748181646963552804348979519690
5831243427276535070377392769243443535738415007940928788395130442
2659812651844124668253773690614455495066332442126760009483009733
7444006601592084071907868512983411981479723005704637263559527332
2092483307234110816230960546886958896866367595083916568942794281
1913647907149922912144845642759706942531914835956935815296060680
8336097733242683756570900074289800892925627430338655649722281087
2108832690287595718868766429973989827488599498828238323766013858
4132415343720586583471412089806758918326659444430924974009529213
0045579609665680214601367146770884699996449500423551096193912248
1762639311039349198018355803690082566967333767995924954286966633
2553039416740755496440215787000637012160015754590089238782667615
8808596254146584769169568132217566383818600105959479919980843630
5851710170506983911275077365793380275443154731666973941987680869
8689095270976545822879665844132150626865634073828737408913616258
7854644782103625502538827825301739166256335772102742436266745821
8087309920376998836903819531479412310065693628175004126145692400
2846028381627223799641572146221847473015184448197741668368378701
8768474837580261941849866875142198997147111534557722016125881629
0303431088994695874051942696406204504061682115528988662390493619
7732363939294082740284960459718709755395766421731246982052821911
1182318663677215149867184088019186637432091859867928641992199215
8802512890811472861312608130604304407690054740379633650046712133
8466318893728235831627685655469946340747443992603683526472659870
8877356655359009980228859044672755462034336479886297902534856758
4433327774300136079235541459780333139689736585095994187789097701
3041457274669147412604341235836037187872879989962621016936127514
1965351557341271565826196140816267568570620000255449969347799557
5716477040362096864903778527623585645261729644178062399633424585
3299414470859647800380342774929688783380882329872783374103548767
5427899202461246280356647271775669955730603426634155278645838015
2579806203210045616282697061935761639317352597247807526461629727
2894419031246978204680]
[
-16036847841294577406021754226656561474148033966447388200349679
7528978494307205314155482905329393092185760710201540229570946252
4285905979470817586260951115842175581719241499599904766447016618
```

0387120939161533080399107778728384741444771537907653922319759894
9892037189177735329834429998953334110710608345579591893430349013
3817881490017547983572191637779556906242430907809400383943787442
3397662074039674778320791060793269012306741816670290835859282700
4266385062524514678446957588738048992210145313498295774207549900
7891957507693480732420337737984779499236439410063957963614860720
3870383782866702158458546676461882498748278944745533497547122031
0031976978325039051384181023781937035902246443156099394592116569
8511100311810852778007247680814893993810010965794094758617218791
2363344711398871523122632382536199508988784889511121664982989461
4827549378250570551643712455667323633012464484344071735947589590
9051621508776806611325884897394319344915096495144216940643245957
8075639597977267911753289413618179937603437414778980866101466420
0225264630513299469577205354435088030146797253519034255226454862
5965386404566396981251680448300770072189210351796914451248227508
3590763916770854032550397882750430506899363811690620437674458177
1859476096404547245327726043250105947135800027722412874062815554
8712675225726270837164249351489695306753002874562631637613828332
7051341817671547784446172478777405397645740727692316304905027489
0332413632773410578006364443861219434579735664425998914700296503
5160784206162019197173066715838886238823013556879551094407917268
9176839207112932036985722404197924010064653628113454857728315760
8890052356472893573477570858273664567883261067402062703657517592
1262451808934999380085089404092063011206629572302034907644803068
0623765188813668626650165413652927518384485750979325936171383262
9135785934393479378578374769912745818084799489268728361059482323
9958711362241007207649342787741559927764673118460376336304538824
3407311085883629008582793117780678702334008758518549450449090031
5025393705225180327886470362580983643222600549149947874336491553
6812177142726453277561953962442172546006176198445807634156854731
9704415199206135752245256559320142639667436992446660525914733367
8297309869979834357808092708256214038078184624627189177859528290
1324471885905809914711650798190698800828874157737812120365738047
7644926508362935501173611761755873866318209749/8135565750968424
6623234657341665689035337965306626721356278526239847399777132561
3469711471334231058191286034950375019072050221159641348547025420
3301356282789843819084380198922605117265659766504156617502023199
0009249677073905080644378335517675560925534650265740066976416645
4678303319840432237204869565784165737481816469635528043489795190
2258312434272765350703773927692434435357384150079409287883951304
6626598126518441246682537736906144554950663324421267600094830097
0174440066015920840719078685129834119814797230057046372635595273
9920924833072341108162309605468869588968663675950839165689427942
1519136479071499229121448456427597069425319148359569358152960600
2083360977332426837565709000742898008929256274303386556497222810
3821088326902875957188687664299739898274885994988282383237660138
8941324153437205865834714120898067589183266594444309249740095292
7900455796096656802146013671467708846999964495004235510961939122
2417626393110393491980183558036900825669673337679959249542869660
2325530394167407554964402157870006370121600157545900892387826670
6688085962541465847691695681322175663838186001059594799199808430

```
845851710170506983911275077365793380275443154731666973941987680
288689095270976545822879665844132150626865634073828737408913616
907854644782103625502538827825301739166256335772102742436266745
518087309920376998836903819531479412310065693628175004126145692
712846028381627223799641572146221847473015184448197741668368378
218768474837580261941849866875142198997147111534557722016125881
240303431088994695874051942696406204504061682115528988662390493
247732363939294082740284960459718709755395766421731246982052821
441182318663677215149867184088019186637432091859867928641992199
908802512890811472861312608130604304407690054740379633650046712
308466318893728235831627685655469946340747443992603683526472659
298877356655359009980228859044672755462034336479886297902534856
744433277743001360792355414597803331396897365850959941877890977
603041457274669147412604341235836037187872879989962621016936127
181965351557341271565826196140816267568570620000255449969347799
375716477040362096864903778527623585645261729644178062399633424
953299414470859647800380342774929688783380882329872783374103548
325427899202461246280356647271775669955730603426634155278645838
792579806203210045616282697061935761639317352597247807526461629
162894419031246978204680]
[
670780905754609967862650398220177131177982831134934789266535389
467577330756466207386569264818123399433169738384683949717533896
983235131878998680794641577340432556024086819675201612019250553
464903570471904668198180264901434541450940837169885399568778669
758481874480415152086578115719420645796338395894045303857017872
355021202979751551218581493173606197540792304295608684220170950
456340847868979957454837223853465437437654331101665463453971428
595527017282760740988830574458702780688447115160246692170227156
795568287058609438922666958371226900785641775459048793253446630
414135658896061425931644562780728315670905927070614377744904990
902240154206236820377356130784656721949939599643812946420642048
473826463965716107582709807943519709120212335458654568865245967
466835470878718792567580333908554121695311325318702591696054956
623557772168943006954201525712848276804138615248464308668490975
073954201721355382859123192390170085659680325267657246081239298

...

928314429244369494603036889639548333673675740224437536949675160
369973375028439799429422306911544403225176325840480606862177989
810388539281240900812336423105797732478098723904495464727878588
056992091943741951079153284346249396410564382240882364637327354
973436817603837327486418371973585728398439843035817605025781622
262521626120860881538010948075926730009342426792616932637787456
325537131093989268149488798520736705294531974188597754713310718
045771808934551092406867295977259580506971351647488866655548600
847108291956066627937947317019198837557819540265206082914549338
520868247167207437574575997992524203387225502818363930703114682
165239228163253513714124000051089993869559911410751432954080724
980755705524717129052345928835612479926684917099906598828941719
```

```
0760685549859377566761764659745566748207097534906508557984049224
0713294543551339911461206853268310557291676031095851596124064200
2565394123871523278634705194495615052923259455523257888380624939
936]
[
9452604384841635630448165532705929180013947519126017749118253654
2101869516492466616395590211147173762603314811628294546497345263
6366641558309509006703299406969133491200387089747648502631421030
9186329396271137504491925888413590269688660795169890728450400719
3991295151325555378155861981887132570589489287106795044073018125
9404146682984043518555409524908610591827037335124406995030482168
8485747942925954120831937822556335694309514975936941349683283111
2111626673937106082001550878189006390542393132340296934604234894
4414055385295376960131602467668091862789245398763041962002049700
5983279968298101016493043916461907407976562776170832260799193952
4210888444042385829536994393210450651199728670394153493355137383
9336318965774912537300964996205930449735197556394612320431431657
1522647896946199544760903752333615387015336492881167641155322540
8548901149494658657504048591644907239559563758928695118666640790
1477798082549475014650302220028237616548563954305125772477842115
4905321873804046342547689890649658781709526313599254513013706237
1054377000604587293230405831288638233961825316508199469981546355
1104079639036058919327200406009080117131938728122370062464955375
3382239233458890664397614052899902884286390706548769219295682654
2905768632303202778273065462907624324570343872094765158455209524
6532433653032537586697445046594837627017753329172739735683828562
5316575413386120551381747535283558210016046915661635736158315279
4531595903687486584099485877470410018067517554997896868834157780
2754816452390531359574565456952469199567096359644344425802709752
5526459179067017675005061431661595684972631342447599381832525653
4474154600776120953534094860058553868504396439566981898063636740
9873231971080158640165904170905174920785332077069814393335767393
5573673592792881265743628961607240729477280435568963949073093567
9208924749626242048197726284571299226330550247095469436831654022
1937142596453104473443329048101991005953290761901319438416314409
7290693425678039371853258013775637756445050835553197820844349418
6972864807839554277117559681492762524089402439947751920848126754
8125175996256052285911018920027063666174049188939807179773595877
0535732974431752613386561165658520506420040207431365529867868305
4578587320048479176856369922262578475480783237925833497937251244
8369450898197734388489696412078681753313029835991527348421227545
0898518444176051571779298422319473902516203333/406778287548421232
3116173286708328445176689826533133606781392631199236998885662809
7348557356671155290956430174751875095360251105798206742735127104
6506781413949219095421900994613025586328298832520783087510115995
0046248385369525403221891677588377804627673251328700334882083228
3391516599202161186024347828920828687409082348177640217448975984
1291562171363826753518869638462172176786920750397046439419756522
3132990632592206233412688684530722774753316622106338000474150480
0872200330079604203595393425649170599073986150285231863177976360
9604624165361705540811548027344347944843318379754195828447139714
```

```
7595682395357496145607242282137985347126595741797846790764803O3
04168048866621341878285450037144900446462813715169327824861140 5
910544163451437978594343832149869949137442997494141191618830069
470662076718602932917357060449033794591633297222154624870047646
950227898048328401073006835733854423499982247502117755480969561
208813196555196745990091779018450412834836668839979624771434833
162765197083703777482201078935003185060800078772950446193913338
344042981270732923845847840661087831919093000529797399599904218
229258550852534919556375386828966901377215773658334869709938404
443445476354882729114398329220660753134328170369143687044568081
539273223910518127512694139126508695831281678860513712181333729
590436549601884994184519097657397061550328468140875020630728462
564230141908136118998207860731109237365075922240988708341841893
093842374187901309709249334375710994985735557672788610080629408
201517155444973479370259713482031022520308410577644943311952468
238661819696470413701424802298593548776978832108656234910264109
205911593318386075749335920440095933187160459299339643209960996
544012564454057364306563040653021522038450273701898168250233560
542331594468641179158138428277349731703737219963018417632363299
494386783276795049901144295223363777310171682399431489512674283
722166638871500680396177707298901665698448682925479970938945488
015207286373345737063021706179180185939364399949813105084680637
909826757786706357829130980704081337842853100001277249846738997
878582385201810484324518892638117928226308648220890311998167122
766497072354298239001901713874648443916904411649363916870517743
627139496012306231401783236358878349778653017133170776393229190
962899031016050228081413485309678808196586762986239037632308148
814472095156234891O2340]
[
-358669728503033965955046155185932503931883887379872277455118210
839102543863181438534347689798327235493745396550354862873325284
123204745893692383498832669526276786683673904466835238639111088
176473066762370230956445440152466475196273329776775660504684381
821573429292001039205618744405254772514486944264759109061833637
162688648567335403604919306230696926882577929210556910121365156
592500723474741715873211918157674895500365603131503393909304553
543668772530242757398365829942964382287353102256878852782630392
556077564643767517982473448420357984617974602745319923721806502
767642883521062604109504173422218317030348342025196503076989619
423286320595594532132413487721321345495318953431975504278003666
387349790717313637250573233398872198367767305295605199794826409
961472518999188334798022792864858155348596929474606026648034200
190224710214923347890071014521348202622213279868279871409516214
318179324074936945809504651484175647749810818574041046595389734
992634508545260220249543827081356523822542453738308655755860022
906128149236636770378074701488721310159346866344734332950029141
504192997580817878689368136443866464498705094242183375719200623
882979359169576801457318115952795697391410668510379078300665609
927204709806847601537870675114213815571880306043582436810501121
571883700854830653937068812916727205063269066865426830426904725
008774912463697447837847618627158298698796088264999801798009135
```

```
158996835973189146295769097080900270401304387569951623422424605
728261881102222892659751843527102638515452202500931719814414012
671488664918093548540142593796268318225988095430865948238055874
626873391117067578140701051723867488426343262197157754419061778
217172274961913302352783648798965584644483165038286859822860523
326826430555729456537313352688125566845790002733763072791819915
558944590663637325499801636054483317170105905593324282299894783
992548153505364195296707498890926568747883424771896925615428218
790356827843121796325442426029433988002989914353904213390818854
900153897673520979223287898158896347287821180441205938512652684
913236600002879915013168902752334850484291447438752544708294311
066175496799744632656629037571881596151155659635498334562768209
299436003535895332602149069735256932909673372314773009978374424
683040302147427806569828438714089672217900345012788326328335017
2213821202871617596056917140884160335108012469/813556575096842
662323465734166568903533796530662672135627852623984739977713256
346971147133423105819128603495037501907205022115964134854702542
330135628278984381908438019892260511726565976650415661750202319
000924967707390508064437833551767556092553465026574006697641664
467830331984043223720486956578416573748181646963552804348979519
225831243427276535070377392769243443535738415007940928788395130
662659812651844124668253773690614455495066332442126760009483009
017444006601592084071907868512983411981479723005704637263559527
992092483307234110816230960546886958896866367595083916568942794
151913647907149922912144845642759706942531914835956935815296060
208336097733242683756570900074289800892925627430338655649722281
382108832690287595718868766429973989827488599498828238323766013
894132415343720586583471412089806758918326659444430924974009529
790045579609665680214601367146770884699996449500423551096193912
241762639311039349198018355803690082566967333767995924954286966
232553039416740755496440215787000637012160015754590089238782667
668808596254146584769169568132217566383818600105959479919980843
845851710170506983911275077365793380275443154731666973941987680
288689095270976545822879665844132150626865634073828737408913616
907854644782103625502538827825301739166256335772102742436266745
518087309920376998836903819531479412310065693628175004126145692
712846028381627223799641572146221847473015184448197741668368378
218768474837580261941849866875142198997147111534557722016125881
240303431088994695874051942696406204504061682115528988662390493
247732363939294082740284960459718709755395766421731246982052821
441182318663677215149867184088019186637432091859867928641992199
908802512890811472861312608130604304407690054740379633650046712
308466318893728235831627685655469946340747443992603683526472659
298877356655359009980228859044672755462034336479886297902534856
744433277743001360792355414597803331396897365850959941877890977
603041457274669147412604341235836037187872879989962621016936127
181965351557341271565826196140816267568570620000255449969347799
375716477040362096864903778527623585645261729644178062399633424
953299414470859647800380342774929688783380882329872783374103548
325427899202461246280356647271775669955730603426634155278645838
792579806203210045616282697061935761639317352597247807526461629
```

1628944190312469782046 80]
[
948456083330971331734018881781764580588320909786306044139224842
628856606299168225592655383516946510320600786952561285584383672
237044455792884554250280031801603374349160229430631204927786609
608058260137368298257755184262696481878833634211368529660852154
247612201575741848139194292482051768721430018527906012269426488
826574751349479864156680219982778322980834858523016491290812880
642799801410027988732992323132661816612703280195336359929238106
150235396730932129375230466804467871481290704951816259796110040
083648621958399199458833676631912580890217085217322759160045905
579977128805195393970664633080162475901169418521259673165758670
416259646479821673341594414125367529471312245190727413975078825
236783797643408549511468785705006661025208333673751505027220995
118759319851294597060602544848423704890243733555676538879762905
064692469124926023290781715520535875387470931683441759534773491
178680825934364775560327827993804192835284129963776972425006185
116977877149950623710209284860395553310724156927742081214222212
894254156260975960627977637937664775325312194035175587094218978
290230320131720054621330405559654727322954325832846613693455123
800511141950236266559376215576538963286445503102366695147124237
397004384388419162642901597507898983470661842720515800673527477
690648060161064492509316006695322000698606542204922621200825216
339885718786936207969256237041382613823161350119119429459091502
087963748646415195807484336944765521070550962312638973564387487
956617645497124616853081094950235251864257209358302822682381838
741391151438679870522123091154030115425650722203916304252746847
728556538800193064808839182079147256993832713170403686970379758
850055765710815089206639349980448370748033579146000230115852408
995510261420620024711857820370964177245278356271585944194739806
330636050397092879438893520526866531369671015628647982314388141
390391151262584465141695984982364444981061689940426402678696988
628623163181399562948689541589417374862328392215950494778011076
168915288975095394927862912695483022350031666986962394011941027
862267001848167890754404152582912592274949659557728846373227036
141101871946136303255638946038956530286598425547576754297880487
909166842506531697578529691272393821106179926353716178985129634
966201903031749806863622741998706119661273804076243986703538996
761962783334570101233626149346796901551547014067782875484212320
116173286708328445176689826533133606781392631199236998885662809
348557356671155290956430174751875095360251105798206742735127104
506781413949219095421900994613025586328298832520783087510115995
046248385369525403221891677588377804627673251328700334882083228
391516599202161186024347828920828687409082348177640217448975984
291562171363826753518869638462172176786920750397046439419756522
132990632592206233412688684530722774753316622106338000474150486
872200330079604203595393425649170599073986150285231863177976366
604624165361705540811548027344347944843318379754195828447139714
595682395357496145607242282137985347126595741797846790764803034
416804886662134187828545003714490044646281371516932782486114054
105441634514379785943438321498699491374429974941411916188300692

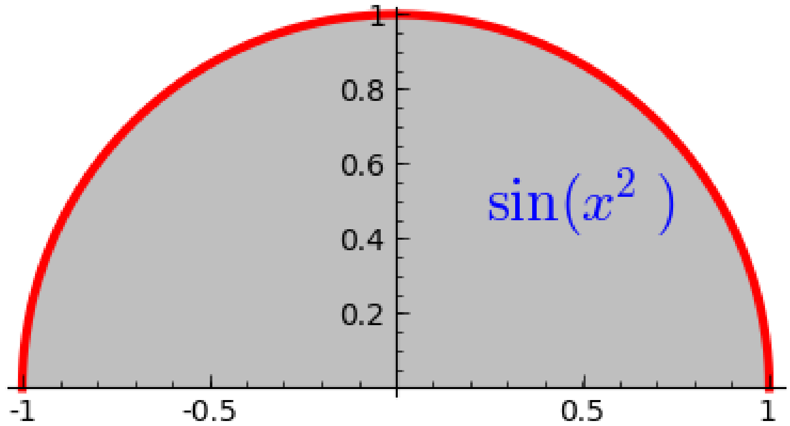
```
706620767186029329173570604490337945916332972221546248700476460
502278980483284010730068357338544234999822475021177554809695612
088131965551967459900917790184504128348366688399796247714348331
627651970837037774822010789350031850608000787729504461939133380
440429812707329238458478406610878319190930005297973995999042181
292585508525349195563753868289669013772157736583348697099384043
434454763548827291143983292206607531343281703691436870445680812
392732239105181275126941391265086958312816788605137121813337291
904365496018849941845190976573970615503284681408750206307284620
642301419081361189982078607311092373650759222409887083418418935
938423741879013097092493343757109949857355576727886100806294081
015171554449734793702597134820310225203084105776449433119524680
386618196964704137014248022985935487769788321086562349102641095
059115933183860757493359204400959331871604592993396432099609960
440125644540573643065630406530215220384502737018981682502335606
423315944686411791581384282773497317037372199630184176323632993
943867832767950499011442952233637773101716823994314895126742837
221666388715006803961777072989016656984486829254799709389454885
152072863733457370630217061791801859393643999498131050846806375
098267577867063578291309807040813378428531000012772498467389977
785823852018104843245188926381179282263086482208903119981671229
664970723542982390019017138746484439169044116493639168705177438
271394960123062314017832363588783497786530171331707763932291900
628990310160502280814134853096788081965867629862390376323081486
1447209515623489102340]
```

full_output.txt

Symbolic Calculus

Symbolic Calculus makes use of **Maxima** and **Ginac** under the hood.

```
var('x')
f = sqrt(1 - x^2)
plot(f, thickness=3, color='red', aspect_ratio=1, fill=True,
figsize=4) + text(r"$\sin(x^2)$", (.5,.5), fontsize=20)
```




```
var('t')
assume(t+1 > 0)
g = integrate(f, (x, -1, t)); show(g)
```

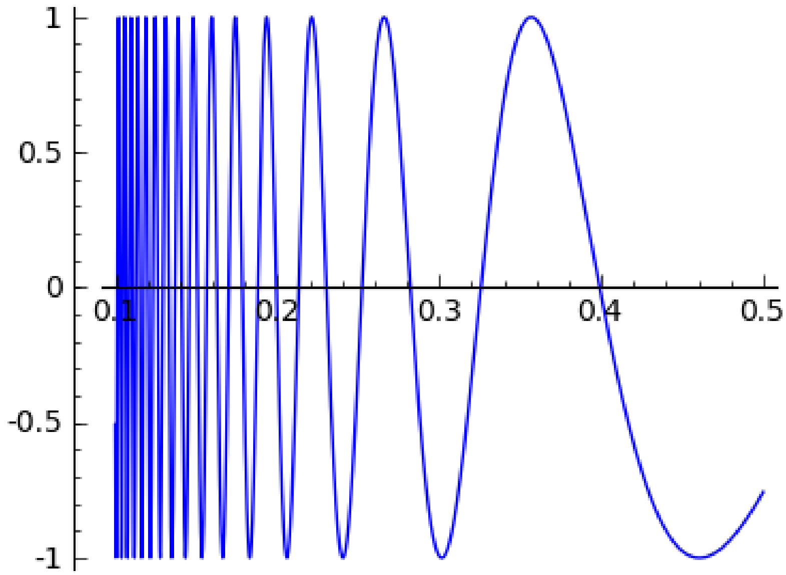
$$\frac{1}{4}\,\pi + \frac{1}{2}\,\sqrt{-t^{2} + 1}t + \frac{1}{2}\,\arcsin\left(t\right)$$

```
show(g(t=1) - g(t=-1))
```

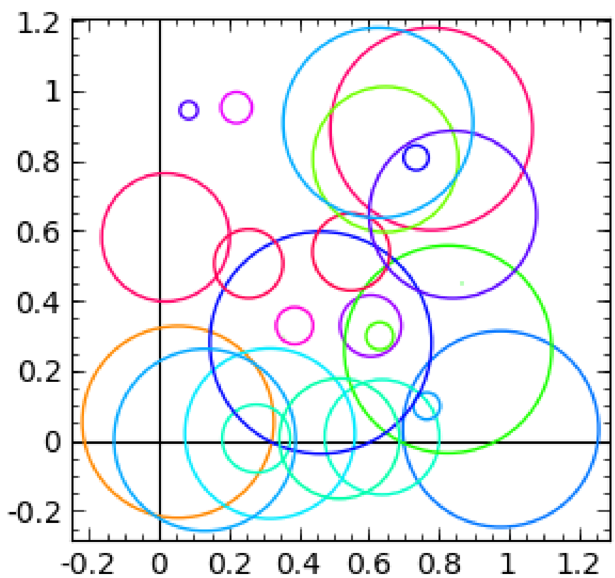
$$\frac{1}{2}\,\pi$$

Plotting in 2D

```
plot(sin(1/x^2), (x,.1,.5), figsize=4)
```




```
G = sum(circle((random(), random()), random()/3,
        color=hue(random())) for i in range(25))
G.show(aspect_ratio=1, frame=True, figsize=4)
```



Plotting in 3D

```
f(x,y) = sin(x - y) * y * cos(x)
time g = plot3d(f, (x,-3,3), (y,-3,3), opacity=.9, color='red',
figsize=3)
```

Time: CPU 0.00 s, Wall: 0.00 s

```
g
```

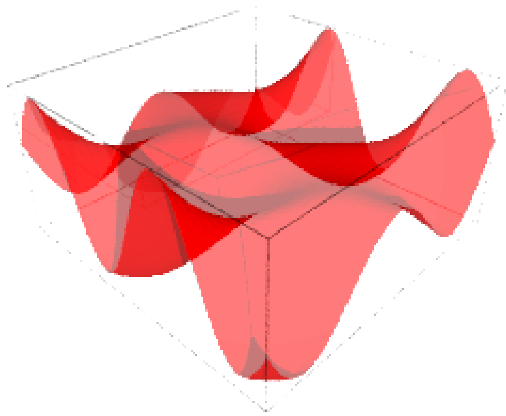
Get Image

```
icosahedron(opacity=.7, color='red') +
cube(opacity=.4,color='green')
```

Get Image

```
# user viewer='tachyon' on the ipad...
```

```
f(x,y) = sin(x - y) * y * cos(x)
plot3d(f, (x,-3,3), (y,-3,3), opacity=.9, color='red', figsize=3,
viewer='tachyon')
```



```
G = sum(sphere((random(), random(), random()), random()/4,
        color=hue(random()), opacity=.6)
        for i in range(20))
G.show(aspect_ratio=1, frame=True, figsize=3)
```

Get Image

Plotting a 3D Model

See http://www.davidson.edu/math/chartier/Starwars/

```
# Yoda! (53,756 vertices)

from scipy import io
yoda = io.loadmat(DATA + 'yodapose.mat')

from sage.plot.plot3d.index_face_set import IndexFaceSet
V = yoda['V']; F3=yoda['F3']-1; F4=yoda['F4']-1
Y = IndexFaceSet(F3,V,color=Color('#444444')) +
IndexFaceSet(F4,V,color=Color('#007700'))
Y = Y.rotateX(-1)
Y.show(aspect_ratio=1, frame=False, zoom=1.2)
```

Get Image

Questions

????

```
search_doc('ode_solver')
```

Search Documentation: "ode_solver"

1. reference/genindex-I.html
2. reference/genindex-O.html
3. reference/genindex-P.html
4. reference/genindex-all.html
5. reference/sage/calculus/desolvers.html
6. reference/sage/gsl/ode.html

If you made it this far, next try the Sage Tutorial.