

Title: Adding Spice to Your Research with Sage: Open Source Software for Mathematics

Date: May 10, 2011 09:30 AM

URL: <http://pirsa.org/11050014>

Abstract: Sage is a collection of mature open source software for mathematics, and new code, all unified into one powerful and easy-to-use package. The mission statement of the Sage project is: "Creating a viable free open source alternative to Magma, Maple, Mathematica and Matlab."

More information is available at www.sagemath.org. I will use the Sage notebook (a web interface) to demonstrate the use of Sage for a variety of mathematical problems and comment on its design and future direction.

Adding Spice to Your Research with Sage

Save Save & quit Discard & quit

last edited on May 09, 2011 07:12 PM by admin

File... Action... Data... sage ☐ Typeset

[Print](#) Worksheet Edit Text Undo Share Publish

Adding Spice to Your Research with Sage Open Source Software for Mathematics

May 10, 2011

Rob Beezer

University of Puget Sound

Perimeter Institute for Theoretical Physics
Waterloo, Canada

Perimeter Institute for Theoretical Physics Waterloo, Canada

1 What is Sage?

- Sage: Open source software for mathematics
- A computer algebra system
- A "distribution" of software for mathematics
- A digital blackboard
- Mission: "Creating a viable free open source alternative to Magma, Maple, Mathematica and Matlab."

2 Who am I?

- "Pure" mathematician - algebra, discrete mathematics
- Teacher at an undergraduate liberal arts college
- Active Sage developer - linear algebra, group theory, graph theory

1 What is Sage?

- Sage: Open source software for mathematics
- A computer algebra system
- A "distribution" of software for mathematics
- A digital blackboard
- Mission: "Creating a viable free open source alternative to Magma, Maple, Mathematica and Matlab."

2 Who am I?

- "Pure" mathematician - algebra, discrete mathematics
- Teacher at an undergraduate liberal arts college
- Active Sage developer - linear algebra, group theory, graph theory
- Interested in integrating Sage into the classroom (NSF education grant)
- Sage Lecturer, African Institute of Mathematical Sciences, Fall 2010

- Interested in integrating Sage into the classroom (NSF education grant)
- Sage Lecturer, African Institute of Mathematical Sciences, Fall 2010







3 Components

- Pynac for symbolic expressions

```
var('t')  
f(t) = t*tan(t^3)  
f
```

```
var('s')  
f(s)
```


3 Components

- Pynac for symbolic expressions

```
var('t')  
f(t) = t*tan(t^3)  
f
```

```
var('s')  
f(s)
```

- Maxima for derivatives, antiderivatives

```
fprime(t) = f.derivative()  
fprime
```

- matplotlib for 2-D graphics

```
fprime.plot()
```

3 Components

- Pynac for symbolic expressions

```
var('t')  
f(t) = t*tan(t^3)  
f
```

```
var('s')  
f(s)
```

- Maxima for derivatives, antiderivatives

```
fprime(t) = f.derivative()  
fprime
```

- matplotlib for 2-D graphics

```
fprime.plot()
```

3 Components

- Pynac for symbolic expressions

```
var('t')
f(t) = t*tan(t^3)|
f
```

[evaluate](#)

```
var('s')
f(s)
```

- Maxima for derivatives, antiderivatives

```
fprime(t) = f.derivative()
fprime
```

- matplotlib for 2-D graphics

```
fprime.plot()
```

3 Components

- Pynac for symbolic expressions

```
var('t')
f(t) = t*tan(t^3)
f
```

```
t |--> t*tan(t^3)
```

```
var('s')
f(s)
```

[evaluate](#)

- Maxima for derivatives, antiderivatives

```
fprime(t) = f.derivative()
fprime
```

- matplotlib for 2-D graphics

```
f.simplify()
f.plot()
```

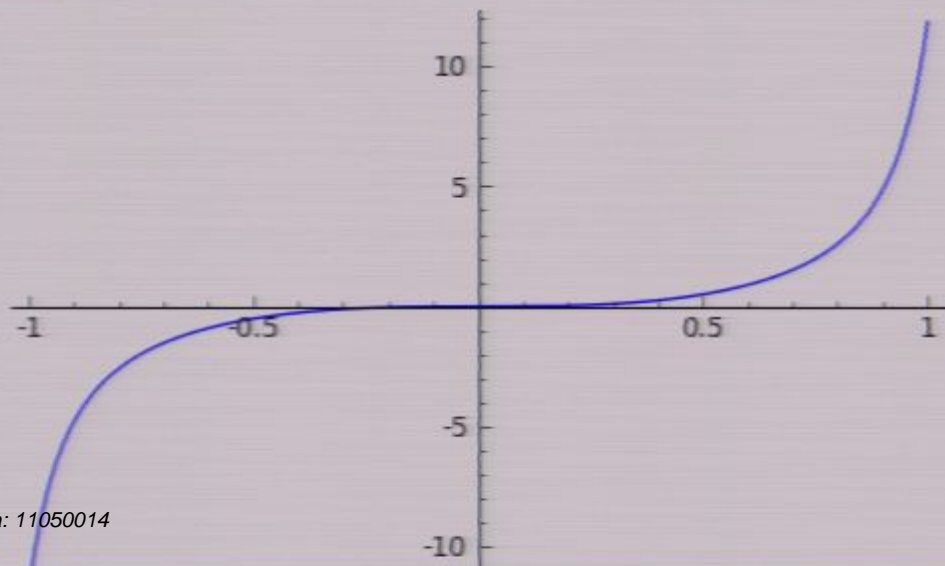

- Maxima for derivatives, antiderivatives

```
fprime(t) = f.derivative()  
fprime
```

[evaluate](#)

- matplotlib for 2-D graphics

```
fprime.plot()
```



- Maxima for derivatives, antiderivatives

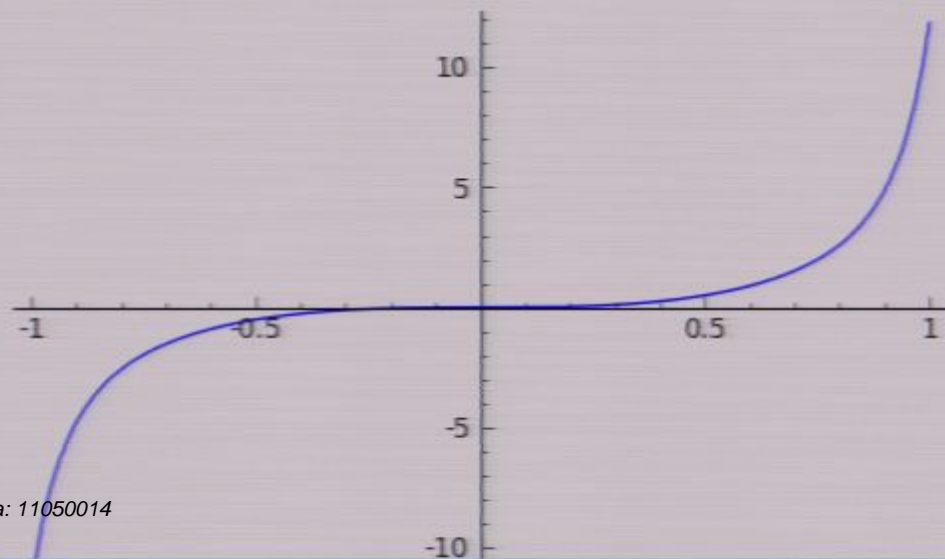
```
fprime(t) = f.derivative()  
fprime
```

```
t |--> 3*(tan(t^3)^2 + 1)*t^3 + tan(t^3)
```

- matplotlib for 2-D graphics

```
fprime.plot()
```

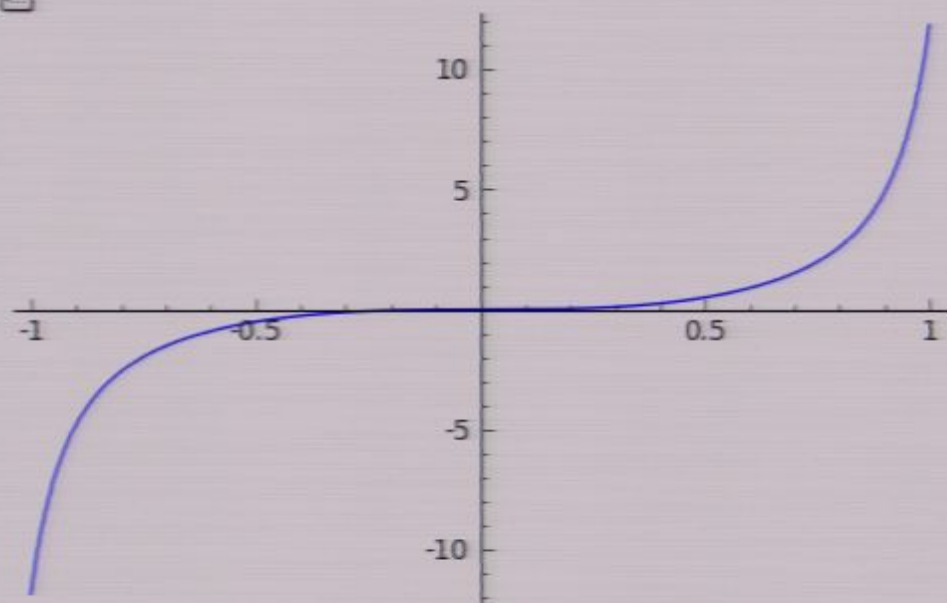
evaluate



• matplotlib for 2-D graphics

```
fprime.plot()
```

evaluate



• GAP (Groups, Algorithms, Programming) for group theory

```
G = DihedralGroup(10)
```

```
G.center()
```

- matplotlib for 2-D graphics

```
fprime.plot()
```

- GAP (Groups, Algorithms, Programming) for group theory

```
G = DihedralGroup(10)
G.center()
```

[evaluate](#)

- BLAS, ATLAS, LAPACK, NumPy, SciPy for linear algebra

```
A = matrix(QQ, [[ 3,  2,  0,  1],
                 [ 2,  1,  1,  0],
                 [ 1, -1,  5, -3],
                 [-2, -3,  5, -4]])
```

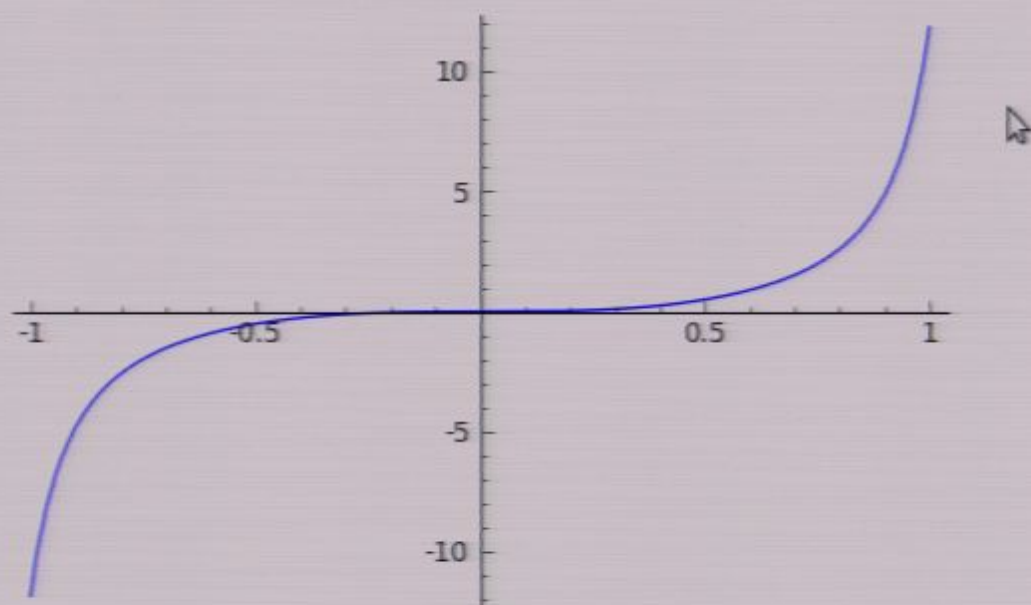
```
A.echelon_form()
```

```
K = A.kernel()
```

```
K
```

• matplotlib for 2-D graphics

```
fprime.plot()
```



• GAP (Groups, Algorithms, Programming) for group theory

```
G = DihedralGroup(10)  
G.center()
```


- GAP (Groups, Algorithms, Programming) for group theory

```
G = DihedralGroup(10)
G.center()
```

[evaluate](#)

- BLAS, ATLAS, LAPACK, NumPy, SciPy for linear algebra

```
A = matrix(QQ, [[ 3,  2,  0,  1],
                 [ 2,  1,  1,  0],
                 [ 1, -1,  5, -3],
                 [-2, -3,  5, -4]])
A.echelon_form()
```

```
K = A.kernel()
K
```

```
K.dimension()
```

```
A.eigenvalues()
```


• GAP (Groups, Algorithms, Programming) for group theory

```
G = DihedralGroup(10)
G.center()
```

Permutation Group with generators [(1,6)(2,7)(3,8)(4,9)(5,10)]

• BLAS, ATLAS, LAPACK, NumPy, SciPy for linear algebra

```
A = matrix(QQ, [[ 3, 2, 0, 1],
                 [ 2, 1, 1, 0],
                 [ 1, -1, 5, -3],
                 [-2, -3, 5, -4]])
A.echelon_form()
```

[evaluate](#)

```
K = A.kernel()
K
```

```
K.dimension()
```

• GAP (Groups, Algorithms, Programming) for group theory

```
G = DihedralGroup(10)
G.center()
```

Permutation Group with generators [(1,6)(2,7)(3,8)(4,9)(5,10)]

• BLAS, ATLAS, LAPACK, NumPy, SciPy for linear algebra

```
A = matrix(QQ, [[ 3,  2,  0,  1],
                 [ 2,  1,  1,  0],
                 [ 1, -1,  5, -3],
                 [-2, -3,  5, -4]])
A.echelon_form()
```

[evaluate](#)

```
K = A.kernel()
K
```

```
K.dimension()
```

```
A.eigenvalues()
```

• BLAS, ATLAS, LAPACK, NumPy, SciPy for linear algebra

```
A = matrix(QQ, [[ 3, 2, 0, 1],
                [ 2, 1, 1, 0],
                [ 1, -1, 5, -3],
                [-2, -3, 5, -4]])
```

```
A.echelon_form()
```

[evaluate](#)

```
K = A.kernel()
K
```

```
K.dimension()
```

```
A.eigenvalues()
```

```
B = A.change_ring(RDF)
B.eigenvalues()
```

• BLAS, ATLAS, LAPACK, NumPy, SciPy for linear algebra

```
A = matrix(QQ,[[ 3,  2,  0,  1],
               [ 2,  1,  1,  0],
               [ 1, -1,  5, -3],
               [-2, -3,  5, -4]])
```

```
A.echelon_form()
```

[evaluate](#)

```
K = A.kernel()
K
```

```
K.dimension()
```

```
A.eigenvalues()
```

```
B = A.change_ring(RDF)
B.eigenvalues()
```


• BLAS, ATLAS, LAPACK, NumPy, SciPy for linear algebra

```
A = matrix(QQ, [[ 3, 2, 0, 1],
                 [ 2, 1, 1, 0],
                 [ 1, -1, 5, -3],
                 [-2, -3, 5, -4]])
```

```
A.echelon_form()
```

```
[ 1  0  2 -1]
[ 0  1 -3  2]
[ 0  0  0  0]
[ 0  0  0  0]
```

```
K = A.kernel()
K
```

[evaluate](#)

```
K.dimension()
```

```
A.eigenvalues()
```

```
B = A.change_ring(RDF)
B.eigenvalues()
```


• BLAS, ATLAS, LAPACK, NumPy, SciPy for linear algebra

```
A = matrix(QQ, [[ 3, 2, 0, 1],
                 [ 2, 1, 1, 0],
                 [ 1, -1, 5, -3],
                 [-2, -3, 5, -4]])
```

```
A.echelon_form()
```

```
[ 1  0  2 -1]
[ 0  1 -3  2]
[ 0  0  0  0]
[ 0  0  0  0]
```

```
K = A.kernel()
```

```
K
```

Vector space of degree 4 and dimension 2 over Rational Field

Basis matrix:

```
[ 1  0 -1  1]
[ 0  1 -4/5 3/5]
```

```
K.dimension()
```

[evaluate](#)

```
A.eigenvalues()
```

• BLAS, ATLAS, LAPACK, NumPy, SciPy for linear algebra

```
A = matrix(QQ, [[ 3,  2,  0,  1],
                 [ 2,  1,  1,  0],
                 [ 1, -1,  5, -3],
                 [-2, -3,  5, -4]])
```

```
A.echelon_form()
```

```
[ 1  0  2 -1]
[ 0  1 -3  2]
[ 0  0  0  0]
[ 0  0  0  0]
```

```
K = A.kernel()
```

```
K
```

Vector space of degree 4 and dimension 2 over Rational Field

Basis matrix:

```
[ 1  0 -1  1]
[ 0  1 -4/5 3/5]
```

```
K.dimension()
```

```
2
```

```
A.eigenvalues()
```

```
[ 1  0  2 -1]
[ 0  1 -3  2]
[ 0  0  0  0]
[ 0  0  0  0]
```

```
K = A.kernel()
K
```

Vector space of degree 4 and dimension 2 over Rational Field

Basis matrix:

```
[ 1  0 -1  1]
[ 0  1 -4/5 3/5]
```

```
K.dimension()
```

2

```
A.eigenvalues()
```

[evaluate](#)

```
B = A.change_ring(RDF)
B.eigenvalues()
```

• JMWL for 3D graphics

$$\begin{bmatrix} 1 & 0 & 2 & -1 \\ 0 & 1 & -3 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

```
K = A.kernel()
K
```

Vector space of degree 4 and dimension 2 over Rational Field

Basis matrix:

$$\begin{bmatrix} 1 & 0 & -1 & 1 \\ 0 & 1 & -4/5 & 3/5 \end{bmatrix}$$

```
K.dimension()
```

2

```
A.eigenvalues()
```

[0, 0, 0.2087121525220800?, 4.791287847477920?]

```
B = A.change_ring(RDF)
B.eigenvalues()
```

[evaluate](#)

`K = A.represent()`

`K`

Vector space of degree 4 and dimension 2 over Rational Field

Basis matrix:

$$\begin{bmatrix} 1 & 0 & -1 & 1 \\ 0 & 1 & -4/5 & 3/5 \end{bmatrix}$$

`K.dimension()`

2

`A.eigenvalues()`

$[0, 0, 0.2087121525220800?, 4.791287847477920?]$

`B = A.change_ring(RDF)`

`B.eigenvalues()`

[evaluate](#)

• JMOL for 3D graphics

`var('x y')`

`plot3d(x^2+3*y^2, (x, -2, 2), (y, -2, 2))`

Basis matrix:

```
[ 1  0 -1  1]
[ 0  1 -4/5 3/5]
```

K.dimension()

2

A.eigenvalues()

[0, 0, 0.2087121525220800?, 4.791287847477920?]

```
B = A.change_ring(RDF)
B.eigenvalues()
```

[evaluate](#)

• JMOL for 3D graphics

```
var('x y')
plot3d(x^2+3*y^2, (x, -2, 2), (y, -2, 2))
```


Basis matrix:

$$\begin{bmatrix} 1 & 0 & -1 & 1 \\ 0 & 1 & -4/5 & 3/5 \end{bmatrix}$$

K.dimension()

2

A.eigenvalues()

[0, 0, 0.2087121525220800?, 4.791287847477920?]

B = A.change_ring(RDF)

B.eigenvalues()

[4.79128784748, 0.208712152522, -1.53876545589e-15, -4.18343440413e-15]

JMOL for 3D graphics

```
var('x y')
plot3d(x^2+3*y^2, (x, -2, 2), (y, -2, 2))
```

[evaluate](#)

```
A.eigenvalues()
```

```
[0, 0, 0.2087121525220800?, 4.791287847477920?]
```

```
B = A.change_ring(RDF)
```

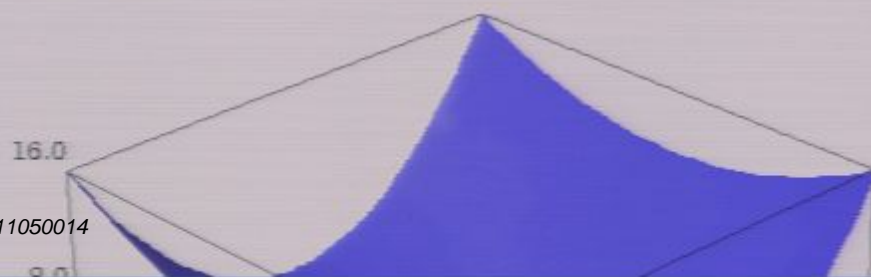
```
B.eigenvalues()
```

```
[4.79128784748, 0.208712152522, -1.53876545589e-15, -4.18343440413e-15]
```

• Jmol for 3D graphics

```
var('x y')  
plot3d(x^2+3*y^2, (x, -2, 2), (y, -2, 2))
```

[evaluate](#)



```
A.eigenvalues()
```

```
[0, 0, 0.2087121525220800?, 4.791287847477920?]
```

```
B = A.change_ring(RDF)
```

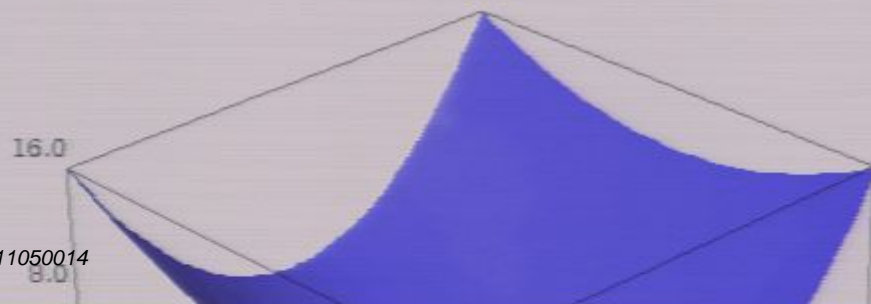
```
B.eigenvalues()
```

```
[4.79128784748, 0.208712152522, -1.53876545589e-15, -4.18343440413e-15]
```

• JMWL for 3D graphics

```
var('x y')
```

```
plot3d(x^2+3*y^2, (x, -2, 2), (y, -2, 2))
```



File Edit View History Bookmarks Tools Help

http://localhost:8000/home/admin/2388/

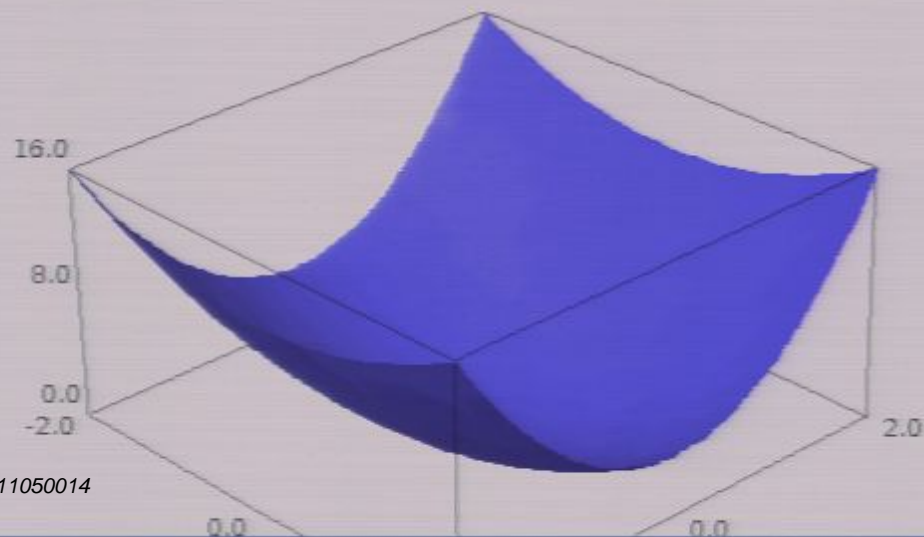
Home - Perimeter Institute ... Adding Spice to Your Resea...

B.eigenvalues()

[4.79128784748, 0.208712152522, -1.53876545589e-15, -4.18343440413e-15]

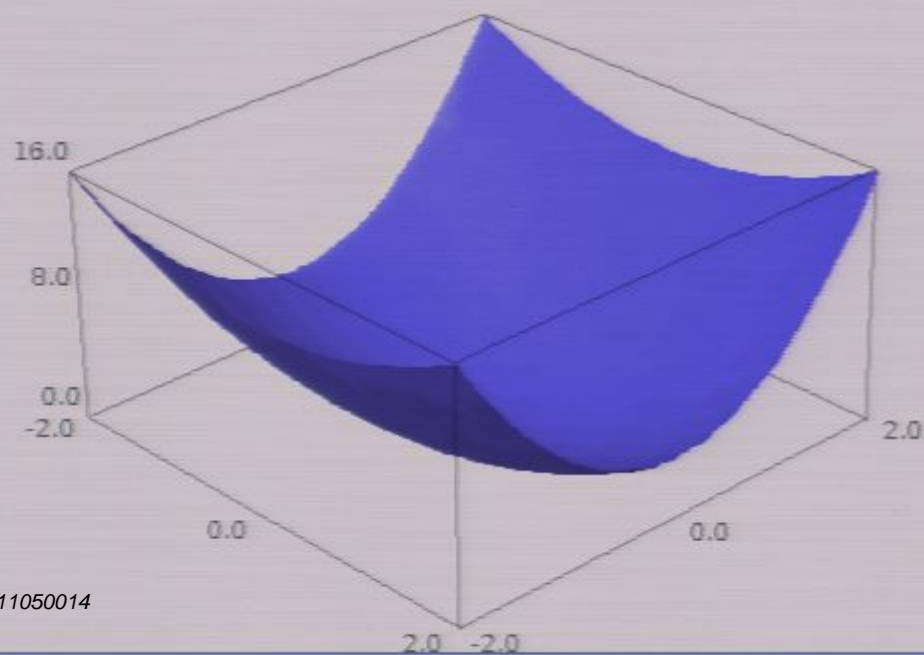
JMOL for 3D graphics

```
var('x y')  
plot3d(x^2+3*y^2, (x, -2, 2), (y, -2, 2))
```



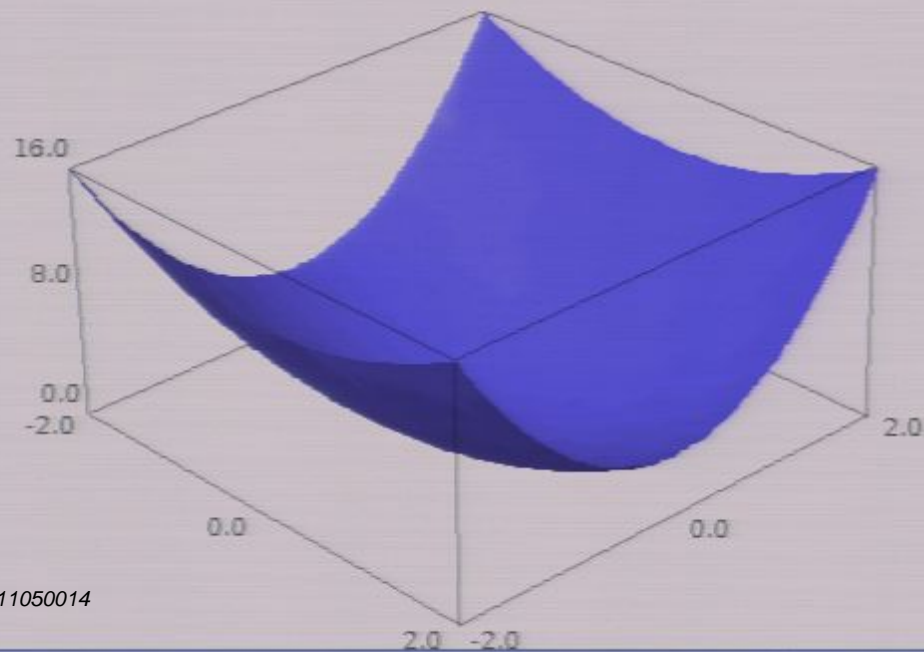
JMOL for 3D graphics

```
var('x y')  
plot3d(x^2+3*y^2, (x, -2, 2), (y, -2, 2))
```

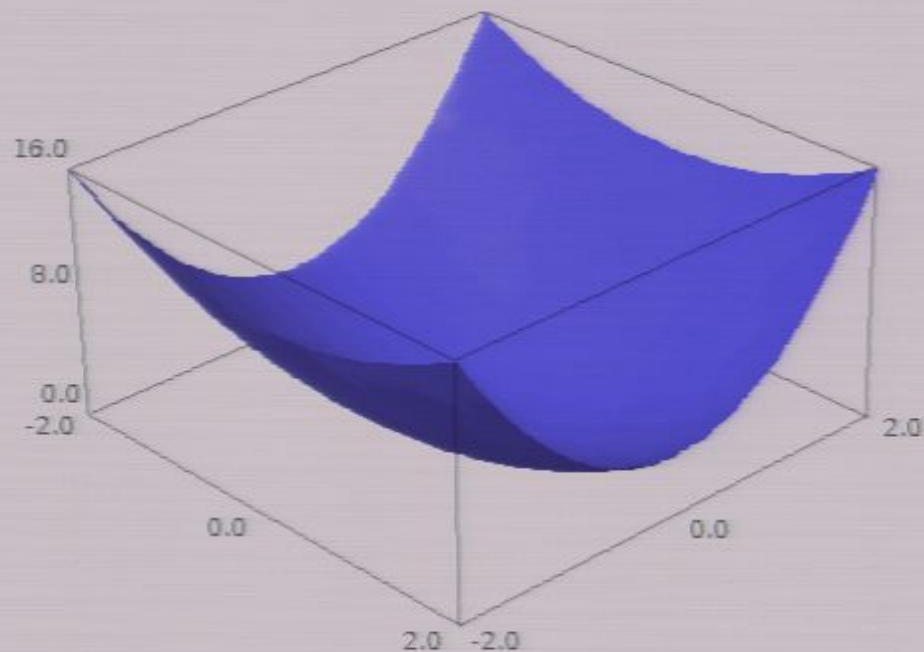


JMOL for 3D graphics

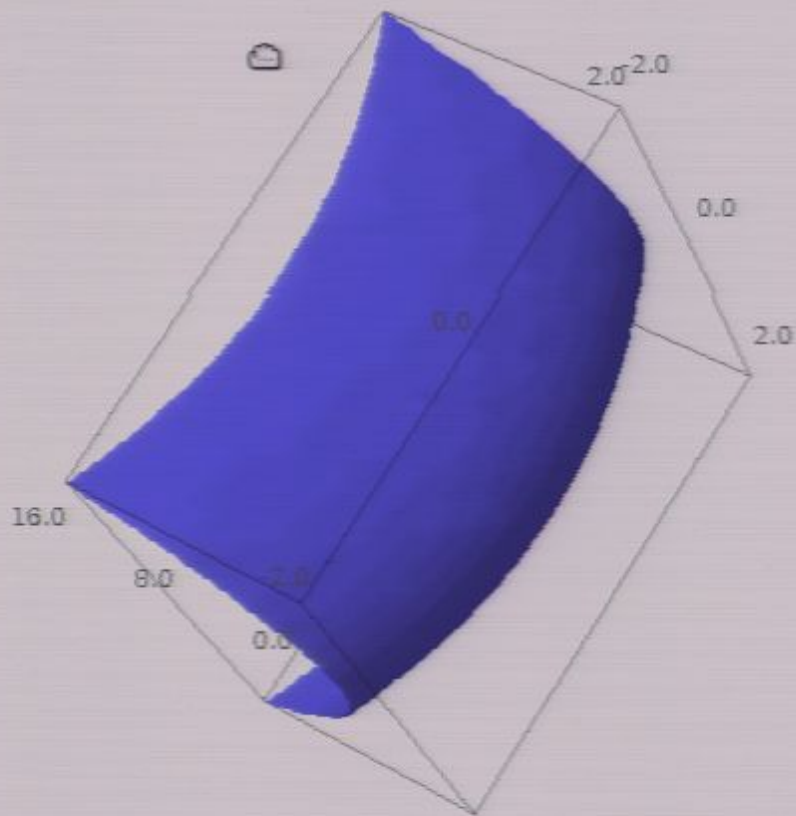
```
var('x y')  
plot3d(x^2+3*y^2, (x, -2, 2), (y, -2, 2))
```



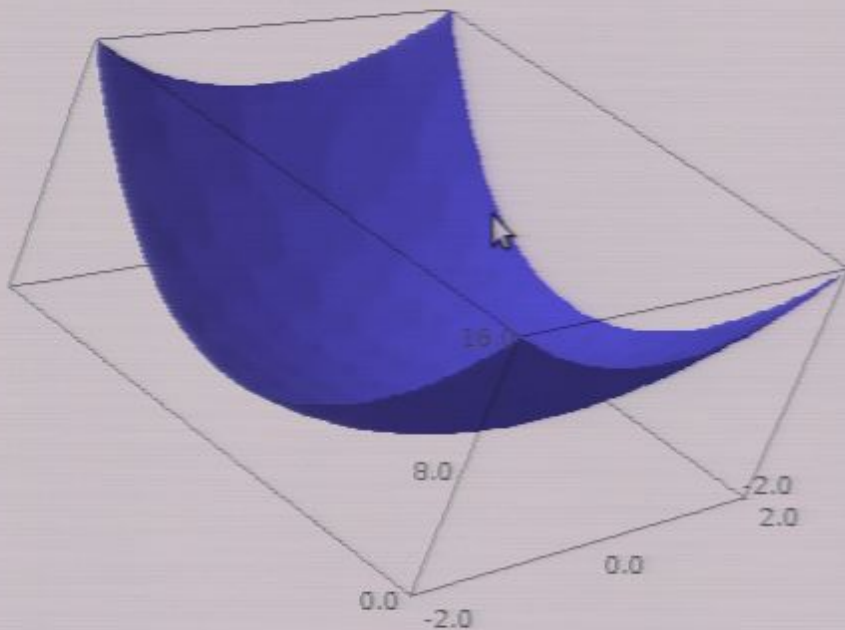

```
var('x y')  
plot3d(x^2+3*y^2, (x, -2, 2), (y, -2, 2))
```



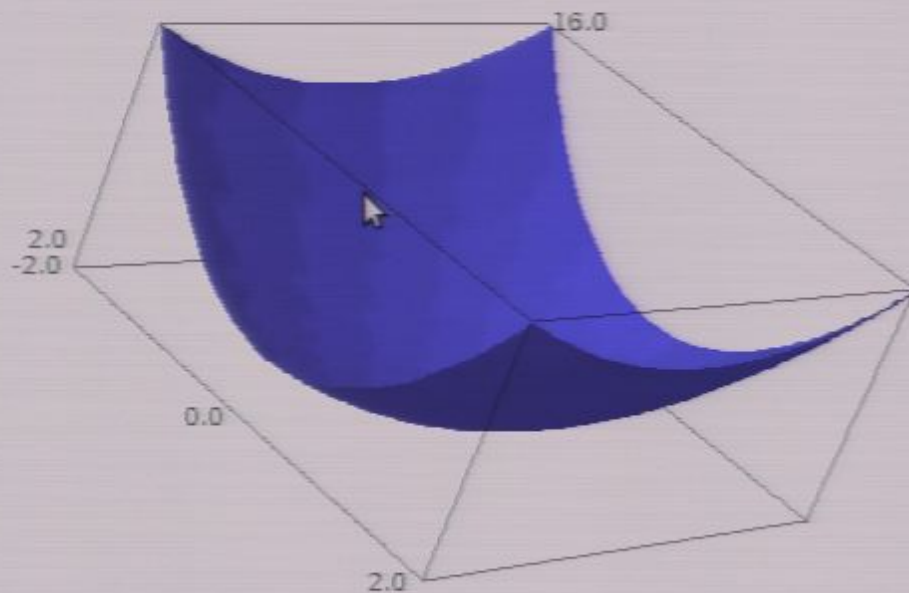
```
var('x y')  
plot3d(x^2+3*y^2, (x, -2, 2), (y, -2, 2))
```



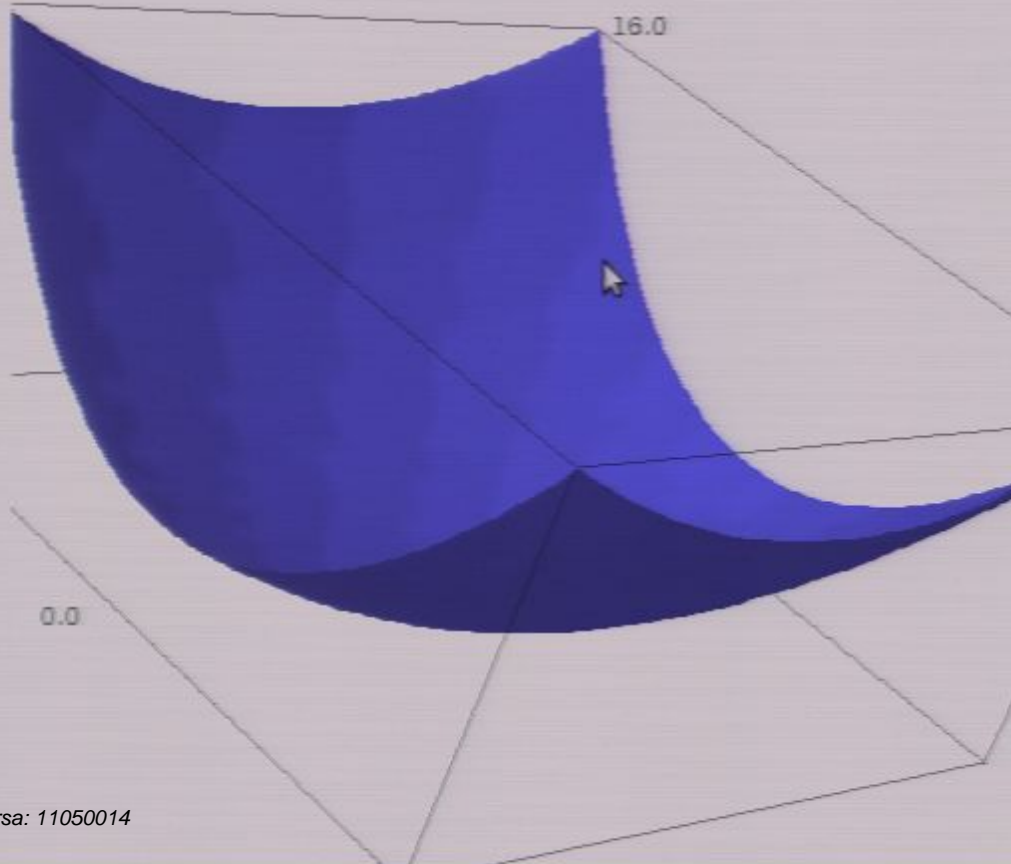
```
var('x y')  
plot3d(x^2+3*y^2, (x, -2, 2), (y, -2, 2))
```



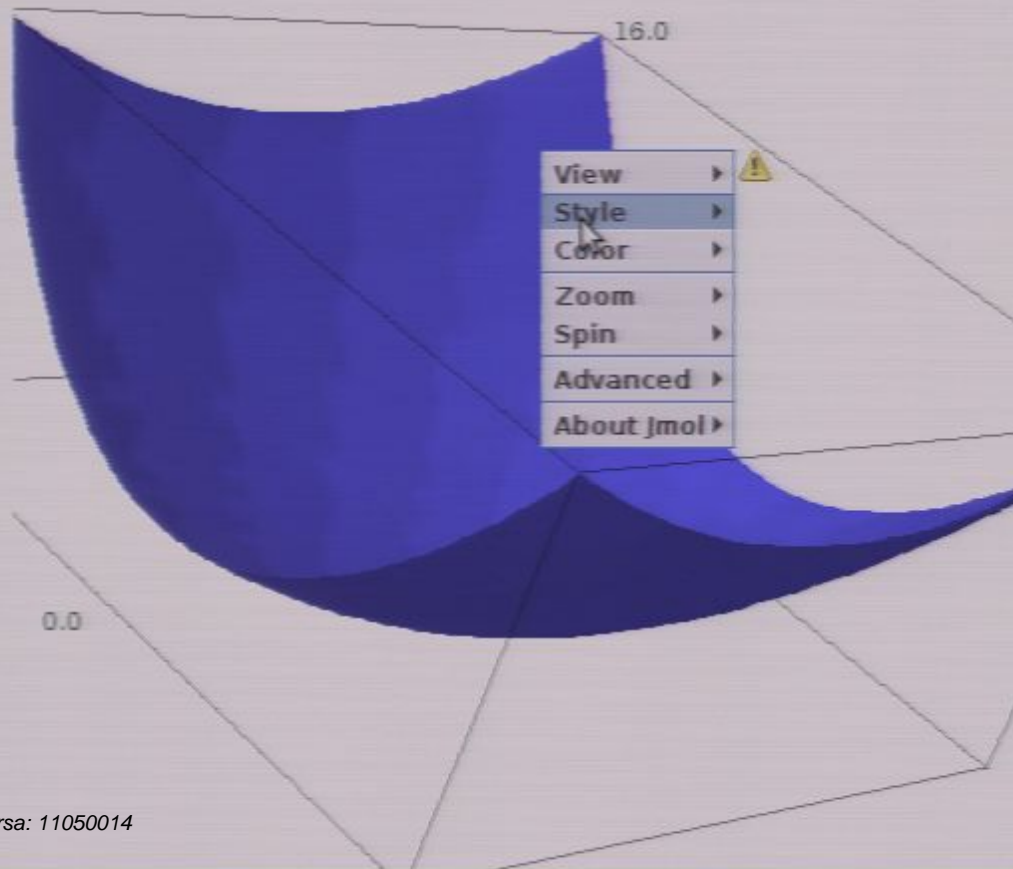

```
var('x y')  
plot3d(x^2+3*y^2, (x, -2, 2), (y, -2, 2))
```



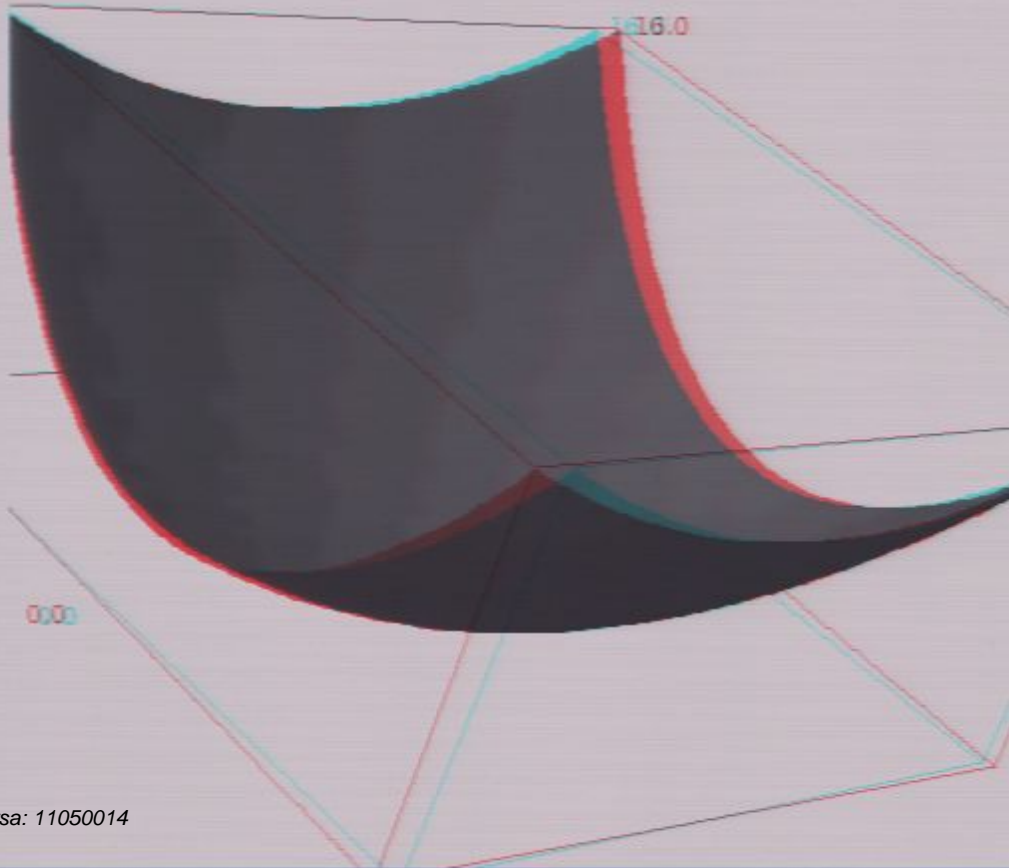
```
var('x y')  
plot3d(x^2+3*y^2, (x, -2, 2), (y, -2, 2))
```



```
var('x y')  
plot3d(x^2+3*y^2, (x, -2, 2), (y, -2, 2))
```



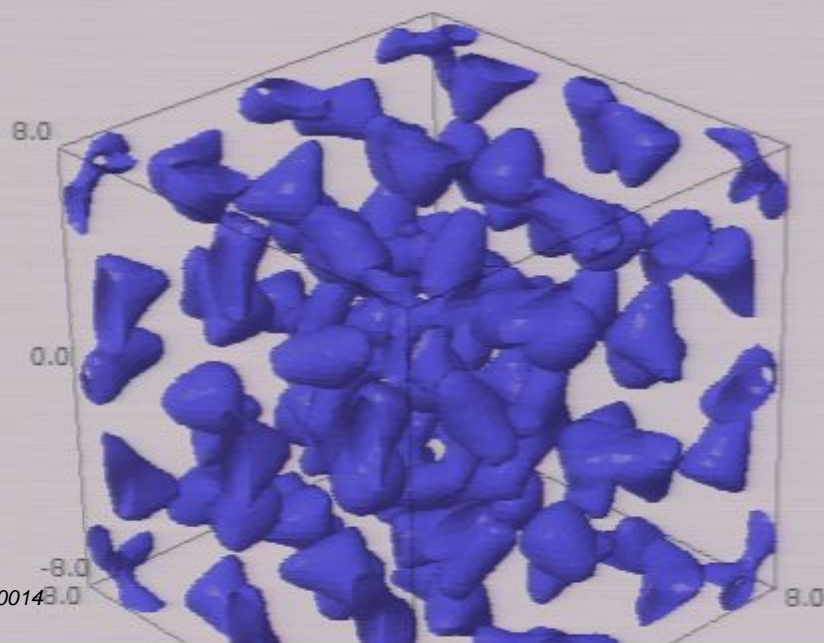

```
var('x y')  
plot3d(x^2+3*y^2, (x, -2, 2), (y, -2, 2))
```



[Get Image](#)

```
var('x y z')
T = RDF(golden_ratio)
p = 2 - (cos(x + T*y) + cos(x - T*y) + cos(y + T*z) + cos(y - T*z) + cos(z - T*x) + cos(z + T*x))
r = 4.77
implicit_plot3d(p, (x, -r, r), (y, -r, r), (z, -r, r), plot_points=40).show()
```

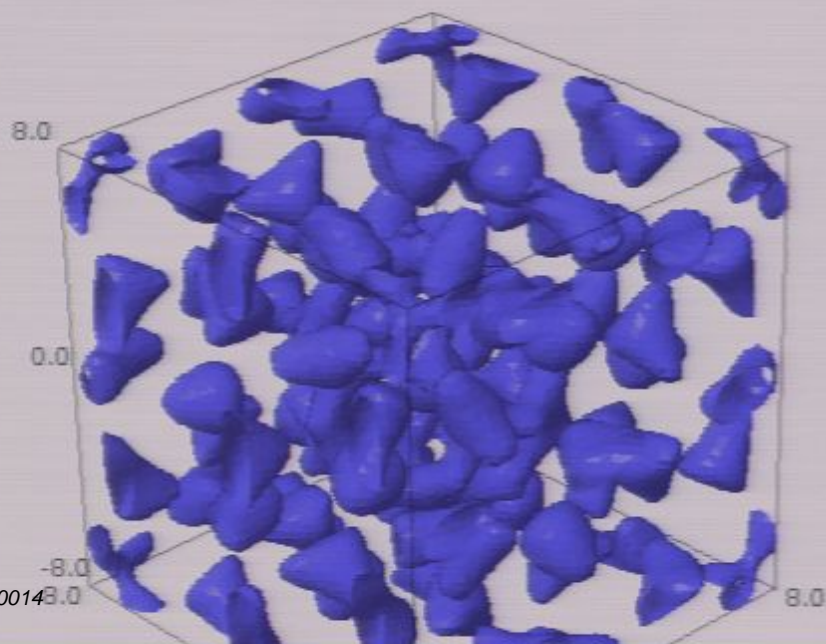
[evaluate](#)



[Get Image](#)

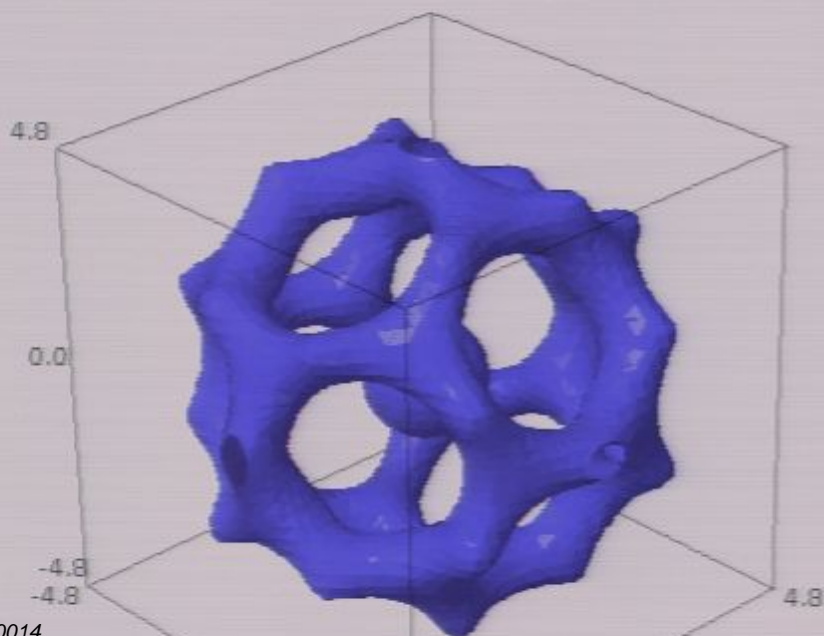
```
var('x y z')
T = RDF(golden_ratio)
p = 2 - (cos(x + T*y) + cos(x - T*y) + cos(y + T*z) + cos(y - T*z) + cos(z - T*x) + cos(z + T*x))
r = 4.77
implicit_plot3d(p, (x, -r, r), (y, -r, r), (z, -r, r), plot_points=40).show()
```

[evaluate](#)

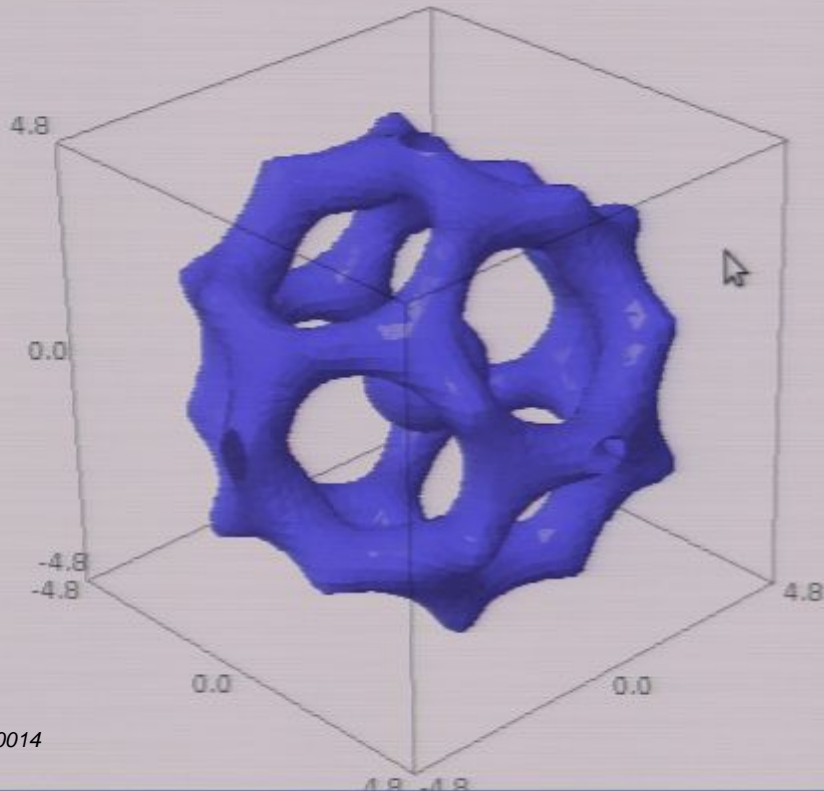


[Get Image](#)

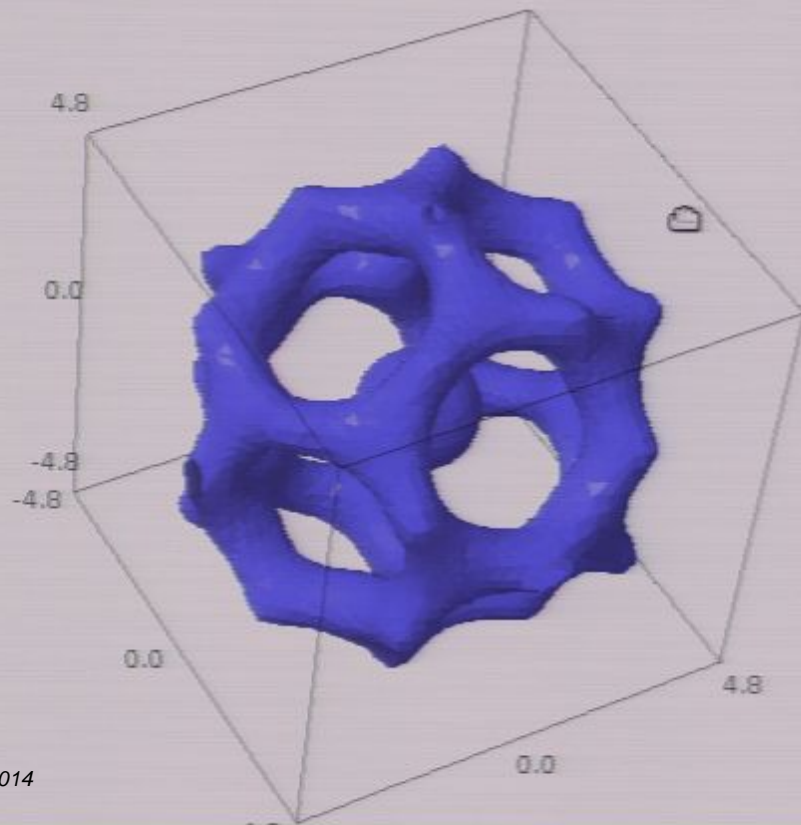
```
var('x y z')
T = RDF(golden_ratio)
p = 2 - (cos(x + T*y) + cos(x - T*y) + cos(y + T*z) + cos(y - T*z) + cos(z - T*x) + cos(z + T*x))
r = 4.77
implicit_plot3d(p, (x, -r, r), (y, -r, r), (z, -r, r), plot_points=40).show()
```



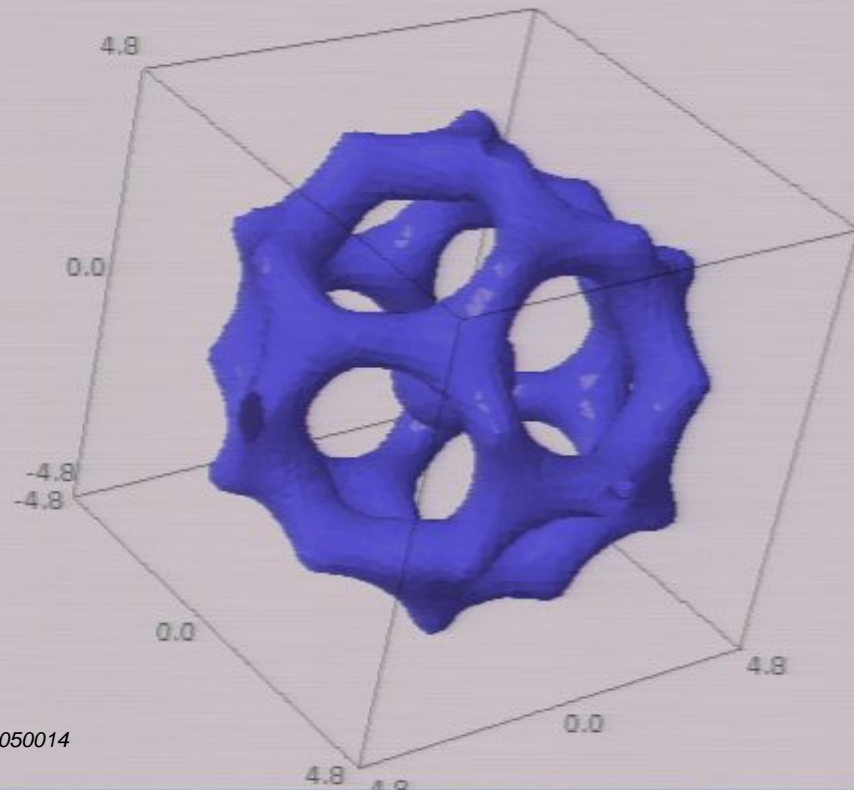
```
T = RUF(golden_ratio)
p = 2 - (cos(x + T*y) + cos(x - T*y) + cos(y + T*z) + cos(y - T*z) + cos(z - T*x) + cos(z + T*x))
r = 4.77
implicit_plot3d(p, (x, -r, r), (y, -r, r), (z, -r, r), plot_points=40).show()
```



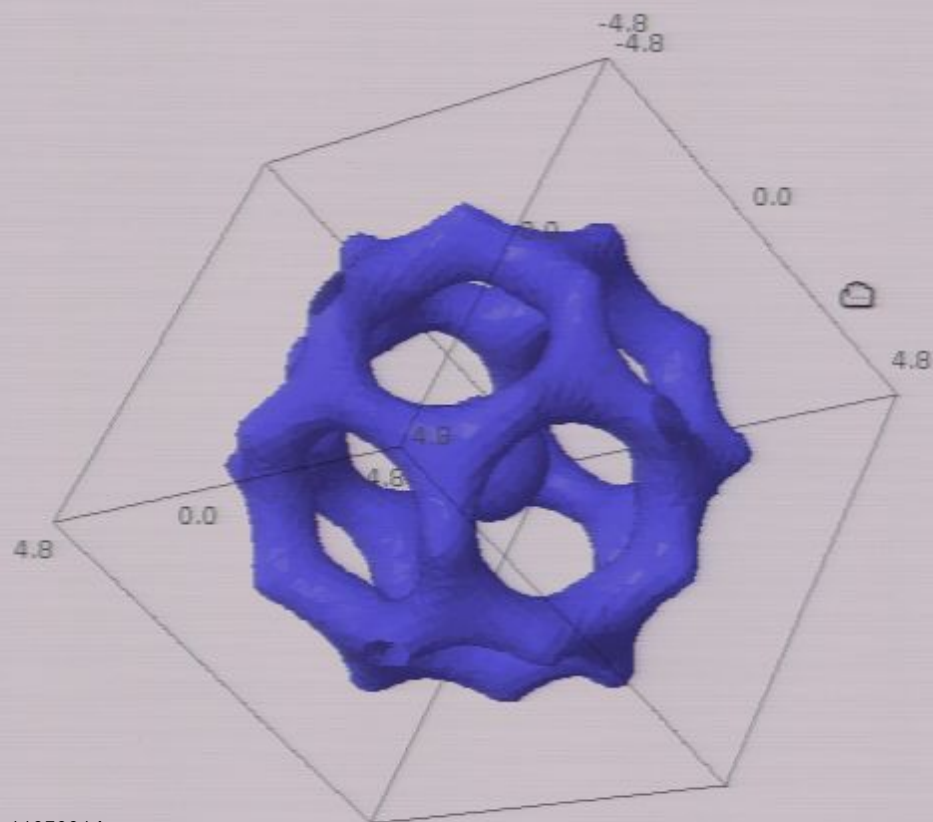
```
T = RUF(golden_ratio)
p = 2 - (cos(x + T*y) + cos(x - T*y) + cos(y + T*z) + cos(y - T*z) + cos(z - T*x) + cos(z + T*x))
r = 4.77
implicit_plot3d(p, (x, -r, r), (y, -r, r), (z, -r, r), plot_points=40).show()
```



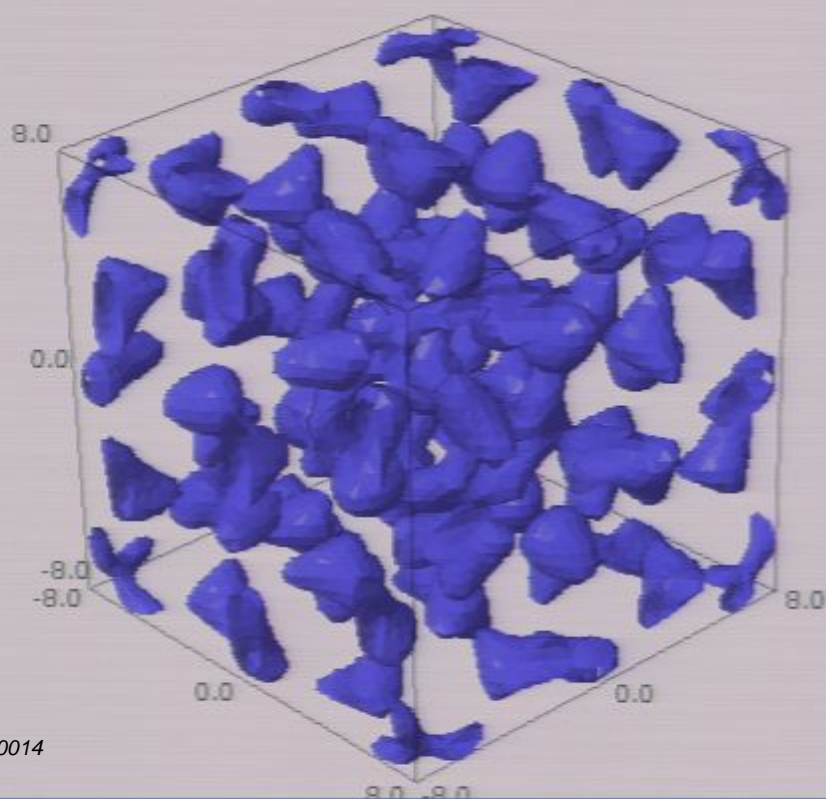

```
T = RDF(golden_ratio)
p = 2 - (cos(x + T*y) + cos(x - T*y) + cos(y + T*z) + cos(y - T*z) + cos(z - T*x) + cos(z + T*x))
r = 4.77
implicit_plot3d(p, (x, -r, r), (y, -r, r), (z, -r, r), plot_points=40).show()
```



```
T = RUF(golden_ratio)
p = 2 - (cos(x + T*y) + cos(x - T*y) + cos(y + T*z) + cos(y - T*z) + cos(z - T*x) + cos(z + T*x))
r = 4.77
implicit_plot3d(p, (x, -r, r), (y, -r, r), (z, -r, r), plot_points=40).show()
```



```
T = RUF(golden_ratio)
p = 2 - (cos(x + T*y) + cos(x - T*y) + cos(y + T*z) + cos(y - T*z) + cos(z - T*x) + cos(z + T*x))
r = 8
implicit_plot3d(p, (x, -r, r), (y, -r, r), (z, -r, r), plot_points=40).show()
```



R and GSL for probability and statistics

```
sage: data = [1.2, 3.6, 9.8, 5.4, 7.6]
sage: R_data = r.c([1.2, 3.6, 9.8, 5.4, 7.6])
sage: R_data.parent()
```

[evaluate](#)

```
sage: R_data.summary()
```

```
sage: python_data = R_data.sage()
sage: python_data
```

```
sage: sigma = 2
sage: T = RealDistribution('gaussian', sigma)
```

```
sage: T.get_random_element()
```

```
sage: T.cum_distribution_function(1)
```

```
sage: T.cum_distribution_function_inv(.95)
```

R and GSL for probability and statistics

```
sage: data = [1.2, 3.6, 9.8, 5.4, 7.6]
sage: R_data = r.c([1.2, 3.6, 9.8, 5.4, 7.6])
sage: R_data.parent()
```

[evaluate](#)

```
sage: R_data.summary()
```

```
sage: python_data = R_data.sage()
sage: python_data
```

```
sage: sigma = 2
sage: T = RealDistribution('gaussian', sigma)
```

```
sage: T.get_random_element()
```

```
sage: T.cum_distribution_function(1)
```

```
sage: T.cum_distribution_function_inv(.95)
```

[Get Image](#)

• R and GSL for probability and statistics

```
sage: data = [1.2, 3.6, 9.8, 5.4, 7.6]
sage: R_data = r.c([1.2, 3.6, 9.8, 5.4, 7.6])
sage: R_data.parent()
```

[evaluate](#)

```
sage: R_data.summary()
```

```
sage: python_data = R_data.sage()
sage: python_data
```

```
sage: sigma = 2
sage: T = RealDistribution('gaussian', sigma)
```

```
sage: T.get_random_element()
```

```
sage: T.cum_distribution_function(1)
```


[Get Image](#)

• R and GSL for probability and statistics

```
sage: data = [1.2, 3.6, 9.8, 5.4, 7.6]
sage: R_data = r.c([1.2, 3.6, 9.8, 5.4, 7.6])
sage: R_data.parent()
```

[evaluate](#)

```
sage: R_data.summary()
```

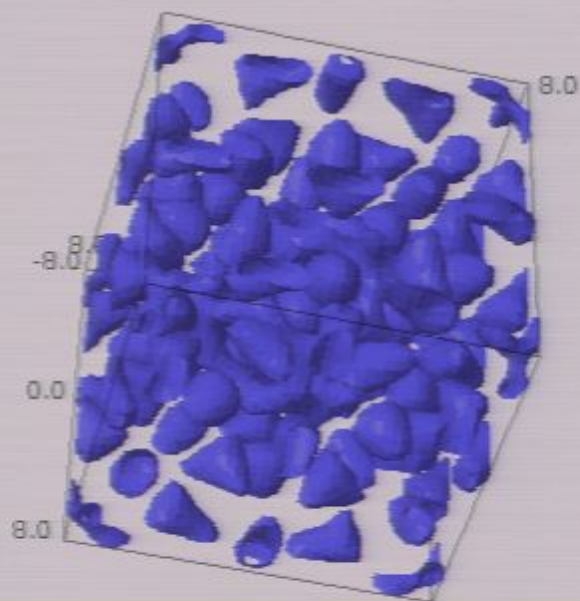
```
sage: python_data = R_data.sage()
sage: python_data
```

```
sage: sigma = 2
sage: T = RealDistribution('gaussian', sigma)
```

```
sage: T.get_random_element()
```

[Get Image](#)

```
var('x y z')
T = RDF(golden_ratio)
p = 2 - (cos(x + T*y) + cos(x - T*y) + cos(y + T*z) + cos(y - T*z) + cos(z - T*x) + cos(z + T*x))
r = 8
implicit_plot3d(p, (x, -r, r), (y, -r, r), (z, -r, r), plot_points=40).show()
```



[Get Image](#)

• R and GSL for probability and statistics

reset(|

[evaluate](#)

```
sage: data = [1.2, 3.6, 9.8, 5.4, 7.6]
sage: R_data = r.c([1.2, 3.6, 9.8, 5.4, 7.6])
sage: R_data.parent()
```

```
sage: R_data.summary()
```

```
sage: python_data = R_data.sage()
sage: python_data
```

```
sage: sigma = 2
sage: r = RealDistribution('gaussian', sigma)
```


[Get Image](#)

• R and GSL for probability and statistics

reset('r')

```
sage: data = [1.2, 3.6, 9.8, 5.4, 7.6]
sage: R_data = r.c([1.2, 3.6, 9.8, 5.4, 7.6])
sage: R_data.parent()
```

[evaluate](#)

```
sage: R_data.summary()
```

```
sage: python_data = R_data.sage()
sage: python_data
```

```
sage: sigma = 2
sage: r = RealDistribution('gaussian', sigma)
```

[Get Image](#)

• R and GSL for probability and statistics

```
reset('r')
```

```
sage: data = [1.2, 3.6, 9.8, 5.4, 7.6]
sage: R_data = r.c([1.2, 3.6, 9.8, 5.4, 7.6])
sage: R_data.parent()
```

[R Interpreter](#)

```
sage: R_data.summary()
```

[evaluate](#)

```
sage: python_data = R_data.sage()
sage: python_data
```

```
sage: sigma = 2
sage: T = RealDistribution('gaussian', sigma)
```

[Get Image](#)

- R and GSL for probability and statistics

```
reset('r')
```

```
sage: data = [1.2, 3.6, 9.8, 5.4, 7.6]
sage: R_data = r.c([1.2, 3.6, 9.8, 5.4, 7.6])
sage: R_data.parent()
```

R Interpreter

```
sage: R_data.summary()
```

[evaluate](#)

```
sage: python_data = R_data.sage()
sage: python_data
```

```
sage: sigma = 2
sage: T = RealDistribution('gaussian', sigma)
```

```
sage: T.get_random_element()
```


[Get Image](#)

• R and GSL for probability and statistics

```
reset('r')
```

```
sage: data = [1.2, 3.6, 9.8, 5.4, 7.6]
sage: R_data = r.c([1.2, 3.6, 9.8, 5.4, 7.6])
sage: R_data.parent()
```

[R Interpreter](#)

```
sage: R_data.summary()
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.20	3.60	5.40	5.52	7.60	9.80

```
sage: python_data = R_data.sage()
sage: python_data
```

[evaluate](#)

```
sage: sigma = 2
sage: T = RealDistribution('gaussian', sigma)
```

```
reset('r')
```

```
sage: data = [1.2, 3.6, 9.8, 5.4, 7.6]
sage: R_data = r.c([1.2, 3.6, 9.8, 5.4, 7.6])
sage: R_data.parent()
```

R Interpreter

```
sage: R_data.summary()
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.20	3.60	5.40	5.52	7.60	9.80

```
sage: python_data = R_data.sage()
sage: python_data
```

[evaluate](#)

```
sage: sigma = 2
sage: T = RealDistribution('gaussian', sigma)
```

```
sage: T.get_random_element()
```

```
sage: T.cum_distribution_function(1)
```

Adding Spice to Your Research with Sage -- Sage - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:8000/home/admin/2388/

Home - Perimeter Institute ... Adding Spice to Your Resea...

```
sage: data = [1.2, 3.6, 9.8, 5.4, 7.6]
sage: R_data = r.c([1.2, 3.6, 9.8, 5.4, 7.6])
sage: R_data.parent()
```

R Interpreter

```
sage: R_data.summary()
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.20	3.60	5.40	5.52	7.60	9.80

```
sage: python_data = R_data.sage()
sage: python_data
```

[evaluate](#)

```
sage: sigma = 2
sage: T = RealDistribution('gaussian', sigma)
```

```
sage: T.get_random_element()
```

```
sage: T.cum_distribution_function(1)
```

```
sage: T.cum_distribution_function_inv(.95)
```


Adding Spice to Your Research with Sage -- Sage - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:8000/home/admin/2388/

Home - Perimeter Institute ... Adding Spice to Your Resea...

```
sage: data = [1.2, 3.6, 9.8, 5.4, 7.6]
sage: R_data = r.c([1.2, 3.6, 9.8, 5.4, 7.6])
sage: R_data.parent()
```

R Interpreter

```
sage: R_data.summary()
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.20	3.60	5.40	5.52	7.60	9.80

```
sage: python_data = R_data.sage()
sage: python_data
```

```
[1.2, 3.6000000000000001, 9.8000000000000007, 5.4000000000000004,
 7.5999999999999996]
```

```
sage: sigma = 2
sage: T = RealDistribution('gaussian', sigma)
```

[evaluate](#)

```
sage: T.get_random_element()
```

```
sage: T.cum_distribution_function(1)
```

```
sage: R_data.summary()
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.20	3.60	5.40	5.52	7.60	9.80

```
sage: python_data = R_data.sage()
```

```
sage: python_data
```

```
[1.2, 3.6000000000000001, 9.8000000000000007, 5.4000000000000004, 7.5999999999999996]
```

```
sage: sigma = 2
```

```
sage: T = RealDistribution('gaussian', sigma)
```

[evaluate](#)

```
sage: T.get_random_element()
```

```
sage: T.cum_distribution_function(1)
```

```
sage: T.cum_distribution_function_inv(.95)
```

3.1 Other Packages

1.20 3.60 5.40 5.52 7.60 9.80

```
sage: python_data = R_data.sage()
sage: python_data
```

```
[1.2, 3.6000000000000001, 9.8000000000000007, 5.4000000000000004,
7.5999999999999996]
```

```
sage: sigma = 2
sage: T = RealDistribution('gaussian', sigma)
```

[evaluate](#)

```
sage: T.get_random_element()
```

```
sage: T.cum_distribution_function(1)
```

```
sage: T.cum_distribution_function_inv(.95)
```

3.1 Other Packages

- Singular - commutative and non-commutative algebra, algebraic geometry, and singularity theory

- Symmetrica - representation theory


```
sage: python_data
```

```
[1.2, 3.6000000000000001, 9.8000000000000007, 5.4000000000000004,  
7.5999999999999996]
```

```
sage: sigma = 2|
```

```
sage: T = RealDistribution('gaussian', sigma)
```

[evaluate](#)

```
sage: T.get_random_element()
```

```
sage: T.cum_distribution_function(1)
```

```
sage: T.cum_distribution_function_inv(.95)
```

3.1 Other Packages

- Singular - commutative and non-commutative algebra, algebraic geometry, and singularity theory
- Symmetrca - representation theory
- SciPy - optimization, numerical integration, Fourier transforms, signal processing, ODE solvers

```
sage: python_data
```

```
[1.2, 3.6000000000000001, 9.8000000000000007, 5.4000000000000004,  
7.5999999999999996]
```

```
sage: sigma = 2
```

```
sage: T = RealDistribution('gaussian', sigma)
```

```
sage: T.get_random_element()
```

[evaluate](#)

```
sage: T.cum_distribution_function(1)
```

```
sage: T.cum_distribution_function_inv(.95)
```

3.1 Other Packages

- Singular - commutative and non-commutative algebra, algebraic geometry, and singularity theory
- Symmetrca - representation theory
- Scry - optimization, numerical integration, Fourier transforms, signal processing, ODE solvers

```
sage: python_data
```

```
[1.2, 3.6000000000000001, 9.8000000000000007, 5.4000000000000004,  
7.5999999999999996]
```

```
sage: sigma = 2
```

```
sage: T = RealDistribution('gaussian', sigma)
```

```
sage: T.get_random_element()
```

```
-0.631116341092
```

```
sage: T.cum_distribution_function(1)
```

[evaluate](#)

```
sage: T.cum_distribution_function_inv(.95)
```

3.1 Other Packages

- Singular - commutative and non-commutative algebra, algebraic geometry, and singularity theory
- Symmetrca - representation theory
- SciPy - optimization, numerical integration, Fourier transforms, signal processing, ODE solvers


```
sage: python_data
```

```
[1.2, 3.6000000000000001, 9.8000000000000007, 5.4000000000000004,  
7.5999999999999996]
```

```
sage: sigma = 2
```

```
sage: T = RealDistribution('gaussian', sigma)
```

```
sage: T.get_random_element()
```

```
-0.631116341092
```

```
sage: T.cum_distribution_function(1)
```

```
0.691462461274
```

```
sage: T.cum_distribution_function_inv(.95)
```

[evaluate](#)

3.1 Other Packages

- Singular - commutative and non-commutative algebra, algebraic geometry, and singularity theory
- Symmetrca - representation theory
- SciPy - optimization, numerical integration, Fourier transforms, signal processing, ODE solvers

```
sage: T.cum_distribution_function_inv(.95)
```

3.2897072539

3.1 Other Packages

- Singular - commutative and non-commutative algebra, algebraic geometry, and singularity theory
- Symmetrca - representation theory
- SciPy - optimization, numerical integration, Fourier transforms, signal processing, ODE solvers
- 80 others...number theory, finite fields, number fields, cryptography

3.2 The Glue

- Python is the development language
- Python is the user language
- There is no such thing as a kernel
- Any Python package may be simply import'ed

```
import scipy
scipy.version.version
```

```
sage: T.get_random_element()
```

```
-0.631116341092
```

```
sage: T.cum_distribution_function(1)
```

```
0.691462461274
```

```
sage: T.cum_distribution_function_inv(.95)
```

```
3.2897072539
```

3.1 Other Packages

- Singular - commutative and non-commutative algebra, algebraic geometry, and singularity theory
- Symmetriza - representation theory
- SciPy - optimization, numerical integration, Fourier transforms, signal processing, ODE solvers
- 80 others...number theory, finite fields, number fields, cryptography

3.2 The Glue

- Python is the development language
- Python is the user language

3.1 Other Packages

- Singular - commutative and non-commutative algebra, algebraic geometry, and singularity theory
- Symmetriza - representation theory
- SciPy - optimization, numerical integration, Fourier transforms, signal processing, ODE solvers
- 80 others...number theory, finite fields, number fields, cryptography

3.2 The Glue

- Python is the development language
- Python is the user language
- There is no such thing as a kernel
- Any Python package may be simply import'ed

```
import scipy
scipy.version.version
```

[evaluate](#)

- 80 others...number theory, finite fields, number fields, cryptography

3.2 The Glue

- Python is the development language
- Python is the user language
- There is no such thing as a kernel
- Any Python package may be simply `import`'ed

```
import scipy
scipy.version.version
```

[evaluate](#)

```
scipy.finfo('float128').tiny
```

4 History and Development

- Started in 2005 by William Stein (U of Washington), as reaction to proprietary systems
- 95 packages; 500,000 lines of new code

3.2 The Glue

- Python is the development language
- Python is the user language
- There is no such thing as a kernel
- Any Python package may be simply `import`'ed

```
import scipy  
scipy.version.version
```

[evaluate](#)

```
scipy.finfo('float128').tiny
```

4 History and Development

- Started in 2005 by William Stein (U of Washington), as reaction to proprietary systems
- 95 packages; 500,000 lines of new code
- Funding: US National Science Foundation, Google, Microsoft, US Department of Defense, ANR, CNRS, academic institutions, and others

3.2 The Glue

- Python is the development language
- Python is the user language
- There is no such thing as a kernel
- Any Python package may be simply `import`'ed

```
import scipy  
scipy.version.version
```

[evaluate](#)

```
scipy.finfo('float128').tiny
```

4 History and Development

- Started in 2005 by William Stein (U of Washington), as reaction to proprietary systems
- 95 packages; 500,000 lines of new code
- Funding: US National Science Foundation, Google, Microsoft, US Department of Defense, ANR, CNRS, academic institutions, and others

3.2 The Glue

- Python is the development language
- Python is the user language
- There is no such thing as a kernel
- Any Python package may be simply `import`'ed

```
import scipy
scipy.__version__
```

[evaluate](#)

```
scipy.finfo('float128').tiny
```

4 History and Development

- Started in 2005 by William Stein (U of Washington), as reaction to proprietary systems
- 95 packages; 500,000 lines of new code
- Funding: US National Science Foundation, Google, Microsoft, US Department of Defense, ANR, CNRS, academic institutions, and others

3.2 The Glue

- Python is the development language
- Python is the user language
- There is no such thing as a kernel
- Any Python package may be simply `import`'ed

```
import scipy
scipy.version.version
```

[evaluate](#)

```
scipy.finfo('float128').tiny
```

4 History and Development

- Started in 2005 by William Stein (U of Washington), as reaction to proprietary systems
- 95 packages; 500,000 lines of new code
- Funding: US National Science Foundation, Google, Microsoft, US Department of Defense, ANR, CNRS, academic institutions, and others

3.2 The Glue

- Python is the development language
- Python is the user language
- There is no such thing as a kernel
- Any Python package may be simply `import`'ed

```
import scipy
scipy.version.version
evaluate
```

```
scipy.finfo('float128').tiny
```

4 History and Development

- Started in 2005 by William Stein (U of Washington), as reaction to proprietary systems
- 95 packages; 500,000 lines of new code
- Funding: US National Science Foundation, Google, Microsoft, US Department of Defense, ANR, CNRS, academic institutions, and others

3.2 The Glue

- Python is the development language
- Python is the user language
- There is no such thing as a kernel
- Any Python package may be simply import'ed

```
import scipy
scipy.version.version
```

```
'0.8.0'
```

```
scipy.finfo('float128').tiny
```

[evaluate](#)

4 History and Development

- Started in 2005 by William Stein (U of Washington), as reaction to proprietary systems
- 95 packages; 500,000 lines of new code

• Funding: US National Science Foundation, Google, Microsoft, US Department of Defense, ANR, CNRS, academic institutions, and others

3.2 The Glue

- Python is the development language
- Python is the user language
- There is no such thing as a kernel
- Any Python package may be simply `import`'ed

```
import scipy
scipy.version.version
'0.8.0'
```

```
scipy.finfo('float128').tiny
```

[evaluate](#)

4 History and Development

- Started in 2005 by William Stein (U of Washington), as reaction to proprietary systems
- 95 packages; 500,000 lines of new code

• Funding: US National Science Foundation, Google, Microsoft, US Department of Defense, ANR, CNRS, academic institutions, and others

3.2 The Glue

- Python is the development language
- Python is the user language
- There is no such thing as a kernel
- Any Python package may be simply `import`'ed

```
import scipy  
scipy.version.version
```

```
'0.8.0'
```

```
scipy.finfo('float128').tiny
```

[evaluate](#)

4 History and Development

- Started in 2005 by William Stein (U of Washington), as reaction to proprietary systems
- 95 packages; 500,000 lines of new code

• Funding: US National Science Foundation, Google, Microsoft, US Department of Defense, ANR, CNRS, academic institutions, and others

3.2 The Glue

- Python is the development language
- Python is the user language
- There is no such thing as a kernel
- Any Python package may be simply `import`'ed

```
import scipy
scipy.version.version
'0.8.0'
```

```
scipy.finfo('float128').tiny
```

[evaluate](#)

4 History and Development

- Started in 2005 by William Stein (U of Washington), as reaction to proprietary systems
- 95 packages; 500,000 lines of new code

• Funding: US National Science Foundation, Google, Microsoft, US Department of Defense, ANR, CNRS, academic institutions, and others

- Python is the development language
- Python is the user language
- There is no such thing as a kernel
- Any Python package may be simply import'ed

```
import scipy
scipy.version.version
```

```
'0.8.0'
```

```
scipy.finfo('float128').tiny
```

```
3.3621031431120935063e-4932
```

4 History and Development

- Started in 2005 by William Stein (U of Washington), as reaction to proprietary systems
- 95 packages; 500,000 lines of new code
- Funding: US National Science Foundation, Google, Microsoft, US Department of Defense, ANR, CNRS, academic institutions, and others
- 273 developers, about 50 contribute to any one release

4 History and Development

- Started in 2005 by William Stein (U of Washington), as reaction to proprietary systems
- 95 packages; 500,000 lines of new code
- Funding: US National Science Foundation, Google, Microsoft, US Department of Defense, ANR, CNRS, academic institutions, and others
- 273 developers, about 50 contribute to any one release
- Regular releases, very stable intermediate releases
- Every change is peer-reviewed, a single "release manager"
- Linux, OSX; Windows port in-progress
- Trivial to build from source, minimal prerequisites
- sage-devel, sage-support Google groups: 3000 members, 1500 messages/month
- 31 Sage Days Workshops, 3 Sage Educational Days

5 Sage Notebook, a Digital Blackboard

- Web interface, local or remote server

4 History and Development

- Started in 2005 by William Stein (U of Washington), as reaction to proprietary systems
- 95 packages; 500,000 lines of new code
- Funding: US National Science Foundation, Google, Microsoft, US Department of Defense, ANR, CNRS, academic institutions, and others
- 273 developers, about 50 contribute to any one release
- Regular releases, very stable intermediate releases
- Every change is peer-reviewed, a single "release manager"
- Linux, OSX; Windows port in-progress
- Trivial to build from source, minimal prerequisites
- sage-devel, sage-support Google groups: 3000 members, 1500 messages/month
- 31 Sage Days Workshops, 3 Sage Educational Days

5 Sage Notebook, a Digital Blackboard

- Web interface, local or remote server

- Started in 2005 by William Stein (U of Washington), as reaction to proprietary systems
- 95 packages; 500,000 lines of new code
- Funding: US National Science Foundation, Google, Microsoft, US Department of Defense, ANR, CNRS, academic institutions, and others
- 273 developers, about 50 contribute to any one release
- Regular releases, very stable intermediate releases
- Every change is peer-reviewed, a single "release manager"
- Linux, OSX; Windows port in-progress
- Trivial to build from source, minimal prerequisites
- sage-devel, sage-support Google groups: 3000 members, 1500 messages/month
- 31 Sage Days Workshops, 3 Sage Educational Days

5 Sage Notebook, a Digital Blackboard

- Web interface, local or remote server
- Every copy of Sage contains a server
- Worksheet management

- Funding: US National Science Foundation, Google, Microsoft, US Department of Defense, ANR, CNRS, academic institutions, and others
- 273 developers, about 50 contribute to any one release
- Regular releases, very stable intermediate releases
- Every change is peer-reviewed, a single "release manager"
- Linux, OSX; Windows port in-progress
- Trivial to build from source, minimal prerequisites
- sage-devel, sage-support Google groups: 3000 members, 1500 messages/month
- 31 Sage Days Workshops, 3 Sage Educational Days

5 Sage Notebook, a Digital Blackboard

- Web interface, local or remote server
- Every copy of Sage contains a server
- Worksheet management
- Online tab completion, help, source

- 273 developers, about 50 contribute to any one release
- Regular releases, very stable intermediate releases
- Every change is peer-reviewed, a single "release manager"
- Linux, OSX; Windows port in-progress
- Trivial to build from source, minimal prerequisites
- sage-devel, sage-support Google groups: 3000 members, 1500 messages/month
- 31 Sage Days Workshops, 3 Sage Educational Days

5 Sage Notebook, a Digital Blackboard

- Web interface, local or remote server
- Every copy of Sage contains a server
- Worksheet management
- Online tab-completion, help, source

```
A = matrix(QQ, [[ 3, 2, 0, 1],  
                [ 2, 1, 1, 0],  
                [ 1, 1, 5, 2]
```

- Regular releases, very stable intermediate releases
- Every change is peer-reviewed, a single "release manager"
- Linux, OSX; Windows port in-progress
- Trivial to build from source, minimal prerequisites
- sage-devel, sage-support Google groups: 3000 members, 1500 messages/month
- 31 Sage Days Workshops, 3 Sage Educational Days

5 Sage Notebook, a Digital Blackboard

- Web interface, local or remote server
- Every copy of Sage contains a server
- Worksheet management
- Online tab-completion, help, source

```
A = matrix(QQ,[[ 3,  2,  0,  1],  
               [ 2,  1,  1,  0],  
               [ 1, -1,  5, -3],  
               [-2, -3,  5, -4]])
```


- Every change is peer-reviewed, a single "release manager"
- Linux, OSX; Windows port in-progress
- Trivial to build from source, minimal prerequisites
- sage-devel, sage-support Google groups: 3000 members, 1500 messages/month
- 31 Sage Days Workshops, 3 Sage Educational Days

5 Sage Notebook, a Digital Blackboard

- Web interface, local or remote server
- Every copy of Sage contains a server
- Worksheet management
- Online tab-completion, help, source

```
A = matrix(QQ, [[ 3,  2,  0,  1],  
                [ 2,  1,  1,  0],  
                [ 1, -1,  5, -3],  
                [-2, -3,  5, -4]])
```

[evaluate](#)

- Trivial to build from source, minimal prerequisites
- sage-devel, sage-support Google groups: 3000 members, 1500 messages/month
- 31 Sage Days Workshops, 3 Sage Educational Days

5 Sage Notebook, a Digital Blackboard

- Web interface, local or remote server
- Every copy of Sage contains a server
- Worksheet management
- Online tab-completion, help, source

```
A = matrix(QQ,[[ 3,  2, 0,  1],  
               [ 2,  1, 1,  0],  
               [ 1, -1, 5, -3],  
               [-2, -3, 5, -4]])
```

[evaluate](#)

- Tab-completion for methods

5 Sage Notebook, a Digital Blackboard

- Web interface, local or remote server
- Every copy of Sage contains a server
- Worksheet management
- Online tab-completion, help, source

```
A = matrix(QQ, [[ 3,  2,  0,  1],  
                [ 2,  1,  1,  0],  
                [ 1, -1,  5, -3],  
                [-2, -3,  5, -4]])
```

[evaluate](#)

- Tab-completion for methods

A.

- One question mark brings help, with *tested* examples

[New Worksheet](#) [Upload](#) [Download All Active](#)

Current Folder: [Active](#) [Archived](#) [Trash](#)

Active Worksheets	Owner / Collaborators	Last Edited
<input type="checkbox"/> Untitled	fcla Share now	46 days ago by fcla
<input type="checkbox"/> Graph theory in Sage	fcla Share now	273 days ago by fcla
<input type="checkbox"/> 3 Public-Key Cryptography	fcla Share now	292 days ago by admin
<input type="checkbox"/> 6 Cosets and Lagrange's Theorem	fcla Share now	294 days ago by admin
<input type="checkbox"/> billy	fcla Share now	297 days ago by admin
<input type="checkbox"/> Approximating Polynomials	fcla Share now	367 days ago by fcla
<input type="checkbox"/> Second Draft of Tutorial for PREP	kcrisman / fcla, jason3 Share now	367 days ago by jason3
<input type="checkbox"/> Untitled	fcla Share now	372 days ago by fcla
<input type="checkbox"/> Draft of PREP 2010 Tutorial	kcrisman / fcla, jason3 Share now	377 days ago by kcrisman
<input type="checkbox"/> Worksheet for Loci Article	kcrisman / byungchul.cha, fcla, jason3 Share now	420 days ago by byungchul.cha
<input type="checkbox"/> Approximating Polynomials	fcla Share now	443 days ago by fcla
<input type="checkbox"/> Sage-Enhanced Textbook	fcla Share now	489 days ago by admin
<input type="checkbox"/> Untitled	fcla Share now	489 days ago by fcla

[New Worksheet](#) [Upload](#) [Download All Active](#)

Current Folder: [Active](#) [Archived](#) [Trash](#)

Active Worksheets	Owner / Collaborators	Last Edited
<input type="checkbox"/> Untitled	fcla Share now	46 days ago by fcla
<input type="checkbox"/> Graph theory in Sage	fcla Share now	273 days ago by fcla
<input type="checkbox"/> 3 Public-Key Cryptography	fcla Share now	292 days ago by admin
<input type="checkbox"/> 6 Cosets and Lagrange's Theorem	fcla Share now	294 days ago by admin
<input type="checkbox"/> billy	fcla Share now	297 days ago by admin
<input type="checkbox"/> Approximating Polynomials	fcla Share now	367 days ago by fcla
<input type="checkbox"/> Second Draft of Tutorial for PREP	kcrisman / fcla, jason3 Share now	367 days ago by jason3
<input type="checkbox"/> Untitled	fcla Share now	372 days ago by fcla
<input type="checkbox"/> Draft of PREP 2010 Tutorial	kcrisman / fcla, jason3 Share now	377 days ago by kcrisman
<input type="checkbox"/> Worksheet for Loci Article	kcrisman / byungchul.cha, fcla, jason3 Share now	420 days ago by byungchul.cha
<input type="checkbox"/> Approximating Polynomials	fcla Share now	443 days ago by fcla
<input type="checkbox"/> Sage-Enhanced Textbook	fcla Share now	489 days ago by admin
<input type="checkbox"/> Untitled	fcla Share now	489 days ago by fcla

5 Sage Notebook, a Digital Blackboard

- Web interface, local or remote server
- Every copy of Sage contains a server
- Worksheet management
- Online tab-completion, help, source

```
A = matrix(QQ,[[ 3,  2,  0,  1],  
               [ 2,  1,  1,  0],  
               [ 1, -1,  5, -3],  
               [-2, -3,  5, -4]])
```

[evaluate](#)

- Tab-completion for methods

A.

- One question mark brings help, with *tested* examples

- Web interface, local or remote server
- Every copy of Sage contains a server
- Worksheet management
- Online tab-completion, help, source

```
A = matrix(QQ, [[ 3,  2,  0,  1],  
                [ 2,  1,  1,  0],  
                [ 1, -1,  5, -3],  
                [-2, -3,  5, -4]])
```

[evaluate](#)

- Tab-completion for methods

A.

- One question mark brings help, with *tested* examples

A. jordan_form?

- Web interface, local or remote server
- Every copy of Sage contains a server
- Worksheet management
- Online tab-completion, help, source

```
A = matrix(QQ, [[ 3,  2,  0,  1],
                [ 2,  1,  1,  0],
                [ 1, -1,  5, -3],
                [-2, -3,  5, -4]])
```

- Tab-completion for methods

A.

[evaluate](#)

- One question mark brings help, with *tested* examples

A.jordan form?


```
A = matrix(QQ, [[ 3, 2, 0, 1],
                 [ 2, 1, 1, 0],
                 [ 1, -1, 5, -3],
                 [-2, -3, 5, -4]])
```

- Tab-completion for methods

A.

[evaluate](#)

- One question mark brings help, with *tested* examples

A. jordan_form?

- Two question marks brings source code

A. jordan_form??

- Source code display

`[-2, -3, 5, -4]])`

- Tab-completion for methods

A.

[evaluate](#)

- One question mark brings help, with *tested* examples

A. jordan_form?

- Two question marks brings source code

A. jordan_form??

- Source code display

`search_def('jordan')`

`[-2, -3, 5, -4]])`

Tab-completion for methods

A.

evaluate

A.N	A.hessenbergize	A.plot
A.abs	A.image	A.prod_of_row_sums
A.act_on_polynomial	A.integer_kernel	A.randomize
A.add_multiple_of_column	A.inverse	A.rank
A.add_multiple_of_row	A.invert	A.rename
A.additive_order	A.is_bistochastic	A.rescale_col
A.adjoint	A.is_dense	A.rescale_row
A.antitranspose	A.is_idempotent	A.reset_name
A.apply_map	A.is_immutable	A.restrict
A.apply_morphism	A.is_invertible	A.restrict_codomain
A.as_sum_of_permutations	A.is_mutable	A.restrict_domain
A.augment	A.is_nilpotent	A.right_eigenmatrix
A.base_extend	A.is_one	A.right_eigenspaces
A.base_ring	A.is_scalar	A.right_eigenvectors
A.block_sum	A.is_singular	A.right_kernel
A.block_schurian_product	A.is_skew_symmetric	A.right_nullity
A.category	A.is_sparse	A.rook_vector

A.N	A.hessenbergize	A.plot
A.abs	A.image	A.prod_of_row_sums
A.act_on_polynomial	A.integer_kernel	A.randomize
A.add_multiple_of_column	A.inverse	A.rank
A.add_multiple_of_row	A.invert	A.rename
A.additive_order	A.is_bistochastic	A.rescale_col
A.adjoint	A.is_dense	A.rescale_row
A.antitranspose	A.is_idempotent	A.reset_name
A.apply_map	A.is_immutable	A.restrict
A.apply_morphism	A.is_invertible	A.restrict_codomain
A.as_sum_of_permutations	A.is_mutable	A.restrict_domain
A.augment	A.is_nilpotent	A.right_eigenmatrix
A.base_extend	A.is_one	A.right_eigenspaces
A.base_ring	A.is_scalar	A.right_eigenvectors
A.block_sum	A.is_singular	A.right_kernel
A.cartesian_product	A.is_skew_symmetric	A.right_nullity
A.category	A.is_sparse	A.rook_vector
A.change_ring	A.is_square	A.row
A.characteristic_polynomial	A.is_symmetric	A.row_module
A.charpoly	A.is_unit	A.row_space
A.cholesky_decomposition	A.is_zero	A.rows
A.column	A.iterates	A.rref
A.column_module	A.jordan_form	A.save
A.column_space	A.kernel	A.set_block
A.columns	A.kernel on	A.set_col to multiple of col

A.N	A.hessenbergize	A.plot
A.abs	A.image	A.prod_of_row_sums
A.act_on_polynomial	A.integer_kernel	A.randomize
A.add_multiple_of_column	A.inverse	A.rank
A.add_multiple_of_row	A.invert	A.rename
A.additive_order	A.is_bistochastic	A.rescale_col
A.adjoint	A.is_dense	A.rescale_row
A.antitranspose	A.is_idempotent	A.reset_name
A.apply_map	A.is_immutable	A.restrict
A.apply_morphism	A.is_invertible	A.restrict_codomain
A.as_sum_of_permutations	A.is_mutable	A.restrict_domain
A.augment	A.is_nilpotent	A.right_eigenmatrix
A.base_extend	A.is_one	A.right_eigenspaces
A.base_ring	A.is_scalar	A.right_eigenvectors
A.block_sum	A.is_singular	A.right_kernel
A.cartesian_product	A.is_skew_symmetric	A.right_nullity
A.category	A.is_sparse	A.rook_vector
A.change_ring	A.is_square	A.row
A.characteristic_polynomial	A.is_symmetric	A.row_module
A.charpoly	A.is_unit	A.row_space
A.cholesky_decomposition	A.is_zero	A.rows
A.column	A.iterates	A.rref
A.column_module	A.jordan_form	A.save
A.column_space	A.kernel	A.set_block
A.columns	A.kernel on	A.set_col to multiple of col

evaluate

A.N	A.hessenbergize	A.plot
A.abs	A.image	A.prod_of_row_sums
A.act_on_polynomial	A.integer_kernel	A.randomize
A.add_multiple_of_column	A.inverse	A.rank
A.add_multiple_of_row	A.invert	A.rename
A.additive_order	A.is_bistochastic	A.rescale_col
A.adjoint	A.is_dense	A.rescale_row
A.antitranspose	A.is_idempotent	A.reset_name
A.apply_map	A.is_immutable	A.restrict
A.apply_morphism	A.is_invertible	A.restrict_codomain
A.as_sum_of_permutations	A.is_mutable	A.restrict_domain
A.augment	A.is_nilpotent	A.right_eigenmatrix
A.base_extend	A.is_one	A.right_eigenspaces
A.base_ring	A.is_scalar	A.right_eigenvectors
A.block_sum	A.is_singular	A.right_kernel
A.cartesian_product	A.is_skew_symmetric	A.right_nullity
A.category	A.is_sparse	A.rook_vector
A.change_ring	A.is_square	A.row
A.characteristic_polynomial	A.is_symmetric	A.row_module
A.charpoly	A.is_unit	A.row_space
A.cholesky_decomposition	A.is_zero	A.rows
A.column	A.iterates	A.rref
A.column_module	A.jordan_form	A.save
A.column_space	A.kernel	A.set_block
A.columns	A.kernel on	A.set_col to multiple of col

evaluate		
A.N	A.hessenbergize	A.plot
A.abs	A.image	A.prod_of_row_sums
A.act_on_polynomial	A.integer_kernel	A.randomize
A.add_multiple_of_column	A.inverse	A.rank
A.add_multiple_of_row	A.invert	A.rename
A.additive_order	A.is_bistochastic	A.rescale_col
A.adjoint	A.is_dense	A.rescale_row
A.antitranspose	A.is_idempotent	A.reset_name
A.apply_map	A.is_immutable	A.restrict
A.apply_morphism	A.is_invertible	A.restrict_codomain
A.as_sum_of_permutations	A.is_mutable	A.restrict_domain
A.augment	A.is_nilpotent	A.right_eigenmatrix
A.base_extend	A.is_one	A.right_eigenspaces
A.base_ring	A.is_scalar	A.right_eigenvectors
A.block_sum	A.is_singular	A.right_kernel
A.cartesian_product	A.is_skew_symmetric	A.right_nullity
A.category	A.is_sparse	A.rook_vector
A.change_ring	A.is_square	A.row
A.characteristic_polynomial	A.is_symmetric	A.row_module
A.charpoly	A.is_unit	A.row_space
A.cholesky_decomposition	A.is_zero	A.rows
A.column	A.iterates	A.rref
A.column_module	A.jordan_form	A.save
A.column_space	A.kernel	A.set_block
A.columns	A.kernel on	A.set_col to multiple of col

A.N	A.hessenbergize	A.plot
A.abs	A.image	A.prod_of_row_sums
A.act_on_polynomial	A.integer_kernel	A.randomize
A.add_multiple_of_column	A.inverse	A.rank
A.add_multiple_of_row	A.invert	A.rename
A.additive_order	A.is_bistochastic	A.rescale_col
A.adjoint	A.is_dense	A.rescale_row
A.antitranspose	A.is_idempotent	A.reset_name
A.apply_map	A.is_immutable	A.restrict
A.apply_morphism	A.is_invertible	A.restrict_codomain
A.as_sum_of_permutations	A.is_mutable	A.restrict_domain
A.augment	A.is_nilpotent	A.right_eigenmatrix
A.base_extend	A.is_one	A.right_eigenspaces
A.base_ring	A.is_scalar	A.right_eigenvectors
A.block_sum	A.is_singular	A.right_kernel
A.cartesian_product	A.is_skew_symmetric	A.right_nullity
A.category	A.is_sparse	A.rook_vector
A.change_ring	A.is_square	A.row
A.characteristic_polynomial	A.is_symmetric	A.row_module
A.charpoly	A.is_unit	A.row_space
A.cholesky_decomposition	A.is_zero	A.rows
A.column	A.iterates	A.rref
A.column_module	A.jordan_form	A.save
A.column_space	A.kernel	A.set_block
A.columns	A.kernel on	A.set_col to multiple of col

A.N	A.hessenbergize	A.plot
A.abs	A.image	A.prod_of_row_sums
A.act_on_polynomial	A.integer_kernel	A.randomize
A.add_multiple_of_column	A.inverse	A.rank
A.add_multiple_of_row	A.invert	A.rename
A.additive_order	A.is_bistochastic	A.rescale_col
A.adjoint	A.is_dense	A.rescale_row
A.antitranspose	A.is_idempotent	A.reset_name
A.apply_map	A.is_immutable	A.restrict
A.apply_morphism	A.is_invertible	A.restrict_codomain
A.as_sum_of_permutations	A.is_mutable	A.restrict_domain
A.augment	A.is_nilpotent	A.right_eigenmatrix
A.base_extend	A.is_one	A.right_eigenspaces
A.base_ring	A.is_scalar	A.right_eigenvectors
A.block_sum	A.is_singular	A.right_kernel
A.cartesian_product	A.is_skew_symmetric	A.right_nullity
A.category	A.is_sparse	A.rook_vector
A.change_ring	A.is_square	A.row
A.characteristic_polynomial	A.is_symmetric	A.row_module
A.charpoly	A.is_unit	A.row_space
A.cholesky_decomposition	A.is_zero	A.rows
A.column	A.iterates	A.rref
A.column_module	A.jordan_form	A.save
A.column_space	A.kernel	A.set_block
A.columns	A.kernel on	A.set_col to multiple of col

A.N	A.hessenbergize	A.plot
A.abs	A.image	A.prod_of_row_sums
A.act_on_polynomial	A.integer_kernel	A.randomize
A.add_multiple_of_column	A.inverse	A.rank
A.add_multiple_of_row	A.invert	A.rename
A.additive_order	A.is_bistochastic	A.rescale_col
A.adjoint	A.is_dense	A.rescale_row
A.antitranspose	A.is_idempotent	A.reset_name
A.apply_map	A.is_immutable	A.restrict
A.apply_morphism	A.is_invertible	A.restrict_codomain
A.as_sum_of_permutations	A.is_mutable	A.restrict_domain
A.augment	A.is_nilpotent	A.right_eigenmatrix
A.base_extend	A.is_one	A.right_eigenspaces
A.base_ring	A.is_scalar	A.right_eigenvectors
A.block_sum	A.is_singular	A.right_kernel
A.cartesian_product	A.is_skew_symmetric	A.right_nullity
A.category	A.is_sparse	A.rook_vector
A.change_ring	A.is_square	A.row
A.characteristic_polynomial	A.is_symmetric	A.row_module
A.charpoly	A.is_unit	A.row_space
A.cholesky_decomposition	A.is_zero	A.rows
A.column	A.iterates	A.rref
A.column_module	A.jordan_form	A.save
A.column_space	A.kernel	A.set_block
A.columns	A.kernel on	A.set_col to multiple of col

evaluate

A.N	A.hessenbergize	A.plot
A.abs	A.image	A.prod_of_row_sums
A.act_on_polynomial	A.integer_kernel	A.randomize
A.add_multiple_of_column	A.inverse	A.rank
A.add_multiple_of_row	A.invert	A.rename
A.additive_order	A.is_bistochastic	A.rescale_col
A.adjoint	A.is_dense	A.rescale_row
A.antitranspose	A.is_idempotent	A.reset_name
A.apply_map	A.is_immutable	A.restrict
A.apply_morphism	A.is_invertible	A.restrict_codomain
A.as_sum_of_permutations	A.is_mutable	A.restrict_domain
A.augment	A.is_nilpotent	A.right_eigenmatrix
A.base_extend	A.is_one	A.right_eigenspaces
A.base_ring	A.is_scalar	A.right_eigenvectors
A.block_sum	A.is_singular	A.right_kernel
A.cartesian_product	A.is_skew_symmetric	A.right_nullity
A.category	A.is_sparse	A.rook_vector
A.change_ring	A.is_square	A.row
A.characteristic_polynomial	A.is_symmetric	A.row_module
A.charpoly	A.is_unit	A.row_space
A.cholesky_decomposition	A.is_zero	A.rows
A.column	A.iterates	A.rref
A.column_module	A.jordan_form	A.save
A.column_space	A.kernel	A.set_block
A.columns	A.kernel on	A.set_col to multiple of col

• Online tab-completion, help, source

```
A = matrix(QQ, [[ 3, 2, 0, 1],
                [ 2, 1, 1, 0],
                [ 1, -1, 5, -3],
                [-2, -3, 5, -4]])
```

• Tab-completion for methods

A.

[evaluate](#)

A.N	A.hessenbergize	A.plot
A.abs	A.image	A.prod_of_row_sums
A.act_on_polynomial	A.integer_kernel	A.randomize
A.add_multiple_of_column	A.inverse	A.rank
A.add_multiple_of_row	A.invert	A.rename
A.additive_order	A.is_bistochastic	A.rescale_col
A.adjoint	A.is_dense	A.rescale_row
A.antitranspose	A.is_idempotent	A.reset_name
A.apply_map	A.is_immutable	A.restrict
A.apply_morphism	A.is_invertible	A.restrict_codomain
A.as_sum_of_permutations	A.is_mutable	A.restrict_domain

- Online tab-completion, help, source

```
A = matrix(QQ, [[ 3,  2,  0,  1],  
                [ 2,  1,  1,  0],  
                [ 1, -1,  5, -3],  
                [-2, -3,  5, -4]])
```

- Tab-completion for methods

A.|

[evaluate](#)



- One question mark brings help, with *tested* examples

A.jordan_form?

- Two question marks brings source code

A.jordan_form??

A.

[evaluate](#)

- One question mark brings help, with *tested* examples

A. jordan form? ⓘ

- Two question marks brings source code

A. jordan form??

- Source code display

search_def(jordan)

- Low barriers to development

- Revision control log (Interrupt to stop serving)

A.

- One question mark brings help, with *tested* examples

A. jordan_form?

[evaluate](#)

- Two question marks brings source code

A. jordan_form??

- Source code display

`search_def('jordan')`

- Low barriers to development

- Revision control log (Interrupt to stop serving)

A.

- One question mark brings help, with *tested* examples

A. jordan_form?

evaluate
loading...

- Two question marks brings source code

A. jordan_form??

- Source code display

search_def('jordan')

- Low barriers to development

A.

• One question mark brings help, with *tested* examples

A.jordan_form?

[evaluate](#)

Click here to pop out
unprinted

File: /sage/abstract/devel/sage/sage/matrix/matrix2.pyx

Type: <type 'builtin_function_or_method'>

Definition: A.jordan_form(base_ring=None, sparse=False, subdivide=True, transformation=False)

Docstring:

Compute the Jordan normal form of this square matrix A , if it exists.

This computation is performed in a naive way using the ranks of powers of $A - xI$, where x is an eigenvalue of the matrix A . If desired, a transformation matrix P can be returned, which is such that the Jordan canonical form is given by $P^{-1}AP$.

INPUT:

- `base_ring` - Ring in which to compute the Jordan form.
- `sparse` - (default False) If `sparse=True`, return a sparse matrix.
- `subdivide` - (default True) If `subdivide=True`, the subdivisions for the Jordan blocks in the matrix are shown.

Definition: `A.jordan_form(base_ring=None, sparse=False, subdivide=True, transformation=False)`

Docstring:

Compute the Jordan normal form of this square matrix A , if it exists.

This computation is performed in a naive way using the ranks of powers of $A - xI$, where x is an eigenvalue of the matrix A . If desired, a transformation matrix P can be returned, which is such that the Jordan canonical form is given by $P^{-1}AP$.

INPUT:

- `base_ring` - Ring in which to compute the Jordan form.
- `sparse` - (default False) If `sparse=True`, return a sparse matrix.
- `subdivide` - (default True) If `subdivide=True`, the subdivisions for the Jordan blocks in the matrix are shown.
- `transformation` - (default False) If `transformation=True`, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either 0×0 or 1×1 .

In the case of exact rings, this method does not compute any generalized form of the Jordan normal form, but is only able to compute the result if the characteristic polynomial of the matrix splits over the specific base ring.

EXAMPLES:

```
sage: a = matrix(ZZ,4,[1, 0, 0, 0, 0, 1, 0, 0, 1, \
-1, 1, 0, 1, -1, 1, 2]); a
```


Docstring.

Compute the Jordan normal form of this square matrix A , if it exists.

This computation is performed in a naive way using the ranks of powers of $A - xI$, where x is an eigenvalue of the matrix A . If desired, a transformation matrix P can be returned, which is such that the Jordan canonical form is given by $P^{-1}AP$.

INPUT:

- `base_ring` - Ring in which to compute the Jordan form.
- `sparse` - (default False) If `sparse=True`, return a sparse matrix.
- `subdivide` - (default True) If `subdivide=True`, the subdivisions for the Jordan blocks in the matrix are shown.
- `transformation` - (default False) If `transformation=True`, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either 0×0 or 1×1 .

In the case of exact rings, this method does not compute any generalized form of the Jordan normal form, but is only able to compute the result if the characteristic polynomial of the matrix splits over the specific base ring.

EXAMPLES:

```
sage: a = matrix(ZZ,4,[1, 0, 0, 0, 0, 1, 0, 0, 1, \
-1, 1, 0, 1, -1, 1, 2]); a
[ 1  0  0  0]
[ 0  1  0  0]
[ 1 -1  1  0]
[ 1 -1  1  2]
a.jordan_form()
[2|0 0|0]
```

Docstring.

Compute the Jordan normal form of this square matrix A , if it exists.

This computation is performed in a naive way using the ranks of powers of $A - xI$, where x is an eigenvalue of the matrix A . If desired, a transformation matrix P can be returned, which is such that the Jordan canonical form is given by $P^{-1}AP$.

INPUT:

- `base_ring` - Ring in which to compute the Jordan form.
- `sparse` - (default `False`) If `sparse=True`, return a sparse matrix.
- `subdivide` - (default `True`) If `subdivide=True`, the subdivisions for the Jordan blocks in the matrix are shown.
- `transformation` - (default `False`) If `transformation=True`, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either 0×0 or 1×1 .

In the case of exact rings, this method does not compute any generalized form of the Jordan normal form, but is only able to compute the result if the characteristic polynomial of the matrix splits over the specific base ring.

EXAMPLES:

```
sage: a = matrix(ZZ,4,[1, 0, 0, 0, 0, 1, 0, 0, 1, \
-1, 1, 0, 1, -1, 1, 2]); a
[ 1  0  0  0]
[ 0  1  0  0]
[ 1 -1  1  0]
[ 1 -1  1  2]
a.jordan_form()
[2|0 0|0]
```


Docstring.

Compute the Jordan normal form of this square matrix A , if it exists.

This computation is performed in a naive way using the ranks of powers of $A - xI$, where x is an eigenvalue of the matrix A . If desired, a transformation matrix P can be returned, which is such that the Jordan canonical form is given by $P^{-1}AP$.

INPUT:

- `base_ring` - Ring in which to compute the Jordan form.
- `sparse` - (default `False`) If `sparse=True`, return a sparse matrix.
- `subdivide` - (default `True`) If `subdivide=True`, the subdivisions for the Jordan blocks in the matrix are shown.
- `transformation` - (default `False`) If `transformation=True`, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either 0×0 or 1×1 .

In the case of exact rings, this method does not compute any generalized form of the Jordan normal form, but is only able to compute the result if the characteristic polynomial of the matrix splits over the specific base ring.

EXAMPLES:

```
sage: a = matrix(ZZ, 4, [1, 0, 0, 0, 0, 1, 0, 0, 1, \
-1, 1, 0, 1, -1, 1, 2]); a
[ 1  0  0  0]
[ 0  1  0  0]
[ 1 -1  1  0]
[ 1 -1  1  2]
a.jordan_form()
[2|0 0|0]
```


Docstring:

Compute the Jordan normal form of this square matrix A , if it exists.

This computation is performed in a naive way using the ranks of powers of $A - xI$, where x is an eigenvalue of the matrix A . If desired, a transformation matrix P can be returned, which is such that the Jordan canonical form is given by $P^{-1}AP$.

INPUT:

- `base_ring` - Ring in which to compute the Jordan form.
- `sparse` - (default `False`) If `sparse=True`, return a sparse matrix.
- `subdivide` - (default `True`) If `subdivide=True`, the subdivisions for the Jordan blocks in the matrix are shown.
- `transformation` - (default `False`) If `transformation=True`, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either 0×0 or 1×1 .

In the case of exact rings, this method does not compute any generalized form of the Jordan normal form, but is only able to compute the result if the characteristic polynomial of the matrix splits over the specific base ring.

EXAMPLES:

```
sage: a = matrix(ZZ,4,[1, 0, 0, 0, 0, 1, 0, 0, 1, \
-1, 1, 0, 1, -1, 1, 2]); a
[ 1  0  0  0]
[ 0  1  0  0]
[ 1 -1  1  0]
[ 1 -1  1  2]
sage: a.jordan_form()
[2|0 0|0]
```

Docstring.

Compute the Jordan normal form of this square matrix A , if it exists.

This computation is performed in a naive way using the ranks of powers of $A - xI$, where x is an eigenvalue of the matrix A . If desired, a transformation matrix P can be returned, which is such that the Jordan canonical form is given by $P^{-1}AP$.

INPUT:

- `base_ring` - Ring in which to compute the Jordan form.
- `sparse` - (default False) If `sparse=True`, return a sparse matrix.
- `subdivide` - (default True) If `subdivide=True`, the subdivisions for the Jordan blocks in the matrix are shown.
- `transformation` - (default False) If `transformation=True`, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either 0×0 or 1×1 .

In the case of exact rings, this method does not compute any generalized form of the Jordan normal form, but is only able to compute the result if the characteristic polynomial of the matrix splits over the specific base ring.

EXAMPLES:

```
sage: a = matrix(ZZ,4,[1, 0, 0, 0, 0, 1, 0, 0, 1, \
-1, 1, 0, 1, -1, 1, 2]); a
[ 1  0  0  0]
[ 0  1  0  0]
[ 1 -1  1  0]
[ 1 -1  1  2]
a.jordan_form()
[2|0 0|0]
```


Docstring:

Compute the Jordan normal form of this square matrix A , if it exists.

This computation is performed in a naive way using the ranks of powers of $A - xI$, where x is an eigenvalue of the matrix A . If desired, a transformation matrix P can be returned, which is such that the Jordan canonical form is given by $P^{-1}AP$.

INPUT:

- `base_ring` - Ring in which to compute the Jordan form.
- `sparse` - (default `False`) If `sparse=True`, return a sparse matrix.
- `subdivide` - (default `True`) If `subdivide=True`, the subdivisions for the Jordan blocks in the matrix are shown.
- `transformation` - (default `False`) If `transformation=True`, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either 0×0 or 1×1 .

In the case of exact rings, this method does not compute any generalized form of the Jordan normal form, but is only able to compute the result if the characteristic polynomial of the matrix splits over the specific base ring.

EXAMPLES:

```
sage: a = matrix(ZZ,4,[1, 0, 0, 0, 0, 1, 0, 0, 1, \
-1, 1, 0, 1, -1, 1, 2]); a
[ 1  0  0  0]
[ 0  1  0  0]
[ 1 -1  1  0]
[ 1 -1  1  2]
a.jordan_form()
[2|0 0|0]
```


Docstring.

Compute the Jordan normal form of this square matrix A , if it exists.

This computation is performed in a naive way using the ranks of powers of $A - xI$, where x is an eigenvalue of the matrix A . If desired, a transformation matrix P can be returned, which is such that the Jordan canonical form is given by $P^{-1}AP$.

INPUT:

- `base_ring` - Ring in which to compute the Jordan form.
- `sparse` - (default False) If `sparse=True`, return a sparse matrix.
- `subdivide` - (default True) If `subdivide=True`, the subdivisions for the Jordan blocks in the matrix are shown.
- `transformation` - (default False) If `transformation=True`, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either 0×0 or 1×1 .

In the case of exact rings, this method does not compute any generalized form of the Jordan normal form, but is only able to compute the result if the characteristic polynomial of the matrix splits over the specific base ring.

EXAMPLES:

```
sage: a = matrix(ZZ,4,[1, 0, 0, 0, 0, 1, 0, 0, 1, \
-1, 1, 0, 1, -1, 1, 2]); a
[ 1  0  0  0]
[ 0  1  0  0]
[ 1 -1  1  0]
[ 1 -1  1  2]
a.jordan_form()
[2|0 0|0]
```

Docstring:

Compute the Jordan normal form of this square matrix A , if it exists.

This computation is performed in a naive way using the ranks of powers of $A - xI$, where x is an eigenvalue of the matrix A . If desired, a transformation matrix P can be returned, which is such that the Jordan canonical form is given by $P^{-1}AP$.

INPUT:

- `base_ring` - Ring in which to compute the Jordan form.
- `sparse` - (default False) If `sparse=True`, return a sparse matrix.
- `subdivide` - (default True) If `subdivide=True`, the subdivisions for the Jordan blocks in the matrix are shown.
- `transformation` - (default False) If `transformation=True`, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either 0×0 or 1×1 .

In the case of exact rings, this method does not compute any generalized form of the Jordan normal form, but is only able to compute the result if the characteristic polynomial of the matrix splits over the specific base ring.

EXAMPLES:

```
sage: a = matrix(ZZ,4,[1, 0, 0, 0, 0, 1, 0, 0, 1, \
-1, 1, 0, 1, -1, 1, 2]); a
[ 1  0  0  0]
[ 0  1  0  0]
[ 1 -1  1  0]
[ 1 -1  1  2]
a.jordan_form()
[2|0 0|0]
```


Docstring.

Compute the Jordan normal form of this square matrix A , if it exists.

This computation is performed in a naive way using the ranks of powers of $A - xI$, where x is an eigenvalue of the matrix A . If desired, a transformation matrix P can be returned, which is such that the Jordan canonical form is given by $P^{-1}AP$.

INPUT:

- `base_ring` - Ring in which to compute the Jordan form.
- `sparse` - (default `False`) If `sparse=True`, return a sparse matrix.
- `subdivide` - (default `True`) If `subdivide=True`, the subdivisions for the Jordan blocks in the matrix are shown.
- `transformation` - (default `False`) If `transformation=True`, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either 0×0 or 1×1 .

In the case of exact rings, this method does not compute any generalized form of the Jordan normal form, but is only able to compute the result if the characteristic polynomial of the matrix splits over the specific base ring.

EXAMPLES:

```
sage: a = matrix(ZZ,4,[1, 0, 0, 0, 0, 1, 0, 0, 1, \
-1, 1, 0, 1, -1, 1, 2]); a
[ 1  0  0  0]
[ 0  1  0  0]
[ 1 -1  1  0]
[ 1 -1  1  2]
a.jordan_form()
[2|0 0|0]
```


Docstring.

Compute the Jordan normal form of this square matrix A , if it exists.

This computation is performed in a naive way using the ranks of powers of $A - xI$, where x is an eigenvalue of the matrix A . If desired, a transformation matrix P can be returned, which is such that the Jordan canonical form is given by $P^{-1}AP$.

INPUT:

- `base_ring` - Ring in which to compute the Jordan form.
- `sparse` - (default `False`) If `sparse=True`, return a sparse matrix.
- `subdivide` - (default `True`) If `subdivide=True`, the subdivisions for the Jordan blocks in the matrix are shown.
- `transformation` - (default `False`) If `transformation=True`, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either 0×0 or 1×1 .

In the case of exact rings, this method does not compute any generalized form of the Jordan normal form, but is only able to compute the result if the characteristic polynomial of the matrix splits over the specific base ring.

EXAMPLES:

```
sage: a = matrix(ZZ,4,[1, 0, 0, 0, 0, 1, 0, 0, 1, \
-1, 1, 0, 1, -1, 1, 2]); a
[ 1  0  0  0]
[ 0  1  0  0]
[ 1 -1  1  0]
[ 1 -1  1  2]
a.jordan_form()
[2|0 0|0]
```

Docstring.

Compute the Jordan normal form of this square matrix A , if it exists.

This computation is performed in a naive way using the ranks of powers of $A - xI$, where x is an eigenvalue of the matrix A . If desired, a transformation matrix P can be returned, which is such that the Jordan canonical form is given by $P^{-1}AP$.

INPUT:

- `base_ring` - Ring in which to compute the Jordan form.
- `sparse` - (default False) If `sparse=True`, return a sparse matrix.
- `subdivide` - (default True) If `subdivide=True`, the subdivisions for the Jordan blocks in the matrix are shown.
- `transformation` - (default False) If `transformation=True`, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either 0×0 or 1×1 .

In the case of exact rings, this method does not compute any generalized form of the Jordan normal form, but is only able to compute the result if the characteristic polynomial of the matrix splits over the specific base ring.

EXAMPLES:

```
sage: a = matrix(ZZ,4,[1, 0, 0, 0, 0, 1, 0, 0, 1, \
-1, 1, 0, 1, -1, 1, 2]); a
[ 1  0  0  0]
[ 0  1  0  0]
[ 1 -1  1  0]
[ 1 -1  1  2]
a.jordan_form()
[2|0 0|0]
```


This computation is performed in a naive way using the ranks of powers of $A - xI$, where x is an eigenvalue of the matrix A . If desired, a transformation matrix P can be returned, which is such that the Jordan canonical form is given by $P^{-1}AP$.

INPUT:

- `base_ring` - Ring in which to compute the Jordan form.
- `sparse` - (default False) If `sparse=True`, return a sparse matrix.
- `subdivide` - (default True) If `subdivide=True`, the subdivisions for the Jordan blocks in the matrix are shown.
- `transformation` - (default False) If `transformation=True`, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either 0×0 or 1×1 .

In the case of exact rings, this method does not compute any generalized form of the Jordan normal form, but is only able to compute the result if the characteristic polynomial of the matrix splits over the specific base ring.

EXAMPLES:

```
sage: a = matrix(ZZ,4,[1, 0, 0, 0, 0, 1, 0, 0, 1, \
-1, 1, 0, 1, -1, 1, 2]); a
[ 1  0  0  0]
[ 0  1  0  0]
[ 1 -1  1  0]
[ 1 -1  1  2]
sage: a.jordan_form()
[2|0 0|0]
[+---+]
[0|1 1|0]
[0|1 1|0]
[+---+]
```


EXAMPLES:

```
sage: a = matrix(ZZ,4,[1, 0, 0, 0, 0, 1, 0, 0, 1, \
-1, 1, 0, 1, -1, 1, 2]); a
[ 1 0 0 0]
[ 0 1 0 0]
[ 1 -1 1 0]
[ 1 -1 1 2]
sage: a.jordan_form()
[2|0 0|0]
[-+----+]
[0|1 1|0]
[0|0 1|0]
[-+----+]
[0|0 0|1]
sage: a.jordan_form(subdivide=False)
[2 0 0 0]
[0 1 1 0]
[0 0 1 0]
[0 0 0 1]
sage: b = matrix(ZZ,3,range(9)); b
[0 1 2]
[3 4 5]
[6 7 8]
sage: b.jordan_form()
Traceback (most recent call last):
...
RuntimeError: Some eigenvalue does not exist in Integer Ring.
sage: b.jordan_form(RealField(15))
Traceback (most recent call last):
...
ValueError: Jordan normal form not implemented over inexact rings.
```

If you need the transformation matrix as well as the Jordan form of self, then pass the option transformation=True.

```
sage: M = Matrix(GF(2), [[1,0,1,0,0,0,1], [1,0,0,1,1,1,0], [1,1,0,1,1,1,1], [1,1,1,0,1,1,1], [1,1,1,0,0,1,0], [1,1,1,0,1,0,0], [1,1,1,1,1,1,1]]
sage: J, T = M.jordan_form(transformation=True)
sage: J
[1 1|0 0|0 0|0]
[0 1|0 0|0 0|0]
[---+---+---+]
[0 0|1 1|0 0|0]
[0 0|0 1|0 0|0]
[---+---+---+]
[0 0|0 0|1 1|0]
[0 0|0 0|0 1|0]
[---+---+---+]
[0 0|0 0|0 0|1]
sage: M * T == T * J
True
sage: T.rank()
7
sage: M.rank()
7
```

We verify that the bug from trac ticket #6932 is fixed:

```
sage: M=Matrix(1,1,[1])
sage: M.jordan_form(transformation=True)
([1], [1])
```

We now go through three 10×10 matrices to exhibit cases where there are multiple blocks of the same size:

sage: A = matrix(QQ, [[15, 37/3, -16, -104/3, -29, -7/3, 0, 2/3, -29/3, -1/3], [2, 9, -1, -6, -6, 0, 0, 0, 0, -2, 0], [24, 74/3, -41, -208/3, -58, -23/3, 0, 4/3, -58/3, -2/3], [-6, -19, 3, 21, 19, 0, 0, 0, 6, 0], [2, 6, 3, -6, -3, 1, 0, 0, -2, 0], [-96, -296/3, 176, 832/3, 232, 101/3, 0, -16/3, 232/3, 8/3]]

We now go through three 10×10 matrices to exhibit cases where there are multiple blocks of the same size:

```
sage: A = matrix(QQ, [[15, 37/3, -16, -104/3, -29, -7/3, 0, 2/3, -29/3, -1/3], [2, 9, -1, -6, -6, 0, 0, 0, -2, 0], [24, 74/3, -41, -208/3, -58, -23/3, 0, 4/3, -58/3, -2/3], [-6, -19, 3, 21, 19, 0, 0, 0, 6, 0], [2, 6, 3, -6, -3, 1, 0, 0, -2, 0], [-96, -296/3, 176, 832/3, 232, 101/3, 0, -16/3, 232/3, 8/3], [-4, -2/3, 21, 16/3, 4, 14/3, 3, -1/3, 4/3, -25/3], [20, 26/3, -66, -199/3, -42, -41/3, 0, 13/3, -55/3, -2/3], [18, 57, -9, -54, -57, 0, 0, 0, -15, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 3]]
```

```
sage: J, T = A.jordan_form(transformation=True); J
```

```
[3 1 0|0 0 0|0 0 0]
[0 3 1|0 0 0|0 0 0]
[0 0 3|0 0 0|0 0 0]
[-----+-----]
[0 0 0|3 1 0|0 0 0]
[0 0 0|0 3 1|0 0 0]
[0 0 0|0 0 3|0 0 0]
[-----+-----]
[0 0 0|0 0 0|3 1 0]
[0 0 0|0 0 0|0 3 1]
[0 0 0|0 0 0|0 0 3]
[-----+-----]
[0 0 0|0 0 0|0 0 3]
```

```
sage: T * J * T**(-1) == A
```

```
True
```

```
sage: T.rank()
```

```
10
```

```
sage: A = matrix(QQ, [[15, 37/3, -16, -14/3, -29, -7/3, 0, 2/3, 1/3, 44/3], [2, 9, -1, 0, -6, 0, 0, 0, 0, 3], [24, 74/3, -41, -28/3, -58, -23/3, 0, 4/3, 2/3, 88/3],
```

```
[15, 37/3, -16, -14/3, -29, -7/3, 0, 2/3, 1/3, 44/3]
[2, 9, -1, 0, -6, 0, 0, 0, 0, 3]
[24, 74/3, -41, -28/3, -58, -23/3, 0, 4/3, 2/3, 88/3]
```



```

[ 20 20/3 -00 -20/3 -42 -42/3 0 13/3 27/3 02/3]
[ 18 57 -9 0 -57 0 0 0 3 28]
[ 0 0 0 0 0 0 0 0 0 3]
sage: J, T = A.jordan_form(transformation=True); J
[3 1 0|0 0 0|0 0 0]
[0 3 1|0 0 0|0 0 0]
[0 0 3|0 0 0|0 0 0]
[-----+-----]
[0 0 0|3 1 0|0 0 0]
[0 0 0|0 3 1|0 0 0]
[0 0 0|0 0 3|0 0 0]
[-----+-----]
[0 0 0|0 0 0|3 1 0]
[0 0 0|0 0 0|0 3 0]
[-----+-----]
[0 0 0|0 0 0|0 0 3]
[0 0 0|0 0 0|0 0 3]
sage: T * J * T**(-1) == A
True
sage: T.rank()
10

sage: A = matrix(QQ, [[15, 37/3, -16, -104/3, -29, -7/3, 35, 2/3, -29/3, -1/3], [2, 9, -1, -6, -6, 0, 7, 0, -2, 0], [24, 74/3, -29, -208/3, -58, -14/3, 70, 4/3, -58/3, -2/3], [-6, -19, 3, 21, 19, 0, -21, 0, 6, 0], [2, 6, -1, -6, -3, 0, 7, 0, -2, 0], [-96, -296/3, 128, 832/3, 232, 65/3, -279, -16/3, 232/3, 8/3], [0, 0, 0, 0, 0, 0, 3, 0, 0, 0], [20, 26/3, -30, -199/3, -42, -14/3, 70, 13/3, -55/3, -2/3], [18, 57, -9, -54, -57, 0, 63, 0, -15, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 3]]
sage: J, T = A.jordan_form(transformation=True); J
[3 1 0|0 0 0|0 0 0]
[0 3 1|0 0 0|0 0 0]
[0 0 3|0 0 0|0 0 0]
[-----+-----]

```

Applications Places System Tue May 10, 7:03 AM rob

Adding Spice to Your Research with Sage -- Sage - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:8000/home/admin/2388/ Google

Home - Perimeter Institute ... Adding Spice to Your Resea... Active Worksheets -- Sage

```

[ 15 377/3 -10 -104/3 -23 -11/3 3 2/3 -23/3 -17/3]
[ 2 9 -1 -6 -6 0 7 0 -2 0]
[ 24 74/3 -29 -208/3 -58 -14/3 70 4/3 -58/3 -2/3]
[ -6 -19 3 21 19 0 -21 0 6 0]
[ 2 6 -1 -6 -3 0 7 0 -2 0]
[ -96 -296/3 128 832/3 232 65/3 -279 -16/3 232/3 8/3]
[ 0 0 0 0 0 0 3 0 0 0]
[ 20 26/3 -30 -199/3 -42 -14/3 70 13/3 -55/3 -2/3]
[ 18 57 -9 -54 -57 0 63 0 -15 0]
[ 0 0 0 0 0 0 0 0 0 3]

sage: J, T = A.jordan_form(transformation=True); J
[3 1 0|0 0|0 0|0]
[0 3 1|0 0|0 0|0]
[0 0 3|0 0|0 0|0]
[-----+-----]
[0 0 0|3 1|0 0|0]
[0 0 0|0 3|0 0|0]
[-----+-----]
[0 0 0|0 0|3 1|0]
[0 0 0|0 0|0 3|0]
[-----+-----]
[0 0 0|0 0|0 0|3 1]
[0 0 0|0 0|0 0|0 3]
[-----+-----]
[0 0 0|0 0|0 0|0 3]

sage: T * J * T**(-1) == A
True
sage: T.rank()
10

```

Two question marks brings source code

Pirsa: 11050014

Applications Places System Tue May 10, 7:04 AM rob

Adding Spice to Your Research with Sage -- Sage - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:8000/home/admin/2388/ Google

Home - Perimeter Institute ... Adding Spice to Your Resea... Active Worksheets -- Sage

```

[0 3 1|0 0|0 0|0 0|0]
[0 0 3|0 0|0 0|0 0|0]
[-----+-----]
[0 0 0|3 1|0 0|0 0|0]
[0 0 0|0 3|0 0|0 0|0]
[-----+-----]
[0 0 0|0 0|3 1|0 0|0]
[0 0 0|0 0|0 3|0 0|0]
[-----+-----]
[0 0 0|0 0|0 0|3 1|0]
[0 0 0|0 0|0 0|0 3|0]
[-----+-----]
[0 0 0|0 0|0 0|0 0|3]
sage: T * J * T**(-1) == A
True
sage: T.rank()
10

```

- Two question marks brings source code

A.jordan form??

- Source code display

search_def('jordan')

Applications Places System Tue May 10, 7:04 AM rob

Adding Spice to Your Research with Sage -- Sage - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:8000/home/admin/2388/ Google

Home - Perimeter Institute ... Adding Spice to Your Resea... Active Worksheets -- Sage

```

[0 0 0|0 0|0 3|0 0]
[-----+-----]
[0 0 0|0 0|0 3|1|0]
[0 0 0|0 0|0 0|3|0]
[-----+-----]
[0 0 0|0 0|0 0|0|3]
sage: T * J * T**(-1) == A
True
sage: T.rank()
10

```

- Two question marks brings source code

A. jordan_form??

- Source code display

search_def('jordan')

- Low barriers to development
- Revision control log (Interrupt to stop serving)

```
[0 0 0|0 0|0 0|3 1|0]
[0 0 0|0 0|0 0|3 0]
[-----+-----]
[0 0 0|0 0|0 0|0 3]
sage: T * J * T**(-1) == A
True
sage: T.rank()
10
```

- Two question marks brings source code

A.jordan_form??|

[evaluate](#)

- Source code display

search_def('jordan')

- Low barriers to development

- Revision control log (Interrupt to stop serving)

```
[0 0 0|0 0|0 0|3 1|0]
[0 0 0|0 0|0 0|3 0]
[-----+-----]
[0 0 0|0 0|0 0|0 3]
sage: T * J * T**(-1) == A
True
sage: T.rank()
10
```

Two question marks brings source code

A.jordan_form??|

evaluate



File: /sage/abstract/devel/sage/sage/matrix/matrix2.pyx

Click here to pop out
unprinted

Source Code (starting at line 5882):

```
def jordan_form(self, base_ring=None, sparse=False, subdivide=True, transformation=False):
    """
    Compute the Jordan normal form of this square matrix 'A', if it exists.

    This computation is performed in a naive way using the ranks of powers
    of 'A-xI', where 'x' is an eigenvalue of the matrix 'A'. If desired,
    a transformation matrix 'P' can be returned, which is such that the
    Jordan canonical form is given by 'P^(-1) A P'.
```


A.jordan_form??

[evaluate](#)

File: /sage/abstract/devel/sage/sage/matrix/matrix2.pyx

Click here to pop out
unprinted

Source Code (starting at line 5882):

```
def jordan_form(self, base_ring=None, sparse=False, subdivide=True, transformation=False):
```

Compute the Jordan normal form of this square matrix 'A', if it exists.

This computation is performed in a naive way using the ranks of powers of 'A-xI', where 'x' is an eigenvalue of the matrix 'A'. If desired, a transformation matrix 'P' can be returned, which is such that the Jordan canonical form is given by 'P⁻¹ A P'.

INPUT:

- ``base_ring`` - Ring in which to compute the Jordan form.
- ``sparse`` - (default ``False``) If ``sparse=True``, return a sparse matrix.
- ``subdivide`` - (default ``True``) If ``subdivide=True``, the subdivisions for the Jordan blocks in the matrix are shown.
- ``transformation`` - (default ``False``) If ``transformation=True``, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings

evaluate

File: /sage/abstract/level/sage/sage/matrix/matrix2.pyx

Click here to pop out
unprinted

Source Code (starting at line 5882):

```
def jordan_form(self, base_ring=None, sparse=False, subdivide=True, transformation=False):
    """
```

Compute the Jordan normal form of this square matrix 'A', if it exists.

This computation is performed in a naive way using the ranks of powers of 'A-xI', where 'x' is an eigenvalue of the matrix 'A'. If desired, a transformation matrix 'P' can be returned, which is such that the Jordan canonical form is given by 'P⁻¹ A P'.

INPUT:

- ``base_ring`` - Ring in which to compute the Jordan form.
- ``sparse`` - (default ``False``) If ``sparse=True``, return a sparse matrix.
- ``subdivide`` - (default ``True``) If ``subdivide=True``, the subdivisions for the Jordan blocks in the matrix are shown.
- ``transformation`` - (default ``False``) If ``transformation=True``, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either '0 \times 0' or '1 \times 1'.

In the case of exact rings, this method does not compute any

evaluate

File: /sage/abstract/devel/sage/sage/matrix/matrix2.pyx

Click here to pop out
unprinted

Source Code (starting at line 5882):

```
def jordan_form(self, base_ring=None, sparse=False, subdivide=True, transformation=False):
    """
```

Compute the Jordan normal form of this square matrix 'A', if it exists.

This computation is performed in a naive way using the ranks of powers of 'A-xI', where 'x' is an eigenvalue of the matrix 'A'. If desired, a transformation matrix 'P' can be returned, which is such that the Jordan canonical form is given by 'P⁻¹ A P'.

INPUT:

- ``base_ring`` - Ring in which to compute the Jordan form.
- ``sparse`` - (default ``False``) If ``sparse=True``, return a sparse matrix.
- ``subdivide`` - (default ``True``) If ``subdivide=True``, the subdivisions for the Jordan blocks in the matrix are shown.
- ``transformation`` - (default ``False``) If ``transformation=True``, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either '0 \times 0' or '1 \times 1'.

In the case of exact rings, this method does not compute any

evaluate

Click here to pop out
unprinted

File: /sage/abstract/devel/sage/sage/matrix/matrix2.pyx

Source Code (starting at line 5882):

```
def jordan_form(self, base_ring=None, sparse=False, subdivide=True, transformation=False):
```

```
    """
    Compute the Jordan normal form of this square matrix 'A', if it exists.
```

```
    This computation is performed in a naive way using the ranks of powers
    of 'A-xI', where 'x' is an eigenvalue of the matrix 'A'. If desired,
    a transformation matrix 'P' can be returned, which is such that the
    Jordan canonical form is given by 'P^{-1} A P'.
```

INPUT:

- ``base_ring`` - Ring in which to compute the Jordan form.
- ``sparse`` - (default ``False``) If ``sparse=True``, return a sparse matrix.
- ``subdivide`` - (default ``True``) If ``subdivide=True``, the subdivisions for the Jordan blocks in the matrix are shown.
- ``transformation`` - (default ``False``) If ``transformation=True``, computes also the transformation matrix.

NOTES:

```
    Currently, the Jordan normal form is not computed over inexact rings
    in any but the trivial cases when the matrix is either '0 \times 0'
    or '1 \times 1'.
```

```
    In the case of exact rings, this method does not compute any
```

evaluate

File: /sage/abstract/devel/sage/sage/matrix/matrix2.pyx

Click here to pop out
unprinted

Source Code (starting at line 5882):

```
def jordan_form(self, base_ring=None, sparse=False, subdivide=True, transformation=False):
    """
```

Compute the Jordan normal form of this square matrix 'A', if it exists.

This computation is performed in a naive way using the ranks of powers of 'A-xI', where 'x' is an eigenvalue of the matrix 'A'. If desired, a transformation matrix 'P' can be returned, which is such that the Jordan canonical form is given by 'P⁻¹ A P'.

INPUT:

- ``base_ring`` - Ring in which to compute the Jordan form.
- ``sparse`` - (default ``False``) If ``sparse=True``, return a sparse matrix.
- ``subdivide`` - (default ``True``) If ``subdivide=True``, the subdivisions for the Jordan blocks in the matrix are shown.
- ``transformation`` - (default ``False``) If ``transformation=True``, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either '0 \times 0' or '1 \times 1'.

In the case of exact rings, this method does not compute any

evaluate

File: /sage/abstract/devel/sage/sage/matrix/matrix2.pyx

Click here to pop out
unprinted

Source Code (starting at line 5882):

```
def jordan_form(self, base_ring=None, sparse=False, subdivide=True, transformation=False):
```

Compute the Jordan normal form of this square matrix 'A', if it exists.

This computation is performed in a naive way using the ranks of powers of 'A-xI', where 'x' is an eigenvalue of the matrix 'A'. If desired, a transformation matrix 'P' can be returned, which is such that the Jordan canonical form is given by 'P⁻¹ A P'.

INPUT:

- ``base_ring`` - Ring in which to compute the Jordan form.
- ``sparse`` - (default ``False``) If ``sparse=True``, return a sparse matrix.
- ``subdivide`` - (default ``True``) If ``subdivide=True``, the subdivisions for the Jordan blocks in the matrix are shown.
- ``transformation`` - (default ``False``) If ``transformation=True``, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either '0 \times 0' or '1 \times 1'.

In the case of exact rings, this method does not compute any

File: /sage/abstract/devel/sage/sage/matrix/matrix2.pyx

Source Code (starting at line 5882):

```
def jordan_form(self, base_ring=None, sparse=False, subdivide=True, transformation=False):
```

```
    """
    Compute the Jordan normal form of this square matrix 'A', if it exists.
```

```
    This computation is performed in a naive way using the ranks of powers
    of 'A-xI', where 'x' is an eigenvalue of the matrix 'A'. If desired,
    a transformation matrix 'P' can be returned, which is such that the
    Jordan canonical form is given by 'P^{-1} A P'.
```

INPUT:

- ``base_ring`` - Ring in which to compute the Jordan form.
- ``sparse`` - (default ``False``) If ``sparse=True``, return a sparse matrix.
- ``subdivide`` - (default ``True``) If ``subdivide=True``, the subdivisions for the Jordan blocks in the matrix are shown.
- ``transformation`` - (default ``False``) If ``transformation=True``, computes also the transformation matrix.

NOTES:

```
Currently, the Jordan normal form is not computed over inexact rings
in any but the trivial cases when the matrix is either '0 \times 0'
or '1 \times 1'.
```

```
In the case of exact rings, this method does not compute any
generalized form of the Jordan normal form, but is only able to
compute the result if the characteristic polynomial of the matrix
splits over the specific base ring.
```

compute the result if the characteristic polynomial of the matrix splits over the specific base ring.

EXAMPLES::

```
sage: a = matrix(ZZ,4,[1, 0, 0, 0, 0, 1, 0, 0, 1, \
-1, 1, 0, 1, -1, 1, 2]); a
```

```
[ 1 0 0 0]
[ 0 1 0 0]
[ 1 -1 1 0]
[ 1 -1 1 2]
```

```
sage: a.jordan_form()
```

```
[2|0 0|0]
[-+---+|]
[0|1 1|0]
[0|0 1|0]
[-+---+|]
[0|0 0|1]
```

```
sage: a.jordan_form(subdivide=False)
```

```
[2 0 0 0]
[0 1 1 0]
[0 0 1 0]
[0 0 0 1]
```

```
sage: b = matrix(ZZ,3,range(9)); b
```

```
[0 1 2]
[3 4 5]
[6 7 8]
```

```
sage: b.jordan_form()
```

```
Traceback (most recent call last):
```

```
...
RuntimeError: Some eigenvalue does not exist in Integer Ring.
```

```
sage: b.jordan_form(RealField(15))
```

```
Traceback (most recent call last):
```

```
...
ValueError: Jordan normal form not implemented over inexact rings.
```

```
[0 0 0|0 0|3 1|0 0|0]
[0 0 0|0 0|0 3|0 0|0]
[-----+-----+-----]
[0 0 0|0 0|0 0|3 1|0]
[0 0 0|0 0|0 0|0 3|0]
[-----+-----+-----]
[0 0 0|0 0|0 0|0 0|3]
sage: T * J * T**(-1) == A
True
sage: T.rank()
10
---
from sage.matrix.constructor import block_diagonal_matrix, jordan_block, diagonal_matrix
from sage.combinat.partition import Partition

if self.ncols() != self.nrows():
    raise ValueError, "Jordan normal form not implemented for non-square matrices."

# Set 'n' to the number of rows and handle trivial cases, regardless
# of the underlying ring.
n = self.nrows()
if n == 0:
    if not transformation:
        return self
    else:
        return self, self
elif n == 1:
    if not transformation:
        return self
    else:
        return self, self.parent().identity_matrix()

if (base_ring is None and not self.base_ring().is_exact()) or \
    (not base_ring is None and not base_ring.is_exact()):
    raise ValueError, "Jordan normal form not implemented over inexact rings."

if base_ring is None:
```


• Source code display

```
search_def('jordan')|
```

[evaluate](#)

• Low barriers to development

• Revision control log (Interrupt to stop serving)

```
hg_sage.browse()
```

• L^AT_EX integration (use typeset button)

```
var('x')  
f = x^3*e^-x  
f.integrate(x)
```

• More L^AT_EX integration

```
latex(A)
```

the relation $((\mu + 1)G == self$

EXAMPLES::

```
sage: A = matrix(ZZ, 3, [-1, 2, 5, -11, 1, 1, 1, -1, -3]); A
[ -1  2  5]
[-11  1  1]
[  1 -1 -3]
```

```
sage: G, mu = A.gram_schmidt()
```

```
sage: G
```

```
[  -1      2      5]
[ -52/5  -1/5   -2]
[ 2/187 36/187 -14/187]
```

```
sage: mu
```

```
[  0      0      0]
[ 3/5      0      0]
[-3/5 -7/187      0]
```

```
sage: G.row(0) * G.row(1)
```

```
0
```

```
sage: G.row(0) * G.row(2)
```

```
0
```

```
sage: G.row(1) * G.row(2)
```

```
0
```

The relation between mu and A is as follows::

```
sage: (mu + 1)*G == A
```

```
True
```

```
---
```

```
from sage.modules.misc import gram_schmidt
```

```
from constructor import matrix
```

```
Bstar, mu = gram_schmidt(self.rows())
```

```
return matrix(Bstar), mu
```

```
from sage.modules.misc import gram_schmidt
from constructor import matrix
Bstar, mu = gram_schmidt(self.rows())
return matrix(Bstar), mu
```

```
def jordan_form(self, base_ring=None, sparse=False, subdivide=True, transformation=False):
```

Compute the Jordan normal form of this square matrix 'A', if it exists.

This computation is performed in a naive way using the ranks of powers of 'A-xI', where 'x' is an eigenvalue of the matrix 'A'. If desired, a transformation matrix 'P' can be returned, which is such that the Jordan canonical form is given by 'P⁻¹ A P'.

INPUT:

- ``base_ring`` - Ring in which to compute the Jordan form.
- ``sparse`` - (default ``False``) If ``sparse=True``, return a sparse matrix.
- ``subdivide`` - (default ``True``) If ``subdivide=True``, the subdivisions for the Jordan blocks in the matrix are shown.
- ``transformation`` - (default ``False``) If ``transformation=True``, computes also the transformation matrix.

NOTES:

Currently, the Jordan normal form is not computed over inexact rings in any but the trivial cases when the matrix is either '0 \times 0' or '1 \times 1'.

In the case of exact rings, this method does not compute any generalized form of the Jordan normal form, but is only able to compute the result if the characteristic polynomial of the matrix


```
from sage.modules.misc import gram_schmidt
from constructor import matrix
Bstar, mu = gram_schmidt(self.rows())
return matrix(Bstar), mu
```

```
def jordan_form(self, base_ring=None, sparse=False, subdivide=True, transformation=False):
```

```
    """
    Compute the Jordan normal form of this square matrix 'A', if it exists.
```

```
    This computation is performed in a naive way using the ranks of powers
    of 'A-xI', where 'x' is an eigenvalue of the matrix 'A'. If desired,
    a transformation matrix 'P' can be returned, which is such that the
    Jordan canonical form is given by 'P^{-1} A P'.
```

INPUT:

- ``base_ring`` - Ring in which to compute the Jordan form.
- ``sparse`` - (default ``False``) If ``sparse=True``, return a sparse matrix.
- ``subdivide`` - (default ``True``) If ``subdivide=True``, the subdivisions for the Jordan blocks in the matrix are shown.
- ``transformation`` - (default ``False``) If ``transformation=True``, computes also the transformation matrix.

NOTES:

```
Currently, the Jordan normal form is not computed over inexact rings
in any but the trivial cases when the matrix is either '0 \times 0'
or '1 \times 1'.
```

```
In the case of exact rings, this method does not compute any
generalized form of the Jordan normal form, but is only able to
compute the result if the characteristic polynomial of the matrix
```

1. [matrix/constructor.py](#)
2. [matrix/matrix2.pyx](#)
3. [quadratic forms/quadratic form equivalence testing.py](#)
4. [quadratic forms/quadratic form local normal form.py](#)

- Low barriers to development
- Revision control log (Interrupt to stop serving)

```
hg_sage.browse()|
```

[evaluate](#)

- L^AT_EX integration (use typeset button)

```
var('x')
f = x^3*e^-x
f.integrate(x)
```

- More L^AT_EX integration

jsMath

1. [matrix/constructor.py](#)
2. [matrix/matrix2.pyx](#)
3. [quadratic forms/quadratic form equivalence testing.py](#)
4. [quadratic forms/quadratic form local normal form.py](#)



/home/sage/abstract/devel/sage-main

log

[less more](#) | [rev 15418: \(0\) -10000 -3000 -1000 -300 -100 -60 tip](#)

log	age	author	description
graph	3 months ago	Jeroen Demeyer	4.6.2.alpha3 default tip
tags	3 months ago	Jeroen Demeyer	Added tag 4.6.2.alpha3 for changeset bc99aa75148a
branches	3 months ago	Stefan Reiterer	Trac 9791: Updated Weave documentation, because Weave is now included in Scipy. 4.6.2.alpha3
changeset	7 months ago	David Loeffler	#9437: problems with matrix groups over non-field base rings
browse	3 months ago	Willem Jan Palenstijn	#10610: Fix tachyon's block=False
help	3 months ago	André Apitzsch	trac 10593: replace __getslice__
	3 months ago	Volker Braun	Trac 9918: Triangulate point configurations
	3 months ago	Joris Vankerschaver	Rebase of 10251: Bessel functions have small imaginary component
	4 months ago	Gagan Sekhon	trac_10457_new_func_HypEll_ov_ff.patch
	3 months ago	Alexandru Ghitza	#10686: speed up computation of T_p in characteristic p
	3 months ago	David Loeffler	#10450: doctest a minor bug fix in cuspidal/eisenstein submodules
	3 months ago	David Loeffler	#10450: problem computing Hecke matrices on subspaces of modular forms spaces
	5 months ago	André Apitzsch	trac 10449: DeprecationWarning for popen3 in ECM
	3 months ago	Karl-Dieter Crisman	Trac 7524 - document yet another option does get passed to save after #7981
	3 months ago	Jean-Pierre Flori	#10564: Non-regression test
	3 months ago	Minh Van Nguyen	#10697: documentation bug in linear programming tutorial
	3 months ago	Simon King	#1396 and #9599 Allow Singular's options in libSingular

Adding Spice to Your Research with Sage

Save Save & quit Discard & quit

last edited on May 10, 2011 07:06 AM by admin

File... Action... Data... sage Typeset Print Worksheet Edit Text Undo Share Publish

- Action...
- Interrupt
- Restart worksheet
- Save and quit worksheet
-
- Evaluate All
- Hide All Output
- Show All Output
- Delete All Output
-
- One Cell Mode
- Multi Cell Mode

Adding Spice to Your Research with Sage

Open Source Software for Mathematics

May 10, 2011

Rob Beezer

University of Puget Sound

Perimeter Institute for Theoretical Physics
Waterloo, Canada

jsMath


```
* Open your web browser to http://localhost:8200 *
*
*****
__SAGE__
```

- $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ integration (use typeset button)

```
var('x')
f = x^3*e^-x
f.integrate(x)
```

[evaluate](#)

- More $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ integration

```
latex(A)
```

- Integrated mini-editor, supports $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ snippets (shift-click above)
- Interactive demonstrations, explorations

jsMath

* Open your web browser to <http://localhost:8200> *

*

SAGE

- $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ integration (use typeset button)

```
var('x')
f = x^3*e^-x
f.integrate(x)
```

- More $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ integration

`latex(A)`

[evaluate](#)

- Integrated mini-editor, supports $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ snippets (shift-click above)

- Interactive demonstrations, explorations

* Open your web browser to http://localhost:8200 *

__SAGE__

• IAT_EX integration (use typeset button)

```
var('x')
f = x^3*e^-x
f.integrate(x)
```

$$-(x^3 + 3x^2 + 6x + 6)e^{(-x)}$$

• More IAT_EX integration

$\backslash\text{latex}(A)$

[evaluate](#)

• Integrated mini-editor, supports IAT_EX snippets (shift-click above)

• Interactive demonstrations, explorations

* Open your web browser to http://localhost:8200 *

__SAGE__

- $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ integration (use typeset button)

```
var('x')
f = x^3*e^-x
f.integrate(x)
```

$$-(x^3 + 3x^2 + 6x + 6)e^{-x}$$

- More $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ integration

$\text{\textbackslash latex}(A)$

[evaluate](#)

- Integrated mini-editor, supports $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ snippets (shift-click above)

- Interactive demonstrations, explorations

SAGE

- $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ integration (use typeset button)

```
var('x')
f = x^3*e^-x
f.integrate(x)
```

$$-(x^3 + 3x^2 + 6x + 6)e^{-x}$$

- More $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ integration

```
latex(A)
```

[evaluate](#)

- Integrated mini-editor, supports $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ snippets (shift-click above)

- Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
```

SAGE

- \LaTeX integration (use typeset button)

[evaluate](#)

```
var('x')
f = x^3*e^-x
f.integrate(x)
```

$$-(x^3 + 3x^2 + 6x + 6)e^{-x}$$

- More \LaTeX integration

`latex(A)`

- Integrated mini-editor, supports \LaTeX snippets (shift-click above)

- Interactive demonstrations explorations

jsMath

SAGE

- $\text{IAT}_\text{E}\text{X}$ integration (use typeset button)

[evaluate](#)

```
var('x')
f = x^3*e^-x
f.integrate(x)
```

$$-(x^3 + 3x^2 + 6x + 6)e^{(-x)}$$

- More $\text{IAT}_\text{E}\text{X}$ integration

latex(A)

- Integrated mini-editor, supports $\text{IAT}_\text{E}\text{X}$ snippets (shift-click above)

- Interactive demonstrations, explorations

jsMath

SAGE

- \LaTeX integration (use typeset button)

[evaluate](#)

```
var('x')
f = x^3*e^-x
f.integrate(x)
```

$$-(x^3 + 3x^2 + 6x + 6)e^{(-x)}$$

- More \LaTeX integration

latex(A)

- Integrated mini-editor, supports \LaTeX snippets (shift-click above)

- Interactive demonstrations, explorations

SAGE

- \LaTeX integration (use typeset button)

[evaluate](#)

```
var('x')
f = x^3*e^-x
f.integrate(x)
```

$$-(x^3 + 3x^2 + 6x + 6)e^{-x}$$

- More \LaTeX integration

latex(A)

jsMath

SAGE

- $\text{IAT}_\text{E}^\text{X}$ integration (use typeset button)

[evaluate](#)


```
var('x')
f = x^3*e^-x
f.integrate(x)
```































$$-(x^3 + 3x^2 + 6x + 6)e^{-x}$$

- More $\text{IAT}_\text{E}^\text{X}$ integration

jsMath

SAGE

• L^AT_EX integration (use typeset button)

Paragraph Font family Font size B I U ABC A                              

SAGE

• L^AT_EX integration (use typeset button)

$x^2 + y^2 \tan(x)$

evaluate

I

```
var('x')
f = x^3*e^-x
f.integrate(x)
```

$-(x^3 + 3x^2 + 6x + 6)e^{(-x)}$

jsMath

EX integration (use type set button)

$$x^2 + y^2 \tan(x)$$

[evaluate](#)

jsMath

```
var('x')  
f = x^3*e^-x  
f.integrate(x)
```

$$-\left(x^3 + 3x^2 + 6x + 6\right)e^{-x}$$

- More L^AT_EX integration

jsMath

$$-(x^3 + 3x^2 + 6x + 6)e^{-x}$$

- More L^AT_EX integration

latex(A)

```
\left(\begin{array}{rrrr}
3 & 2 & 0 & 1 \\
2 & 1 & 1 & 0 \\
1 & -1 & 5 & -3 \\
-2 & -3 & 5 & -4
\end{array}\right)
```

- Integrated mini-editor, supports L^AT_EX snippets (shift-click above)
- Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), 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))
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
```


• More L^AT_EX integration

latex(A)

```
\left(\begin{array}{rrrr}
3 & 2 & 0 & 1 \\
2 & 1 & 1 & 0 \\
1 & -1 & 5 & -3 \\
-2 & -3 & 5 & -4
\end{array}\right)
```

• Integrated mini-editor, supports L^AT_EX snippets (shift-click above)

• Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), 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))
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
    html('<h2>Compressed using %s singular values</h2>%i')
```

evaluate

jsMath

latex(A)

```
\left(\begin{array}{rrrr}
3 & 2 & 0 & 1 \\
2 & 1 & 1 & 0 \\
1 & -1 & 5 & -3 \\
-2 & -3 & 5 & -4
\end{array}\right)
```

- Integrated mini-editor, supports $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ snippets (shift-click above)
- Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), 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)
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
    html('<h2>Compressed using %s singular values</h2>%i')
```

[evaluate](#)

```
1 & -1 & 5 & -3 \\
-2 & -3 & 5 & -4 \\
\end{array}\right)
```

- Integrated mini-editor, supports L^AT_EX snippets (shift-click above)
- Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), 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))
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
    html('<h2>Compressed using %s singular values</h2>%i')
```

[evaluate](#)

6 Fast

- Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
```


- Integrated mini-editor, supports \LaTeX snippets (shift-click above)

- Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), 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))
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
    html('<h2>Compressed using %s singular values</h2>%i')
```

[evaluate](#)

6 Fast

- Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), 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))
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
    html('<h2>Compressed using %s singular values</h2>%i')
```

[evaluate](#)

6 Fast

Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
N(log(abs(d), 10), digits=5)
```

Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), 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))
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
    html('<h2>Compressed using %s singular values</h2>%i')
```

[evaluate](#)

6 Fast

- Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
N(log(abs(d), 10), digits=5)
```


Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), 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))
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
    html('<h2>Compressed using %s singular values</h2>%i')
```

[evaluate](#)

6 Fast

- Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
N(log(abs(d), 10), digits=5)
```

Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), 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))
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
    html('<h2>Compressed using %s singular values</h2>%i')
```

[evaluate](#)

6 Fast

- Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
N(log(abs(d), 10), digits=5)
```

Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), 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))
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
    html('<h2>Compressed using %s singular values</h2>%i')
```

[evaluate](#)

6 Fast

- Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
N(log(abs(d), 10), digits=5)
```


Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), 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))
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
    html('<h2>Compressed using %s singular values</h2>%i')
```

[evaluate](#)

6 Fast

- Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
N(log(abs(d), 10), digits=5)
```

Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), 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))
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
    html('<h2>Compressed using %s singular values</h2>%i')
```

[evaluate](#)

6 Fast

- Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
N(log(abs(d), 10), digits=5)
```

Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), 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))
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
    html('<h2>Compressed using %s singular values</h2>%i')
```

[evaluate](#)

6 Fast

- Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
N(log(abs(d), 10), digits=5)
```


Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), 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))
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
    html('<h2>Compressed using %s singular values</h2>%i')
```

[evaluate](#)

6 Fast

- Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
N(log(abs(d), 10), digits=5)
```

Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), 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))
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
    html('<h2>Compressed using %s singular values</h2>%i')
```

[evaluate](#)

6 Fast

Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
N(log(abs(d), 10), digits=5)
```

Interactive demonstrations, explorations

```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), 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))
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
    html('<h2>Compressed using %s singular values</h2>%i')
```

evaluate

6 Fast

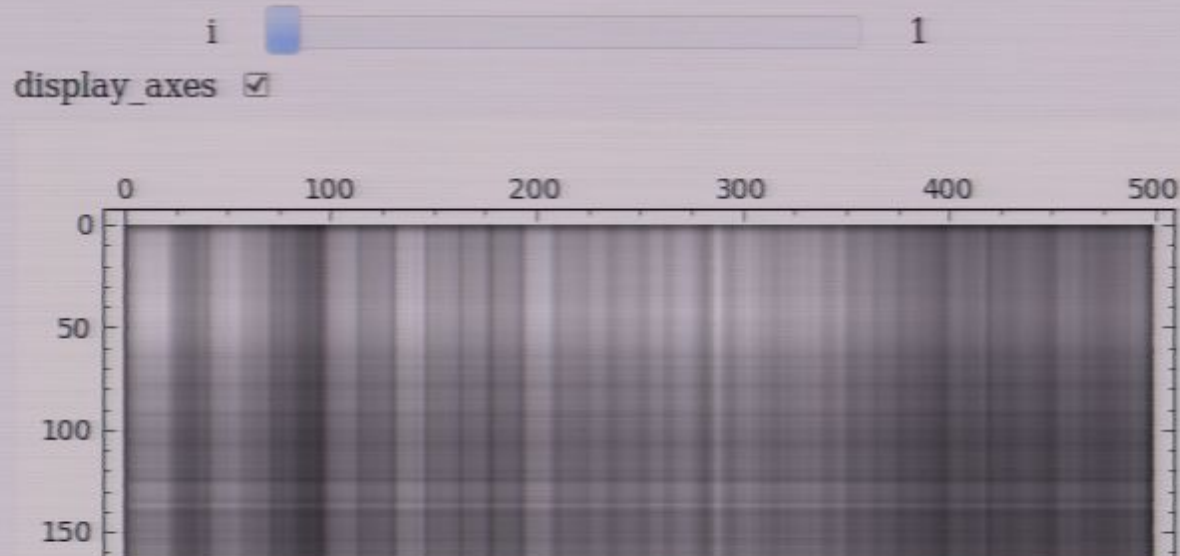
Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
N(log(abs(d), 10), digits=5)
```


Interactive demonstrations, explorations

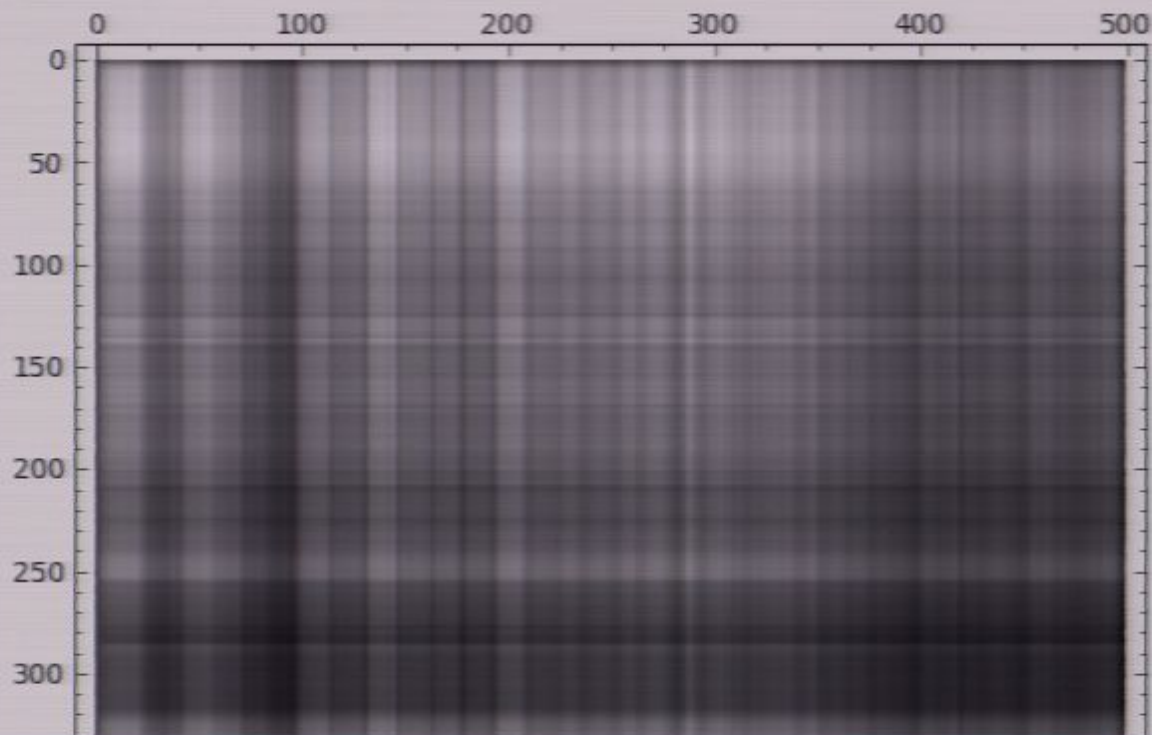
```
import pylab
A_image = pylab.mean(pylab.imread(DATA + 'mystery.png'), 2)
@interact
def svd_image(i=(1,(1..50)), display_axes=True):
    u,s,v = pylab.linalg.svd(A_image)
    A = sum(s[j]*pylab.outer(u[0:,j], v[j,0:])) for j in range(i)
    show(matrix_plot(A), axes=display_axes, figsize=(11,5))
    html('<h2>Compressed using %s singular values</h2>%i')
```

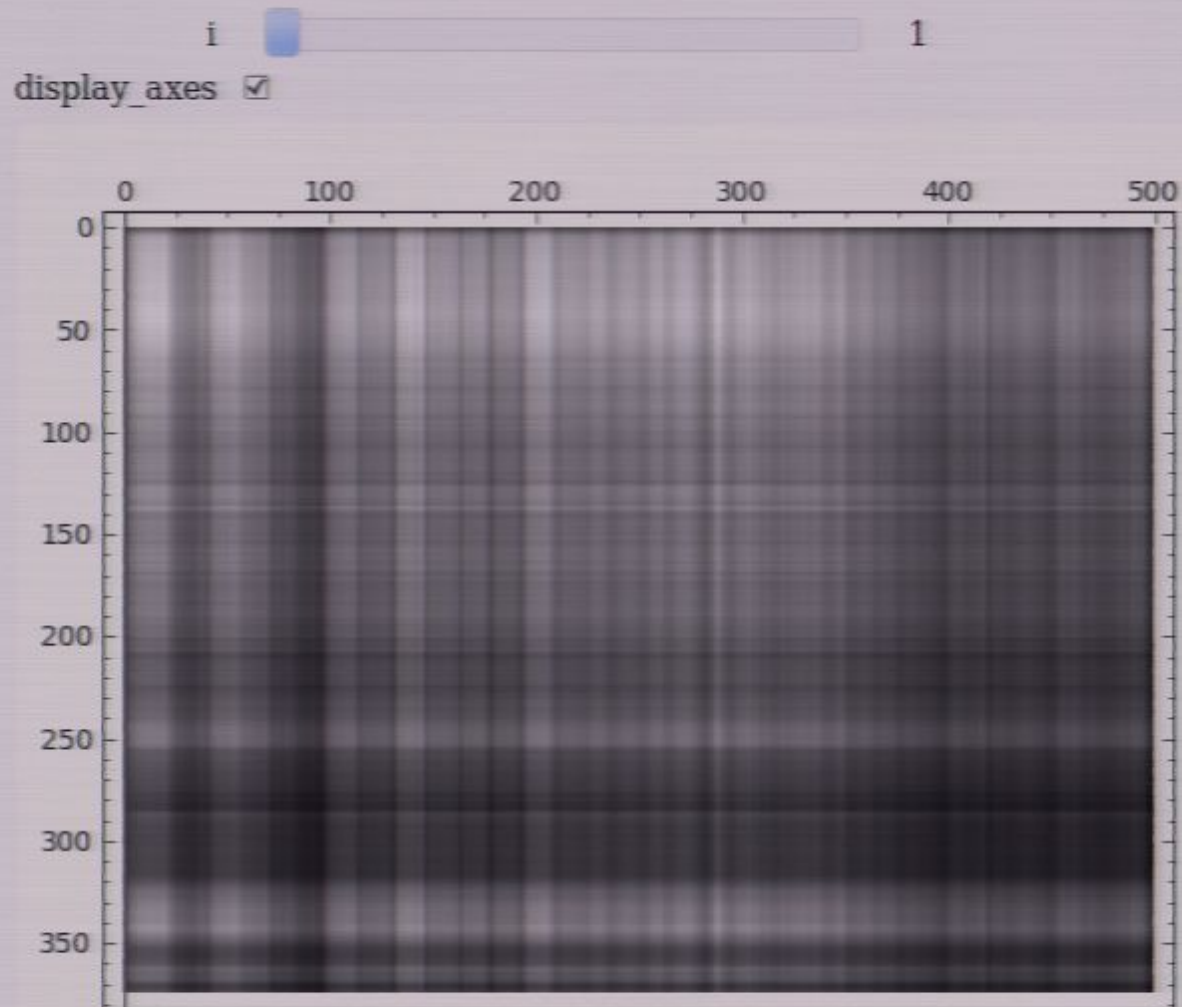


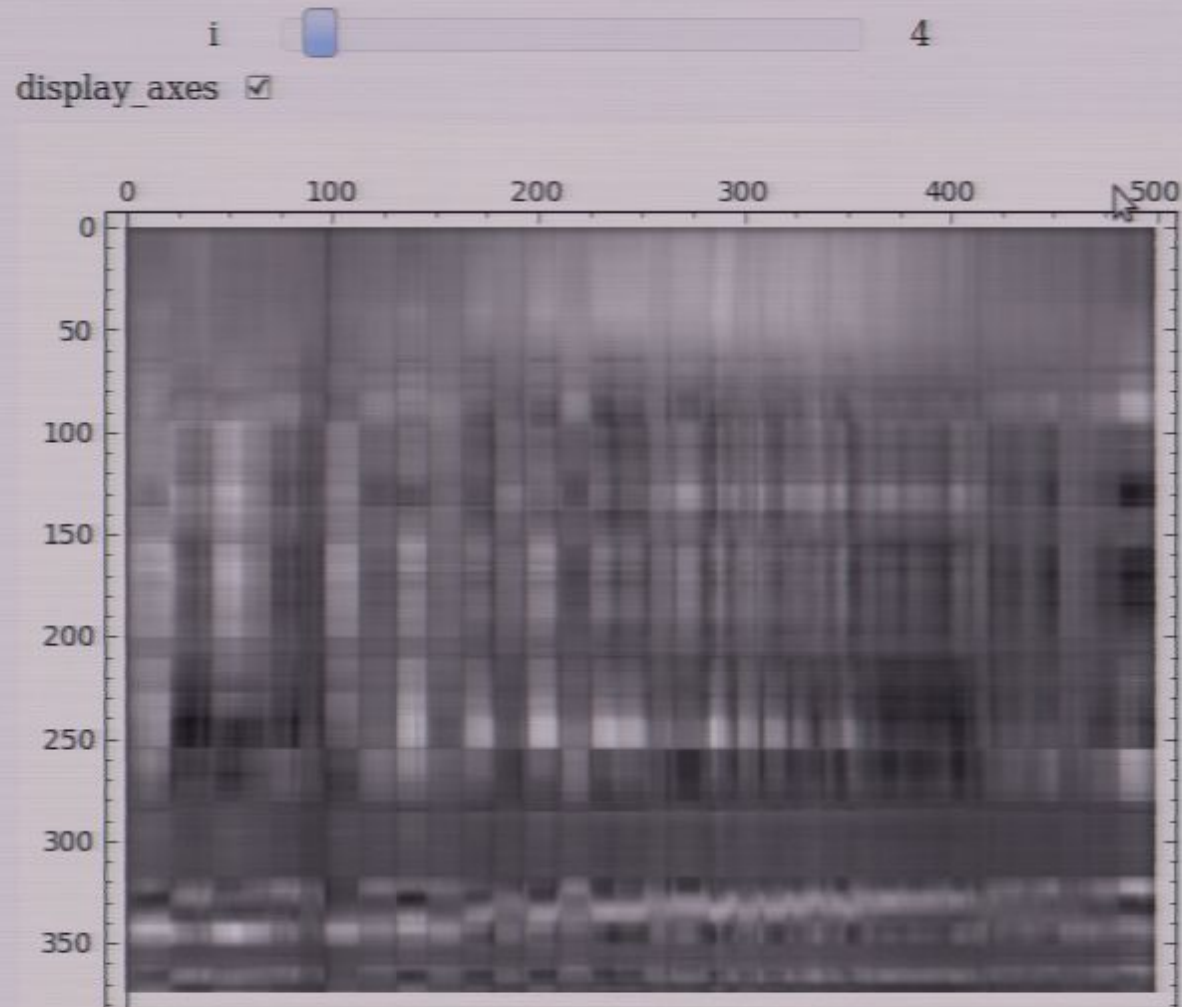
```
show(matrix_plot(A), axes=display_axes, figsize=(11,5))  
html('<h2>Compressed using %s singular values</h2>'%i)
```

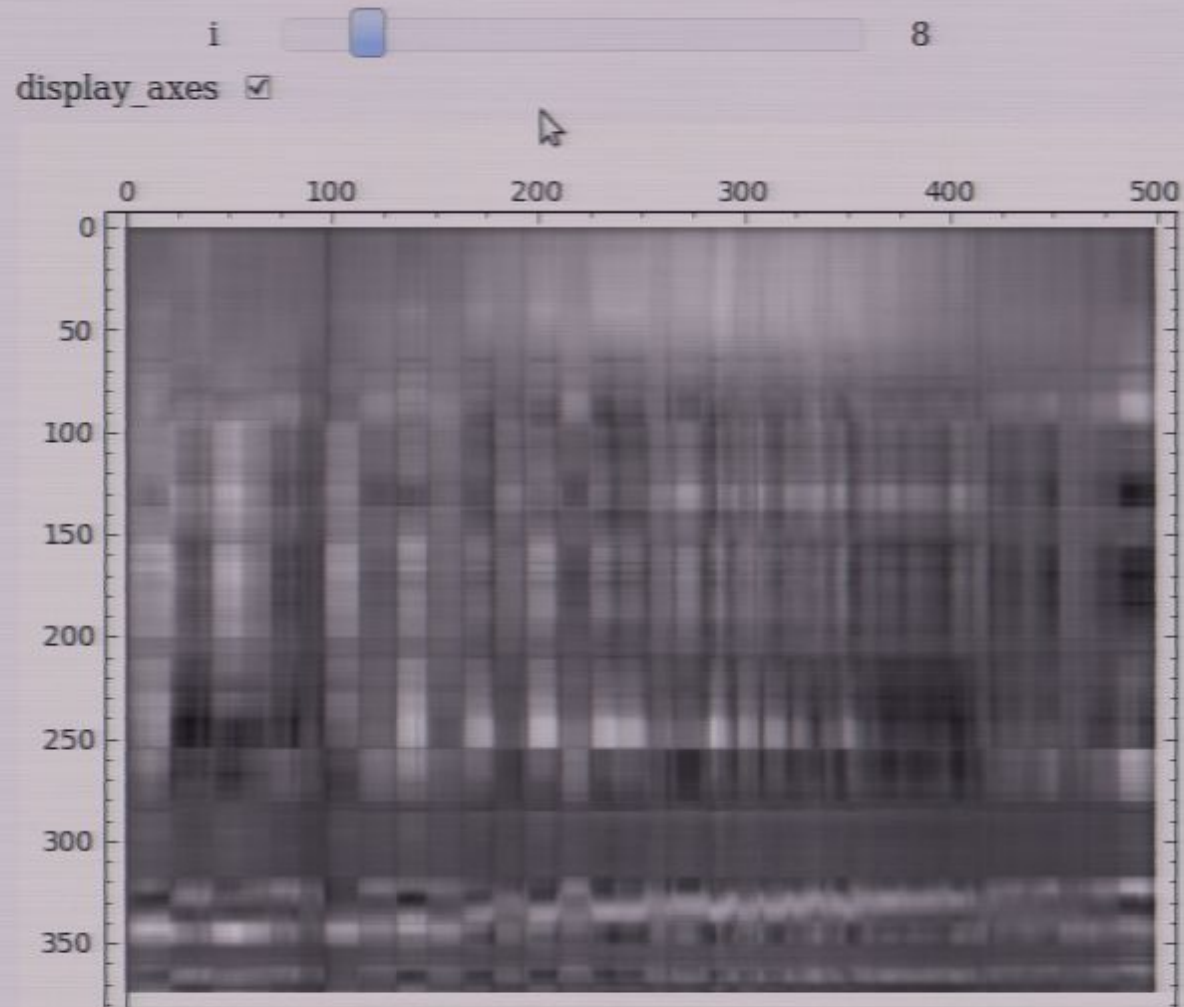
i 1

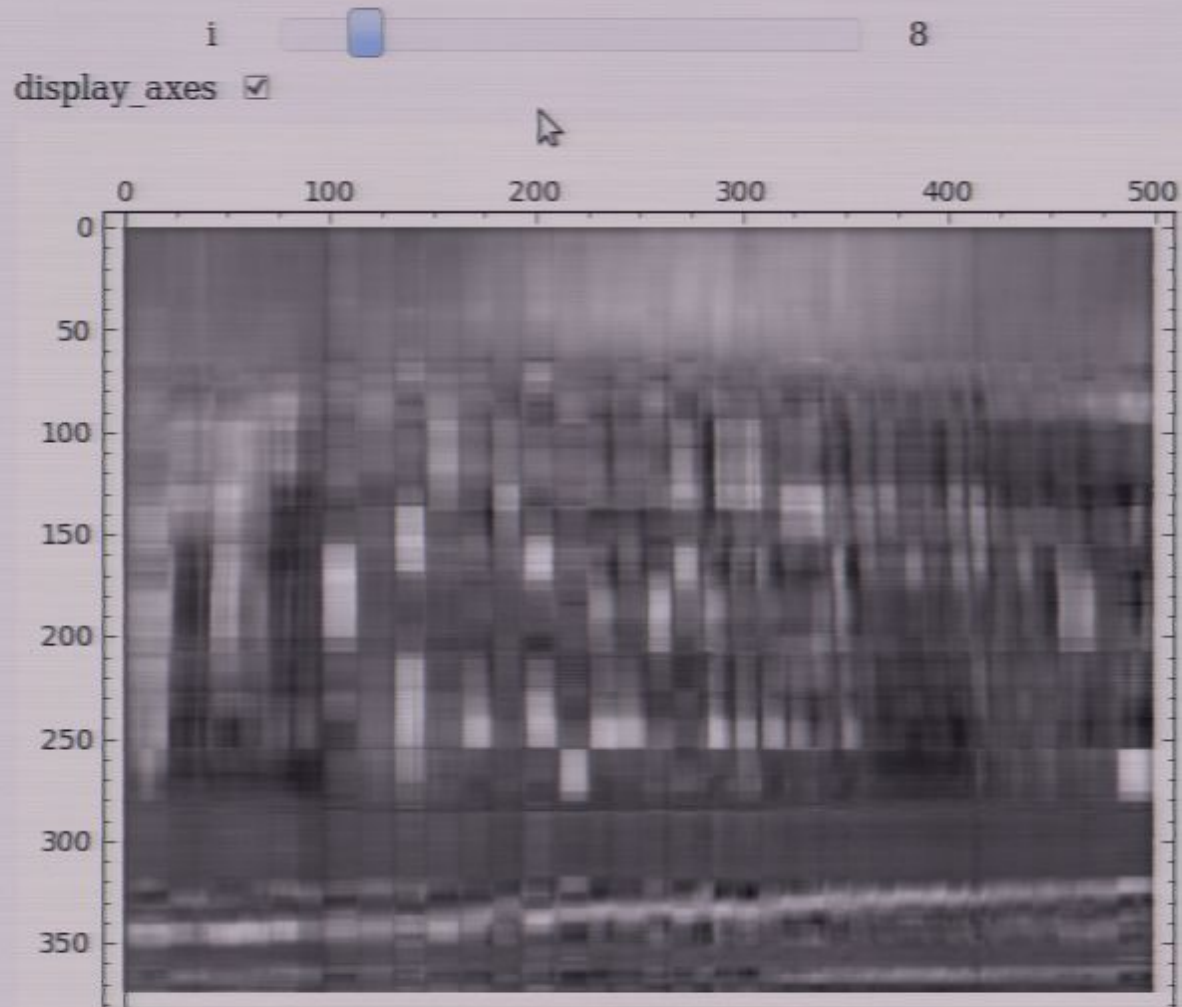
display_axes ☒

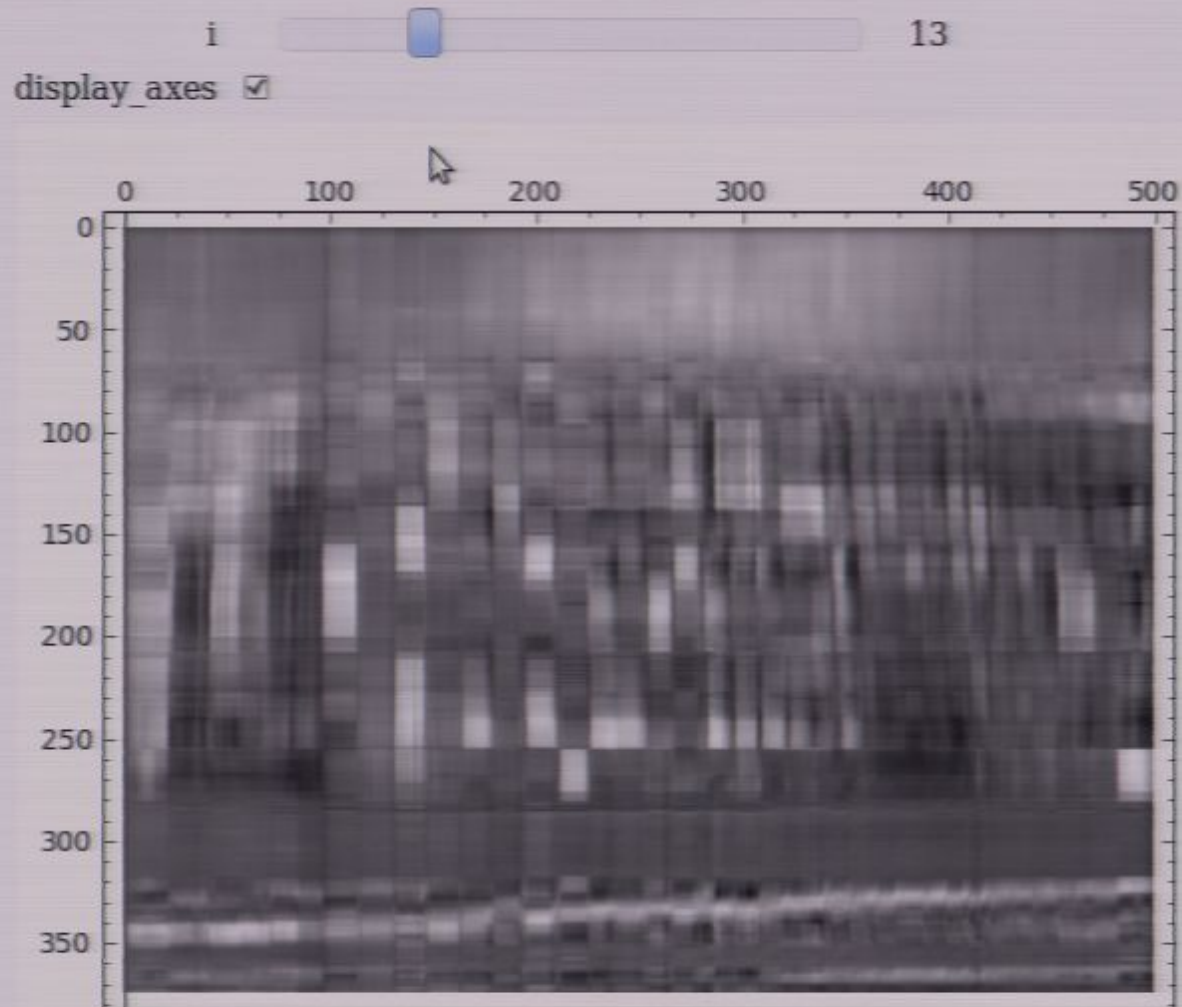


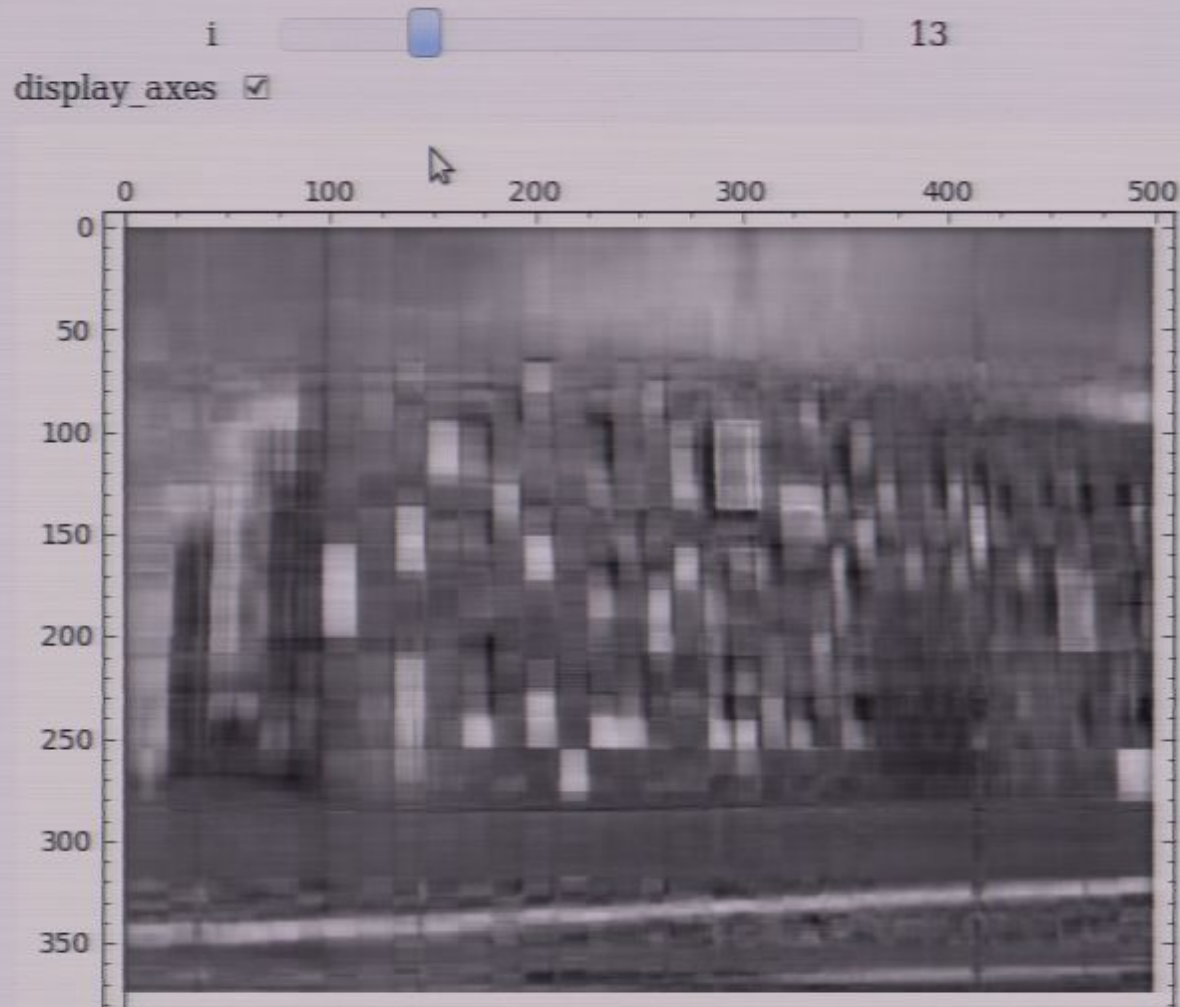


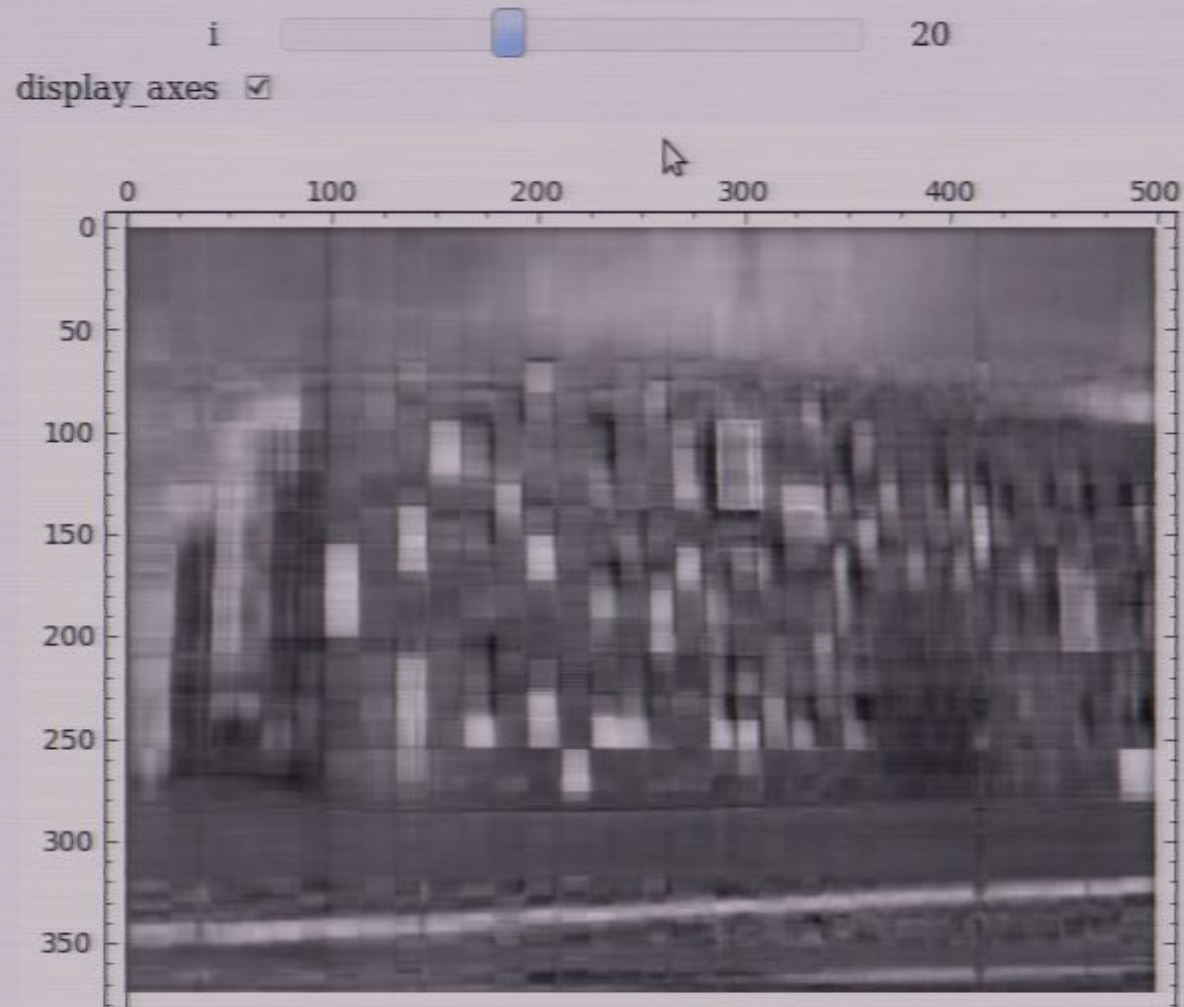


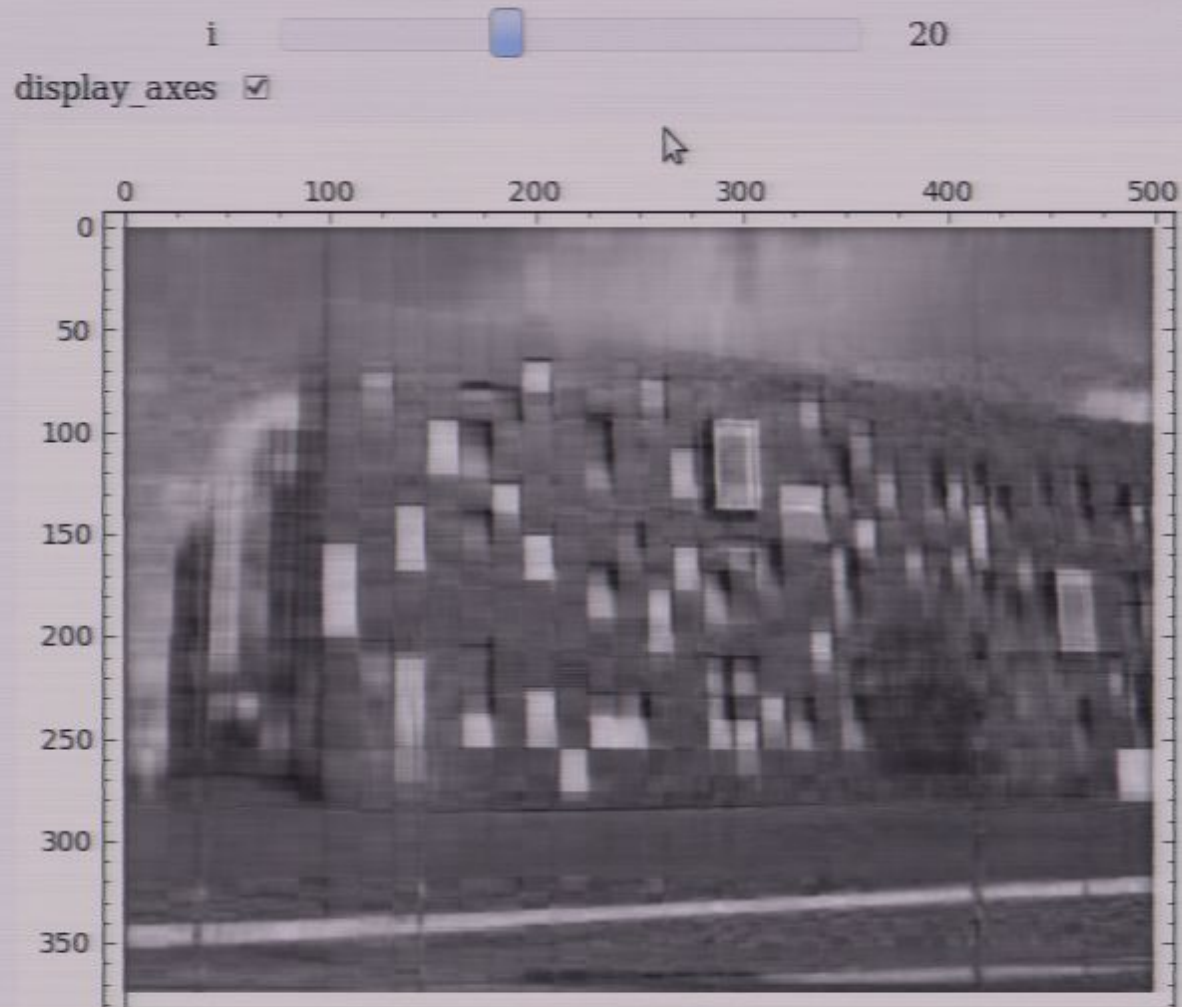


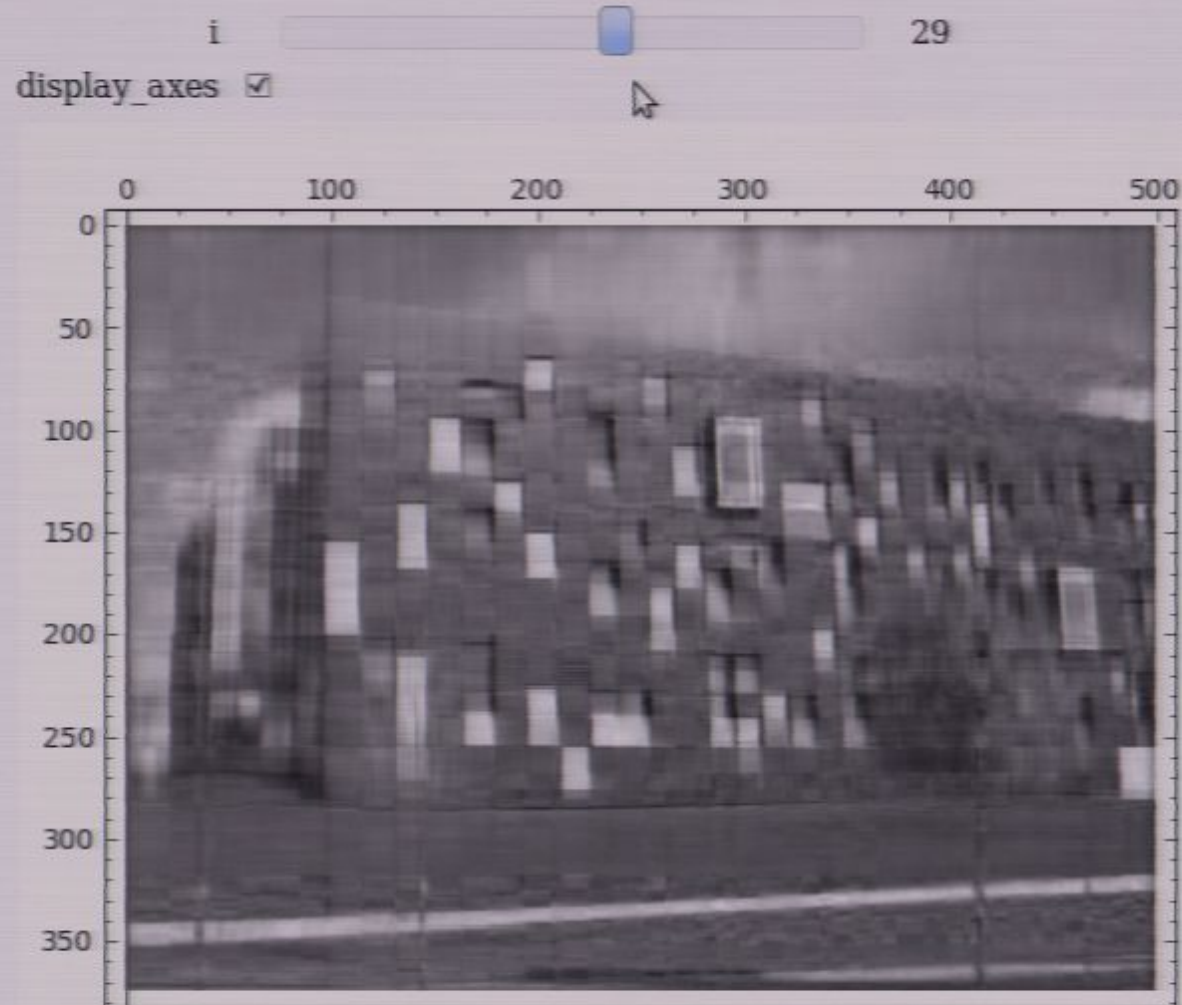




















compressed using 44 singular values

6 Fast

- Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

[evaluate](#)

```
N(log(abs(d), 10), digits=5)
```

- Precise timings

```
timeit("A.determinant()", number=2, repeat=3)
```

6 Fast

- Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

[evaluate](#)

```
N(log(abs(d), 10), digits=5)
```

- Precise timings

```
timeit("A.determinant()", number=2, repeat=3)
```

7 Easy

6 Fast

- Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d|
```

[evaluate](#)

```
N(log(abs(d), 10), digits=5)
```

- Precise timings

```
timeit("A.determinant()", number=2, repeat=3)
```

7 Easy

6 Fast

- Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
N(log(abs(d), 10), digits=5)
```

[evaluate](#)

- Precise timings

```
timeit("A.determinant()", number=2, repeat=3)
```

7 Easy

6 Fast

- Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
729835381172507117332987738336635716491681295469509506865202982765116676\
427109574924403111349553119207781579831530780253621094543530238628866604\
706834255490502003683831236032756684401475914597030059949713778227730055\
774320327570564319506286696289477656962887790113657322358115501098708488\
709120003156303148787031815686306477821428151369253783900528568741000248\
223116954848898058801511940412781117747587907947874590415481787388543225\
291878906183387084657713228450638295854373792772543108310916175674061700\
647927896074980998658867493971950381381999945255979502294701327754520362\
706883265703685974441505248311951529973709776232126416592118552802423497\
797655524768517227298420954213191688826488808764622367512532448348739975\
927282079160079865255159571779037480966246646359936117043648239296670715\
316596193764571315232596702360243882575416133962477212800507013251623718\
684487167346950563618120490848967159302240473174454358039620663614700902\
056937214195862355104244613114600307813181370215019098482731770592148047\
639774286024776595392037168877425938083423952086440337703600444731414978\
```


6 Fast

- Exact determinant of a 500×500 integer matrix

```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

I

```
729835381172507117332987738336635716491681295469509506865202982765116676\
427109574924403111349553119207781579831530780253621094543530238628866604\
706834255490502003683831236032756684401475914597030059949713778227730055\
774320327570564319506286696289477656962887790113657322358115501098708488\
709120003156303148787031815686306477821428151369253783900528568741000248\
223116954848898058801511940412781117747587907947874590415481787388543225\
291878906183387084657713228450638295854373792772543108310916175674061700\
647927896074980998658867493971950381381999945255979502294701327754520362\
706883265703685974441505248311951529973709776232126416592118552802423497\
797655524768517227298420954213191688826488808764622367512532448348739975\
927282079160079865255159571779037480966246646359936117043648239296670715\
316596193764571315232596702360243882575416133962477212800507013251623718\
684487167346950563618120490848967159302240473174454358039620663614700902\
056937214195862355104244613114600307813181370215019098482731770592148047\
639774286024776595392037168877425938083423952086440337703600444731414978\
```

jsMath


```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
729835381172507117332987738336635716491681295469509506865202982765116676\
427109574924403111349553119207781579831530780253621094543530238628866604\
706834255490502003683831236032756684401475914597030059949713778227730055\
774320327570564319506286696289477656962887790113657322358115501098708488\
709120003156303148787031815686306477821428151369253783900528568741000248\
223116954848898058801511940412781117747587907947874590415481787388543225\
291878906183387084657713228450638295854373792772543108310916175674061700\
647927896074980998658867493971950381381999945255979502294701327754520362\
706883265703685974441505248311951529973709776232126416592118552802423497\
797655524768517227298420954213191688826488808764622367512532448348739975\
927282079160079865255159571779037480966246646359936117043648239296670715\
316596193764571315232596702360243882575416133962477212800507013251623718\
684487167346950563618120490848967159302240473174454358039620663614700902\
056937214195862355104244613114600307813181370215019098482731770592148047\
639774286024776595392037168877425938083423952086440337703600444731414978\
542980926213411231089598335495995207599992702200856607059633796210037289\
208353072818344601143269724981158664175122639741467036077022063019438203\
151243666020271302854033762614632403443955458489430642791817205684506337\
4818162564995296936792608382710177645983600002792
```

```
N(log(abs(d), 10), digits=5)
```

[evaluate](#)

jsMath


```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
729835381172507117332987738336635716491681295469509506865202982765116676\
427109574924403111349553119207781579831530780253621094543530238628866604\
706834255490502003683831236032756684401475914597030059949713778227730055\
774320327570564319506286696289477656962887790113657322358115501098708488\
709120003156303148787031815686306477821428151369253783900528568741000248\
223116954848898058801511940412781117747587907947874590415481787388543225\
291878906183387084657713228450638295854373792772543108310916175674061700\
647927896074980998658867493971950381381999945255979502294701327754520362\
706883265703685974441505248311951529973709776232126416592118552802423497\
797655524768517227298420954213191688826488808764622367512532448348739975\
927282079160079865255159571779037480966246646359936117043648239296670715\
316596193764571315232596702360243882575416133962477212800507013251623718\
684487167346950563618120490848967159302240473174454358039620663614700902\
056937214195862355104244613114600307813181370215019098482731770592148047\
639774286024776595392037168877425938083423952086440337703600444731414978\
542980926213411231089598335495995207599992702200856607059633796210037289\
208353072818344601143269724981158664175122639741467036077022063019438203\
151243666020271302854033762614632403443955458489430642791817205684506337\
4818162564995296936792608382710177645983600002792
```

```
N(log(abs(d), 10), digits=5)
```

evaluate

jsMath


```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
729835381172507117332987738336635716491681295469509506865202982765116676\
427109574924403111349553119207781579831530780253621094543530238628866604\
706834255490502003683831236032756684401475914597030059949713778227730055\
774320327570564319506286696289477656962887790113657322358115501098708488\
709120003156303148787031815686306477821428151369253783900528568741000248\
223116954848898058801511940412781117747587907947874590415481787388543225\
291878906183387084657713228450638295854373792772543108310916175674061700\
647927896074980998658867493971950381381999945255979502294701327754520362\
706883265703685974441505248311951529973709776232126416592118552802423497\
797655524768517227298420954213191688826488808764622367512532448348739975\
927282079160079865255159571779037480966246646359936117043648239296670715\
316596193764571315232596702360243882575416133962477212800507013251623718\
684487167346950563618120490848967159302240473174454358039620663614700902\
056937214195862355104244613114600307813181370215019098482731770592148047\
639774286024776595392037168877425938083423952086440337703600444731414978\
542980926213411231089598335495995207599992702200856607059633796210037289\
208353072818344601143269724981158664175122639741467036077022063019438203\
151243666020271302854033762614632403443955458489430642791817205684506337\
4818162564995296936792608382710177645983600002792
```

```
N(log(abs(d), 10), digits=5)
```

[evaluate](#)

jsMath


```
A = random_matrix(ZZ, 500)
d = A.determinant()
d
```

```
729835381172507117332987738336635716491681295469509506865202982765116676\
427109574924403111349553119207781579831530780253621094543530238628866604\
706834255490502003683831236032756684401475914597030059949713778227730055\
774320327570564319506286696289477656962887790113657322358115501098708488\
709120003156303148787031815686306477821428151369253783900528568741000248\
223116954848898058801511940412781117747587907947874590415481787388543225\
291878906183387084657713228450638295854373792772543108310916175674061700\
647927896074980998658867493971950381381999945255979502294701327754520362\
706883265703685974441505248311951529973709776232126416592118552802423497\
797655524768517227298420954213191688826488808764622367512532448348739975\
927282079160079865255159571779037480966246646359936117043648239296670715\
316596193764571315232596702360243882575416133962477212800507013251623718\
684487167346950563618120490848967159302240473174454358039620663614700902\
056937214195862355104244613114600307813181370215019098482731770592148047\
639774286024776595392037168877425938083423952086440337703600444731414978\
542980926213411231089598335495995207599992702200856607059633796210037289\
208353072818344601143269724981158664175122639741467036077022063019438203\
151243666020271302854033762614632403443955458489430642791817205684506337\
4818162564995296936792608382710177645983600002792
```

```
N(log(abs(d), 10), digits=5)
```

evaluate



jsMath


```
706834255490502003683831236032756684401475914597030059949713778227730055\
774320327570564319506286696289477656962887790113657322358115501098708488\
709120003156303148787031815686306477821428151369253783900528568741000248\
223116954848898058801511940412781117747587907947874590415481787388543225\
291878906183387084657713228450638295854373792772543108310916175674061700\
647927896074980998658867493971950381381999945255979502294701327754520362\
706883265703685974441505248311951529973709776232126416592118552802423497\
797655524768517227298420954213191688826488808764622367512532448348739975\
927282079160079865255159571779037480966246646359936117043648239296670715\
316596193764571315232596702360243882575416133962477212800507013251623718\
684487167346950563618120490848967159302240473174454358039620663614700902\
056937214195862355104244613114600307813181370215019098482731770592148047\
639774286024776595392037168877425938083423952086440337703600444731414978\
542980926213411231089598335495995207599992702200856607059633796210037289\
208353072818344601143269724981158664175122639741467036077022063019438203\
151243666020271302854033762614632403443955458489430642791817205684506337\
4818162564995296936792608382710177645983600002792
```

$N(\log(\text{abs}(d)), 10), \text{digits}=5)$

1344.9

• Precise timings

```
timeit("A.determinant()", number=2, repeat=3)
```

evaluate

Pirsa: 11050014

Page 210/394

Find: def jordan

Previous Next Highlight all Match case


```
706834255490502003683831236032756684401475914597030059949713778227730055\
774320327570564319506286696289477656962887790113657322358115501098708488\
709120003156303148787031815686306477821428151369253783900528568741000248\
223116954848898058801511940412781117747587907947874590415481787388543225\
291878906183387084657713228450638295854373792772543108310916175674061700\
647927896074980998658867493971950381381999945255979502294701327754520362\
706883265703685974441505248311951529973709776232126416592118552802423497\
797655524768517227298420954213191688826488808764622367512532448348739975\
927282079160079865255159571779037480966246646359936117043648239296670715\
316596193764571315232596702360243882575416133962477212800507013251623718\
684487167346950563618120490848967159302240473174454358039620663614700902\
056937214195862355104244613114600307813181370215019098482731770592148047\
639774286024776595392037168877425938083423952086440337703600444731414978\
542980926213411231089598335495995207599992702200856607059633796210037289\
208353072818344601143269724981158664175122639741467036077022063019438203\
151243666020271302854033762614632403443955458489430642791817205684506337\
4818162564995296936792608382710177645983600002792
```

$N(\log(\text{abs}(d), 10), \text{digits}=5)$

1344.9

• Precise timings

`timeit("A.determinant()", number=2, repeat=3)`

evaluate

Pirsa: 11050014

Page 211/394

Find: def jordan

Previous Next Highlight all Match case


```
706834255490502003683831236032756684401475914597030059949713778227730055\
774320327570564319506286696289477656962887790113657322358115501098708488\
709120003156303148787031815686306477821428151369253783900528568741000248\
223116954848898058801511940412781117747587907947874590415481787388543225\
291878906183387084657713228450638295854373792772543108310916175674061700\
647927896074980998658867493971950381381999945255979502294701327754520362\
706883265703685974441505248311951529973709776232126416592118552802423497\
797655524768517227298420954213191688826488808764622367512532448348739975\
927282079160079865255159571779037480966246646359936117043648239296670715\
316596193764571315232596702360243882575416133962477212800507013251623718\
684487167346950563618120490848967159302240473174454358039620663614700902\
056937214195862355104244613114600307813181370215019098482731770592148047\
639774286024776595392037168877425938083423952086440337703600444731414978\
542980926213411231089598335495995207599992702200856607059633796210037289\
208353072818344601143269724981158664175122639741467036077022063019438203\
151243666020271302854033762614632403443955458489430642791817205684506337\
4818162564995296936792608382710177645983600002792
```

`N(log(abs(d), 10), digits=5)`

1344.9

• Precise timings

`timeit("A.determinant()", number=2, repeat=3)`

evaluate

jsMath

Pirsa: 11050014

Page 212/394

Find: def jordan

Previous Next Highlight all Match case


```
706834255490502003683831236032756684401475914597030059949713778227730055\
774320327570564319506286696289477656962887790113657322358115501098708488\
709120003156303148787031815686306477821428151369253783900528568741000248\
223116954848898058801511940412781117747587907947874590415481787388543225\
291878906183387084657713228450638295854373792772543108310916175674061700\
647927896074980998658867493971950381381999945255979502294701327754520362\
706883265703685974441505248311951529973709776232126416592118552802423497\
797655524768517227298420954213191688826488808764622367512532448348739975\
927282079160079865255159571779037480966246646359936117043648239296670715\
316596193764571315232596702360243882575416133962477212800507013251623718\
684487167346950563618120490848967159302240473174454358039620663614700902\
056937214195862355104244613114600307813181370215019098482731770592148047\
639774286024776595392037168877425938083423952086440337703600444731414978\
542980926213411231089598335495995207599992702200856607059633796210037289\
208353072818344601143269724981158664175122639741467036077022063019438203\
151243666020271302854033762614632403443955458489430642791817205684506337\
4818162564995296936792608382710177645983600002792
```

$N(\log(\text{abs}(d)), 10), \text{digits}=5)$

1344.9

Precise timings

```
timeit("A.determinant()", number=2, repeat=3)
```

evaluate

Pirsa: 11050014

Page 213/394

Find: def jordan

Previous Next Highlight all Match case


```
706834255490502003683831236032756684401475914597030059949713778227730055\
774320327570564319506286696289477656962887790113657322358115501098708488\
709120003156303148787031815686306477821428151369253783900528568741000248\
223116954848898058801511940412781117747587907947874590415481787388543225\
291878906183387084657713228450638295854373792772543108310916175674061700\
647927896074980998658867493971950381381999945255979502294701327754520362\
706883265703685974441505248311951529973709776232126416592118552802423497\
797655524768517227298420954213191688826488808764622367512532448348739975\
927282079160079865255159571779037480966246646359936117043648239296670715\
316596193764571315232596702360243882575416133962477212800507013251623718\
684487167346950563618120490848967159302240473174454358039620663614700902\
056937214195862355104244613114600307813181370215019098482731770592148047\
639774286024776595392037168877425938083423952086440337703600444731414978\
542980926213411231089598335495995207599992702200856607059633796210037289\
208353072818344601143269724981158664175122639741467036077022063019438203\
151243666020271302854033762614632403443955458489430642791817205684506337\
4818162564995296936792608382710177645983600002792
```

$N(\log(\text{abs}(d)), 10), \text{digits}=5)$

1344.9

Precise timings

`timeit("A.determinant()", number=2, repeat=3)`

evaluate

Pirsa: 11050014

Page 214/394

Find: def jordan

Previous Next Highlight all Match case


```
706834255490502003683831236032756684401475914597030059949713778227730055\
774320327570564319506286696289477656962887790113657322358115501098708488\
709120003156303148787031815686306477821428151369253783900528568741000248\
223116954848898058801511940412781117747587907947874590415481787388543225\
291878906183387084657713228450638295854373792772543108310916175674061700\
647927896074980998658867493971950381381999945255979502294701327754520362\
706883265703685974441505248311951529973709776232126416592118552802423497\
797655524768517227298420954213191688826488808764622367512532448348739975\
927282079160079865255159571779037480966246646359936117043648239296670715\
316596193764571315232596702360243882575416133962477212800507013251623718\
684487167346950563618120490848967159302240473174454358039620663614700902\
056937214195862355104244613114600307813181370215019098482731770592148047\
639774286024776595392037168877425938083423952086440337703600444731414978\
542980926213411231089598335495995207599992702200856607059633796210037289\
208353072818344601143269724981158664175122639741467036077022063019438203\
151243666020271302854033762614632403443955458489430642791817205684506337\
4818162564995296936792608382710177645983600002792
```

$N(\log(\text{abs}(d)), 10), \text{digits}=5)$

1344.9

Precise timings

```
timeit("A.determinant()", number=2, repeat=3)
```

evaluate

Pirsa: 11050014

Page 215/394

Find: def jordan

Previous Next Highlight all Match case


```
316596193764571315232596702360243882575416133962477212800507013251623718\
684487167346950563618120490848967159302240473174454358039620663614700902\
056937214195862355104244613114600307813181370215019098482731770592148047\
639774286024776595392037168877425938083423952086440337703600444731414978\
542980926213411231089598335495995207599992702200856607059633796210037289\
208353072818344601143269724981158664175122639741467036077022063019438203\
151243666020271302854033762614632403443955458489430642791817205684506337\
4818162564995296936792608382710177645983600002792
```

`N(log(abs(d), 10), digits=5)`

1344.9

Precise timings

`timeit("A.determinant()", number=2, repeat=3)`

evaluate

7 Easy

4x4 matrix entries

`entries = [[1, -2, -4, 4], [1, -2, -3, 4], [0, -2, -6, 6], [-1, 2, 1, -4]]`

```
639774286024776595392037168877425938083423952086440337703600444731414978\
542980926213411231089598335495995207599992702200856607059633796210037289\
208353072818344601143269724981158664175122639741467036077022063019438203\
151243666020271302854033762614632403443955458489430642791817205684506337\
4818162564995296936792608382710177645983600002792
```

$N(\log(\text{abs}(d)), 10), \text{digits}=5)$

1344.9

• Precise timings

`timeit("A.determinant()", number=2, repeat=3)|`

[evaluate](#)

7 Easy

• 4×4 matrix entries

`entries = [[1, -2, -4, 4], [1, -2, -3, 4], [0, -2, -6, 6], [-1, 2, 1, -4]]`

jsMath


```
639774286024776595392037168877425938083423952086440337703600444731414978\  
542980926213411231089598335495995207599992702200856607059633796210037289\  
208353072818344601143269724981158664175122639741467036077022063019438203\  
151243666020271302854033762614632403443955458489430642791817205684506337\  
4818162564995296936792608382710177645983600002792
```

`N(log(abs(d), 10), digits=5)`

1344.9

• Precise timings

`timeit("A.determinant()", number=2, repeat=3)`

7 Easy

• 4x4 matrix entries

`entries = [[1, -2, -4, 4], [1, -2, -3, 4], [0, -2, -6, 6], [-1, 2, 1, -4]]`

[evaluate](#)

jsMath

```
639774286024776595392037168877425938083423952086440337703600444731414978\
542980926213411231089598335495995207599992702200856607059633796210037289\
208353072818344601143269724981158664175122639741467036077022063019438203\
151243666020271302854033762614632403443955458489430642791817205684506337\
4818162564995296936792608382710177645983600002792
```

`N(log(abs(d), 10), digits=5)`

1344.9

• Precise timings

`timeit("A.determinant()", number=2, repeat=3)`

2 loops, best of 3: 2.37 s per loop

7 Easy

• 4x4 matrix entries

`entries = [[1, -2, -4, 4], [1, -2, -3, 4], [0, -2, -6, 6], [-1, 2, 1, -4]]`

[evaluate](#)

jsMath

2 loops, best of 3: 2.37 s per loop

7 Easy

- 4×4 matrix entries

```
entries = [[1, -2, -4, 4], [1, -2, -3, 4], [0, -2, -6, 6], [-1, 2, 1, -4]]
```

[evaluate](#)

- Over the integers

```
A = matrix(ZZ, entries)
A.echelon_form()
```

- Over the rationals

```
A = matrix(QQ, entries)
A.echelon_form()
```


Easy

• 4×4 matrix entries

```
entries = [[1, -2, -4, 4], [1, -2, -3, 4], [0, -2, -6, 6], [-1, 2, 1, -4]]
```

[evaluate](#)

• Over the integers

```
A = matrix(ZZ, entries)
A.echelon_form()
```

• Over the rationals

```
A = matrix(QQ, entries)
A.echelon_form()
```

• With floating point double-precision reals

Easy

- 4×4 matrix entries

```
entries = [[1, -2, -4, 4], [1, -2, -3, 4], [0, -2, -6, 6], [-1, 2, 1, -4]]
```

- Over the integers

```
A = matrix(ZZ, entries)
A.echelon_form()
```

[evaluate](#)

- Over the rationals

```
A = matrix(QQ, entries)
A.echelon_form()
```

- With floating point double-precision reals

jsMath

```
entries = [[1, -2, -4, 4], [1, -2, -3, 4], [0, -2, -6, 6], [-1, 2, 1, -4]]
```

• Over the integers

```
A = matrix(ZZ, entries)
A.echelon_form()
```

[evaluate](#)

• Over the rationals

```
A = matrix(QQ, entries)
A.echelon_form()
```

• With floating point double-precision reals

```
A = matrix(RDF, entries)
P, L, U = A.LU()
print L.round(5)
```



```
entries = [[1, -2, -4, 4], [1, -2, -3, 4], [0, -2, -6, 6], [-1, 2, 1, -4]]
```

• Over the integers

```
A = matrix(ZZ, entries)
A.echelon_form()
```

evaluate

• Over the rationals

```
A = matrix(QQ, entries)
A.echelon_form()
```

• With floating point double-precision reals

```
A = matrix(RDF, entries)
P, L, U = A.LU()
print L.round(5)
```

```
entries = [[1, -2, -4, 4], [1, -2, -3, 4], [0, -2, -6, 6], [-1, 2, 1, -4]]
```

• Over the integers

```
A = matrix(ZZ, entries)
A.echelon_form()
```

```
[ 1  0  0 -2]
[ 0  2  0 -6]
[ 0  0  1  0]
[ 0  0  0  0]
```

• Over the rationals

```
A = matrix(QQ, entries)
A.echelon_form()
```

[evaluate](#)

• With floating point double-precision reals

Over the integers

```
A = matrix(ZZ, entries)
A.echelon_form()
```

```
[ 1  0  0 -2]
[ 0  2  0 -6]
[ 0  0  1  0]
[ 0  0  0  0]
```

Over the rationals

```
A = matrix(QQ, entries)
A.echelon_form()
```

[evaluate](#)

With floating point double-precision reals

```
A = matrix(RDF, entries)
P, L, U = A.LU()
print L.round(5)
print
print U.round(5)
```


Over the integers

```
A = matrix(ZZ, entries)
A.echelon_form()
```

```
[ 1  0  0 -2]
[ 0  2  0 -6]
[ 0  0  1  0]
[ 0  0  0  0]
```

Over the rationals

```
A = matrix(QQ, entries)
A.echelon_form()
```

```
[ 1  0  0 -2]
[ 0  1  0 -3]
[ 0  0  1  0]
[ 0  0  0  0]
```

With floating point double-precision reals

```
A = matrix(RDF, entries)
P, L, U = A.LU()
print L.round(5)
```

```
A = matrix(QQ, entries)
A.echelon_form()
```

```
[ 1  0  0 -3]
```

• Over the rationals

```
A = matrix(QQ, entries)
A.echelon_form()
```

```
[ 1  0  0 -2]
[ 0  1  0 -3]
[ 0  0  1  0]
[ 0  0  0  0]
```

• With floating point double-precision reals

```
A = matrix(RDF, entries)
P, L, U = A.LU()
print L.round(5)
print
print U.round(5)
```

[evaluate](#)

jsMath

```
[ 0 0 1 0]
[ 0 0 0 0]
```

- With floating point double-precision reals

```
A = matrix(RDF, entries)
P, L, U = A.LU()
print L.round(5)
print
print U.round(5)
```

[evaluate](#)

- With elements of a finite field

```
F.<a> = FiniteField(3^2)
F.list()
```

```
A = random_matrix(F, 4, 4)
A
```



```
[ 0 0 1 0]
[ 0 0 0 0]
```

• With floating point double-precision reals

```
A = matrix(RDF, entries)
P, L, U = A.LU()
print L.round(5)
print
print U.round(5)
```

```
[ 1.0 0.0 0.0 0.0]
[ 0.0 1.0 0.0 0.0]
[ -1.0 -0.0 1.0 0.0]
[ 1.0 -0.0 -0.33333 1.0]
```

```
[ 1.0 -2.0 -4.0 4.0]
[ 0.0 -2.0 -6.0 6.0]
[ 0.0 0.0 -3.0 -0.0]
[ 0.0 0.0 0.0 -0.0]
```

• With elements of a finite field

```
F.<a> = FiniteField(3^2)
F.list()
```

```
[ 1.0 0.0 0.0 0.0]
[ 0.0 1.0 0.0 0.0]
[ -1.0 -0.0 1.0 0.0]
[ 1.0 -0.0 -0.33333 1.0]
```

```
[ 1.0 -2.0 -4.0 4.0]
[ 0.0 -2.0 -6.0 6.0]
[ 0.0 0.0 -3.0 -0.0]
[ 0.0 0.0 0.0 -0.0]
```

• With elements of a finite field

```
F.<a> = FiniteField(3^2)
F.list()
```

[evaluate](#)

```
A = random_matrix(F, 4, 4)
A
```

```
A.characteristic_polynomial('T')
```

```
[ 1.0 0.0 0.0 0.0]
[ 0.0 1.0 0.0 0.0]
[ -1.0 -0.0 1.0 0.0]
[ 1.0 -0.0 -0.33333 1.0]
```

```
[ 1.0 -2.0 -4.0 4.0]
[ 0.0 -2.0 -6.0 6.0]
[ 0.0 0.0 -3.0 -0.0]
[ 0.0 0.0 0.0 -0.0]
```

• With elements of a finite field

```
F.<a> = FiniteField(3^2)
F.list()
```

```
[0, 2*a, a + 1, a + 2, 2, a, 2*a + 2, 2*a + 1, 1]
```

```
A = random_matrix(F, 4, 4)
A
```

[evaluate](#)

```
A.characteristic_polynomial('T')
```


[1.0 -0.0 -0.33333 1.0]

[1.0 -2.0 -4.0 4.0]

[0.0 -2.0 -6.0 6.0]

[0.0 0.0 -3.0 -0.0]

[0.0 0.0 0.0 -0.0]

• With elements of a finite field

F.<a> = FiniteField(3^2)

F.list()

[0, 2*a, a + 1, a + 2, 2, a, 2*a + 2, 2*a + 1, 1]

A = random_matrix(F, 4, 4)

A

[evaluate](#)

A.characteristic_polynomial('T')

8 All-In-One

• Seamlessly move from one area of mathematics to another

```
[ 1.0 -0.0 -0.33333 1.0]
```

```
[ 1.0 -2.0 -4.0 4.0]
```

```
[ 0.0 -2.0 -6.0 6.0]
```

```
[ 0.0 0.0 -3.0 -0.0]
```

```
[ 0.0 0.0 0.0 -0.0]
```

• With elements of a finite field

```
F.<a> = FiniteField(3^2)
```

```
F.list()
```

```
[0, 2*a, a + 1, a + 2, 2, a, 2*a + 2, 2*a + 1, 1]
```

```
A = random_matrix(F, 4, 4)
```

```
A
```

[evaluate](#)

```
A.characteristic_polynomial('T')
```

8 All-In-One

• Seamlessly move from one area of mathematics to another

Pirsa: 11050014

Page 234/394

Find: def jordan

Previous

Next

Highlight all

Match case

```
[ 1.0 -0.0 -0.33333 1.0]
```

```
[ 1.0 -2.0 -4.0 4.0]
```

```
[ 0.0 -2.0 -6.0 6.0]
```

```
[ 0.0 0.0 -3.0 -0.0]
```

```
[ 0.0 0.0 0.0 -0.0]
```

• With elements of a finite field

```
F.<a> = FiniteField(3^2)
```

```
F.list()
```

```
[0, 2*a, a + 1, a + 2, 2, a, 2*a + 2, 2*a + 1, 1]
```

```
A = random_matrix(F, 4, 4)
```

```
A
```

```
[ a + 2 2*a 2*a 2*a + 1]
```

```
[ 1 1 a a + 1]
```

```
[ a + 1 a + 2 a 0]
```

```
[ a + 2 1 1 a + 2]
```

```
A.characteristic_polynomial('T')
```

[evaluate](#)


```
F.<a> = FiniteField(3^2)
F.list()
```

```
[0, 2*a, a + 1, a + 2, 2, a, 2*a + 2, 2*a + 1, 1]
```

```
A = random_matrix(F, 4, 4)
A
```

```
[ a + 2      2*a      2*a 2*a + 1]
[      1      1      a  a + 1]
[ a + 1  a + 2      a      0]
[ a + 2      1      1  a + 2]
```

```
A.characteristic_polynomial('T')
```

evaluate

8 All-In-One

- Seamlessly move from one area of mathematics to another

```
G = graphs.CirculantGraph(12, [1, 5])
G.plot()
```

```
F.<a> = FiniteField(3^2)
F.list()
```

```
[0, 2*a, a + 1, a + 2, 2, a, 2*a + 2, 2*a + 1, 1]
```

```
A = random_matrix(F, 4, 4)
A
```

```
[ a + 2  2*a  2*a 2*a + 1]
[      1      1      a  a + 1]
[ a + 1  a + 2      a      0]
[ a + 2      1      1  a + 2]
```

```
A.characteristic_polynomial('T')
```

```
T^4 + T^3 + 2
```

8 All-In-One

- Seamlessly move from one area of mathematics to another

```
G = graphs.CirculantGraph(12, [1, 5])
G.plot()
```

evaluate

```
F.<a> = FiniteField(3^2)
F.list()
```

```
[0, 2*a, a + 1, a + 2, 2, a, 2*a + 2, 2*a + 1, 1]
```

```
A = random_matrix(F, 4, 4)
```

evaluate

```
[ a + 2      2*a      2*a 2*a + 1]
[      1      1      a  a + 1]
[ a + 1  a + 2      a      0]
[ a + 2      1      1  a + 2]
```

```
A.characteristic_polynomial('T')
```

```
T^4 + T^3 + 2
```

8 All-In-One

- Seamlessly move from one area of mathematics to another

```
G = graphs.CirculantGraph(12, [1, 5])
G.plot()
```



```
F.<a> = FiniteField(3^2)
F.list()
```

```
[0, 2*a, a + 1, a + 2, 2, a, 2*a + 2, 2*a + 1, 1]
```

```
A = random_matrix(F, 4, 4)
A
```

```
[2*a + 1 2*a + 1      a      2]
[ a + 1      0      a + 2      2]
[      0      a      2*a      0]
[      2*a      a + 1      a      a]
```

```
A.characteristic_polynomial('T')
```

evaluate

```
T^4 + T^3 + 2
```

8 All-In-One

- Seamlessly move from one area of mathematics to another

```
G = graphs.CirculantGraph(12, [1, 5])
G.plot()
```

```
F.<a> = FiniteField(3^2)
F.list()
```

```
[0, 2*a, a + 1, a + 2, 2, a, 2*a + 2, 2*a + 1, 1]
```

```
A = random_matrix(F, 4, 4)
A
```

```
[2*a + 1 2*a + 1      a      2]
[ a + 1      0      a + 2      2]
[      0      a      2*a      0]
[      2*a      a + 1      a      a]
```

```
A.characteristic_polynomial('T')
```

```
T^4 + (a + 2)*T^3 + 2*T^2 + 2*a*T + a + 2
```

8 All-In-One

- Seamlessly move from one area of mathematics to another

```
G = graphs.CirculantGraph(12, [1, 5])
G.plot()
```

[evaluate](#)

A.characteristic_polynomial('T')

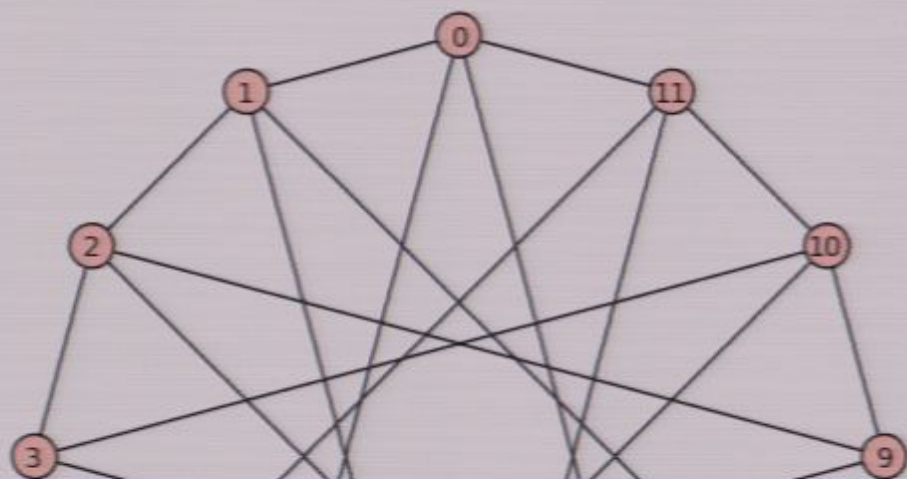
$$T^4 + (a + 2)T^3 + 2T^2 + 2aT + a + 2$$

8 All-In-One

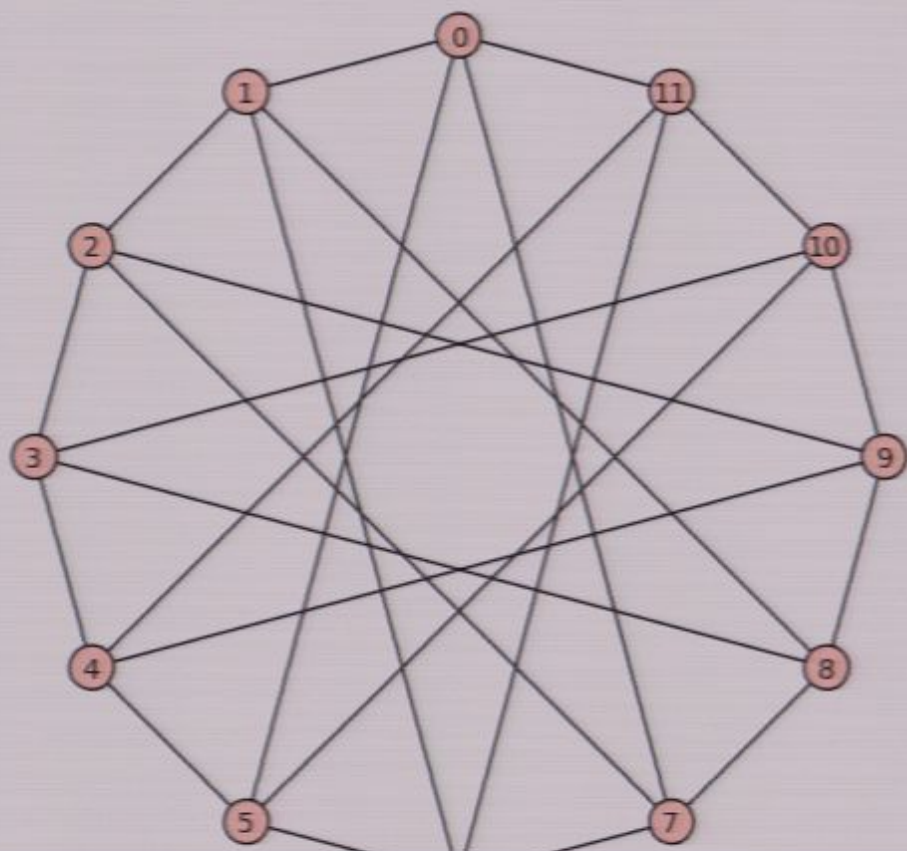
- Seamlessly move from one area of mathematics to another

```
G = graphs.CirculantGraph(12, [1, 5])
G.plot()
```

[evaluate](#)




```
G = graphs.CirculantGraph(12, [1, 5])  
G.plot()
```





G.adjacency_matrix()

[evaluate](#)

G.spectrum()

A = G.automorphism_group(); A

A.order()

9 Versatile

- Drill down into included libraries



G.adjacency_matrix()

```

[0 1 0 0 0 1 0 1 0 0 0 1]
[1 0 1 0 0 0 1 0 1 0 0 0]
[0 1 0 1 0 0 0 1 0 1 0 0]
[0 0 1 0 1 0 0 0 1 0 1 0]
[0 0 0 1 0 1 0 0 0 1 0 1]
[1 0 0 0 1 0 1 0 0 0 1 0]
[0 1 0 0 0 1 0 1 0 0 0 1]
[1 0 1 0 0 0 1 0 1 0 0 0]
[0 1 0 1 0 0 0 1 0 1 0 0]
[0 0 1 0 1 0 0 0 1 0 1 0]
[0 0 0 1 0 1 0 0 0 1 0 1]
[1 0 0 0 1 0 1 0 0 0 1 0]

```

G.spectrum()

[evaluate](#)

A = G.automorphism_group(); A


```
[0 1 0 0 0 1 0 1 0 0 0 1]
[1 0 1 0 0 0 1 0 1 0 0 0]
[0 1 0 1 0 0 0 1 0 1 0 0]
[0 0 1 0 1 0 0 0 1 0 1 0]
[0 0 0 1 0 1 0 0 0 1 0 1]
[1 0 0 0 1 0 1 0 0 0 1 0]
[0 1 0 0 0 1 0 1 0 0 0 1]
[1 0 1 0 0 0 1 0 1 0 0 0]
[0 1 0 1 0 0 0 1 0 1 0 0]
[0 0 1 0 1 0 0 0 1 0 1 0]
[0 0 0 1 0 1 0 0 0 1 0 1]
[1 0 0 0 1 0 1 0 0 0 1 0]
```

G.spectrum()

evaluate

A = G.automorphism_group(); A

A.order()

9 Versatile

```
[0 1 0 0 0 1 0 1 0 0 0 1]
[1 0 1 0 0 0 1 0 1 0 0 0]
[0 1 0 1 0 0 0 1 0 1 0 0]
[0 0 1 0 1 0 0 0 1 0 1 0]
[0 0 0 1 0 1 0 0 0 1 0 1]
[1 0 0 0 1 0 1 0 0 0 1 0]
[0 1 0 0 0 1 0 1 0 0 0 1]
[1 0 1 0 0 0 1 0 1 0 0 0]
[0 1 0 1 0 0 0 1 0 1 0 0]
[0 0 1 0 1 0 0 0 1 0 1 0]
[0 0 0 1 0 1 0 0 0 1 0 1]
[1 0 0 0 1 0 1 0 0 0 1 0]
```

G.spectrum()

```
[4, 2, 2, 0, 0, 0, 0, 0, 0, -2, -2, -4]
```

A = G.automorphism_group(); A

[evaluate](#)

A.order()

9 Versatile

```
[0 1 0 0 0 1 0 1 0 0 0 1]
[1 0 1 0 0 0 1 0 1 0 0 0]
[0 1 0 1 0 0 0 1 0 1 0 0]
[0 0 1 0 1 0 0 0 1 0 1 0]
[0 0 0 1 0 1 0 0 0 1 0 1]
[1 0 0 0 1 0 1 0 0 0 1 0]
[0 1 0 0 0 1 0 1 0 0 0 1]
[1 0 1 0 0 0 1 0 1 0 0 0]
[0 1 0 1 0 0 0 1 0 1 0 0]
[0 0 1 0 1 0 0 0 1 0 1 0]
[0 0 0 1 0 1 0 0 0 1 0 1]
[1 0 0 0 1 0 1 0 0 0 1 0]
```

G.spectrum()

```
[4, 2, 2, 0, 0, 0, 0, 0, 0, -2, -2, -4]
```

A = G.automorphism_group(); A

evaluate

A.order()

9 Versatile


```
[0 1 0 0 0 1 0 1 0 0 0 1]
[1 0 1 0 0 0 1 0 1 0 0 0]
[0 1 0 1 0 0 0 1 0 1 0 0]
[0 0 1 0 1 0 0 0 1 0 1 0]
[0 0 0 1 0 1 0 0 0 1 0 1]
[1 0 0 0 1 0 1 0 0 0 1 0]
[0 1 0 0 0 1 0 1 0 0 0 1]
[1 0 1 0 0 0 1 0 1 0 0 0]
[0 1 0 1 0 0 0 1 0 1 0 0]
[0 0 1 0 1 0 0 0 1 0 1 0]
[0 0 0 1 0 1 0 0 0 1 0 1]
[1 0 0 0 1 0 1 0 0 0 1 0]
```

G.spectrum()

```
[4, 2, 2, 0, 0, 0, 0, 0, 0, -2, -2, -4]
```

A = G.automorphism_group(); A

```
Permutation Group with generators [(5,11), (4,10), (3,9),
(1,2,3,4,5,12)(6,7,8,9,10,11), (1,5)(2,4)(7,11)(8,10), (1,7)]
```

A.order()

[evaluate](#)

9 Versatile

Pirsa: 11050014

Page 248/394

Find: def jordan

Previous Next Highlight all Match case

```
[0 0 1 0 1 0 0 1 0 1 0]
[0 0 0 1 0 1 0 0 0 1 0]
[1 0 0 0 1 0 1 0 0 0 1]
```

G.spectrum()

```
[4, 2, 2, 0, 0, 0, 0, 0, -2, -2, -4]
```

A = G.automorphism_group(); A

```
Permutation Group with generators [(5,11), (4,10), (3,9),
(1,2,3,4,5,12)(6,7,8,9,10,11), (1,5)(2,4)(7,11)(8,10), (1,7)]
```

A.order()

```
768
```

9 Versatile

- Drill down into included libraries
- SciPy Python bindings to FORTRAN LAPACK
- Example by Jason Grout

```
A = matrix(RDF, [[10,2,3],[3,4,6],[2,8,10]])
```

evaluate

Pirsa: 11050014

Page 249/394

Find: def jordan

Previous Next Highlight all Match case

9 Versatile

- Drill down into included libraries
- SciPy Python bindings to FORTRAN LAPACK
- Example by Jason Grout

```
A = matrix(RDF, [[10,2,3],[3,4,6],[2,8,10]])
```

[evaluate](#)



```
import scipy
import scipy.linalg
lu, piv, success = scipy.linalg.flapack.dgetrf(A.rows())
```

```
# Break apart combined L and U
rows, cols = lu.shape
L = copy(matrix(RDF, rows, cols, 1))
U = matrix(RDF, rows, cols)
for i in range(rows):
    for j in range(cols):
        if i>j:
```


• SCIPY Python bindings to FORTRAN LAPACK

• Example by Jason Grout

```
A = matrix(RDF, [[10,2,3],[3,4,6],[2,8,10]])
```

```
import scipy
import scipy.linalg
lu, piv, success = scipy.linalg.flapack.dgetrf(A.rows())
```

[evaluate](#)



```
# Break apart combined L and U
rows, cols = lu.shape
L = copy(matrix(RDF, rows, cols, 1))
U = matrix(RDF, rows, cols)
for i in range(rows):
    for j in range(cols):
        if i>j:
            L[i,j]=lu[i,j]
        else:
            U[i,j]=lu[i,j]
```

```
# Construct permutation matrix from permutation
P = copy(identity matrix(rows))
```

• SCIPY Python bindings to FORTRAN LAPACK

• Example by Jason Grout

```
A = matrix(RDF, [[10,2,3],[3,4,6],[2,8,10]])
```

```
import scipy
import scipy.linalg
lu, piv, success = scipy.linalg.flapack.dgetrf(A.rows())
```

[evaluate](#)



```
# Break apart combined L and U
rows, cols = lu.shape
L = copy(matrix(RDF, rows, cols, 1))
U = matrix(RDF, rows, cols)
for i in range(rows):
    for j in range(cols):
        if i > j:
            L[i,j] = lu[i,j]
        else:
            U[i,j] = lu[i,j]
```

```
# Construct permutation matrix from permutation
P = copy(identity matrix(rows))
```

• SCIPY Python bindings to FORTRAN LAPACK

• Example by Jason Grout

```
A = matrix(RDF, [[10,2,3],[3,4,6],[2,8,10]])
```

```
import scipy
import scipy.linalg
lu, piv, success = scipy.linalg.flapack.dgetrf(A.rows())
```

```
# Break apart combined L and U
rows, cols = lu.shape
L = copy(matrix(RDF, rows, cols, 1))
U = matrix(RDF, rows, cols)
for i in range(rows):
    for j in range(cols):
        if i > j:
            L[i,j] = lu[i,j]
        else:
            U[i,j] = lu[i,j]

# Construct permutation matrix from permutation
P = copy(identity_matrix(rows))
for i,j in enumerate(piv):
```



```
import scipy
import scipy.linalg
lu, piv, success = scipy.linalg.flapack.dgetrf(A.rows())
```

```
# Break apart combined L and U
rows, cols = lu.shape
L = copy(matrix(RDF, rows, cols, 1))
U = matrix(RDF, rows, cols)
for i in range(rows):
    for j in range(cols):
        if i > j:
            L[i, j] = lu[i, j]
        else:
            U[i, j] = lu[i, j]

# Construct permutation matrix from permutation
P = copy(identity_matrix(rows))
for i, j in enumerate(piv):
    P.swap_rows(i, j)

print "\nP: \n", P
print "\nL: \n", L
print "\nU: \n", U
print "\nVerification: \n", P*A - L*U
```

```
# Break apart combined L and U
rows, cols = lu.shape
L = copy(matrix(RDF, rows, cols, 1))
U = matrix(RDF, rows, cols)
for i in range(rows):
    for j in range(cols):
        if i>j:
            L[i,j]=lu[i,j]
        else:
            U[i,j]=lu[i,j]

# Construct permutation matrix from permutation
P = copy(identity_matrix(rows))
for i,j in enumerate(piv):
    P.swap_rows(i,j)

print "\nP: \n", P
print "\nL: \n", L
print "\nU: \n", U
print "\nVerification: \n", P*A - L*U
```

evaluate

10 Convenient

$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

L:

$\begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.2 & 1.0 & 0.0 \\ 0.3 & 0.447368421053 & 1.0 \end{bmatrix}$

U:

$\begin{bmatrix} 10.0 & 2.0 & 3.0 \\ 0.0 & 7.6 & 9.4 \\ 0.0 & 0.0 & 0.894736842105 \end{bmatrix}$

Verification:

$\begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ -4.4408920985e-16 & 4.4408920985e-16 & 0.0 \end{bmatrix}$

10 Convenient

- Install on any computer (don't need to be root)
- Access from any web browser, including smart phones
- Command line or notebook
- Batch or interactive

P.swap_rows(i,j)

```
print "\nP: \n", P
print "\nL: \n", L
print "\nU: \n", U
print "\nVerification: \n", P*A - L*U
```

```
P:
[1 0 0]
[0 0 1]
[0 1 0]
```

```
L:
[ 1.0 0.0 0.0]
[ 0.2 1.0 0.0]
[ 0.3 0.447368421053 1.0]
```

```
U:
[ 10.0 2.0 3.0]
[ 0.0 7.6 9.4]
[ 0.0 0.0 0.894736842105]
```

```
Verification:
[ 0.0 0.0 0.0]
[ 0.0 0.0 0.0]
[-4.4408920985e-16 4.4408920985e-16 0.0]
```

```
print "\nL: \n", L
print "\nU: \n", U
print "\nVerification: \n", P*A - L*U
```

```
P:
[1 0 0]
[0 0 1]
[0 1 0]
```

```
L:
[ 1.0 0.0 0.0]
[ 0.2 1.0 0.0]
[ 0.3 0.447368421053 1.0]
```

```
U:
[ 10.0 2.0 3.0]
[ 0.0 7.6 9.4]
[ 0.0 0.0 0.894736842105]
```

```
Verification:
[ 0.0 0.0 0.0]
[ 0.0 0.0 0.0]
[-4.4408920985e-16 4.4408920985e-16 0.0]
```

10 Convenient

- Install on any computer (don't need to be root)

Verification:

```
[ 0.0 0.0 0.0]
[ 0.0 0.0 0.0]
[-4.4408920985e-16 4.4408920985e-16 0.0]
```

10 Convenient

- Install on any computer (don't need to be root)
- Access from any web browser, including smart phones
- Command line or notebook
- Batch or interactive
- Dedicated applications, such as Android application
- Novel front-ends - one-shot cell phone at http://math3.skku.ac.kr/wap_html
- Some folks install Sage *only* to get GAP, R, PARI, SciPy, etc.

11 The Conifold: from "Toric Geometry and Sage"

Arnold Sommerfeld Center for Theoretical Physics, Munich, April 26-29, 2011

By Volker Braun, Dublin Institute for Advanced Studies

- Novel front-ends - one-shot cell phone at http://math3.skku.ac.kr/wap_html
- Some folks install Sage *only* to get GAP, R, PARI, SciPy, etc.

11 The Conifold: from "Toric Geometry and Sage"

Arnold Sommerfeld Center for Theoretical Physics, Munich, April 26-29, 2011

By Volker Braun, Dublin Institute for Advanced Studies

A singularity that occurs very often is the conifold, which is the simplest non-quotient singularity. In terms of toric geometry, it is defined by the non-simplicial cone over a minimal lattice square at distance 1:

```
# do not evaluate, depends on unofficial code
```

[evaluate](#)

```
conifold = toric_varieties.Conifold()  
conifold.is_smooth()
```

False

```
conifold.is_orbifold()
```

False

Some folks install Sage *only* to get GAP, R, PARI, SciPy, etc.

11 The Conifold: from "Toric Geometry and Sage"

Arnold Sommerfeld Center for Theoretical Physics, Munich, April 26-29, 2011

By Volker Braun, Dublin Institute for Advanced Studies

A singularity that occurs very often is the conifold, which is the simplest non-quotient singularity. In terms of toric geometry, it is defined by the non-simplicial cone over a minimal lattice square at distance 1:

```
# do not evaluate, depends on unofficial code
```

[evaluate](#)

```
conifold = toric_varieties.Conifold()  
conifold.is_smooth()
```

False

```
conifold.is_orbifold()
```

False

```
square_cone = conifold.fan().generating_cone(0)  
square_cone.rays()
```


evaluate

```
conifold = toric_varieties.Conifold()
conifold.is_smooth()
```

False

```
conifold.is_orbifold()
```

False

```
square_cone = conifold.fan().generating_cone(0)
square_cone.rays()
```

(N(0, 0, 1), N(0, 1, 1), N(1, 0, 1), N(1, 1, 1))

```
patch = conifold.affine_algebraic_patch(square_cone)
patch
```

Closed subscheme of Affine Space of dimension 4 over Rational Field
defined by:
 $z_0 z_2 - z_1 z_3$

The conifold is not smooth because the hypersurface equation is not transverse at $\tilde{x} = (0, 0, 0, 0)$. More precisely, jsMath


```
conifold = tonic_varieties.Conifold()
conifold.is_smooth()
```

evaluate

False

```
conifold.is_orbifold()
```

False

```
square_cone = conifold.fan().generating_cone(0)
square_cone.rays()
```

(N(0, 0, 1), N(0, 1, 1), N(1, 0, 1), N(1, 1, 1))

```
patch = conifold.affine_algebraic_patch(square_cone)
patch
```

Closed subscheme of Affine Space of dimension 4 over Rational Field
defined by:
 $z_0z_2 - z_1z_3$

The conifold is not smooth because the hypersurface equation is not transverse at $\tilde{x} = (0, 0, 0, 0)$. More precisely, jsMath

```
conifold = toric_varieties.Conifold()
conifold.is_smooth()
```

evaluate

False

```
conifold.is_orbifold()
```

False

```
square_cone = conifold.fan().generating_cone(0)
square_cone.rays()
```

$(N(0, 0, 1), N(0, 1, 1), N(1, 0, 1), N(1, 1, 1))$

```
patch = conifold.affine_algebraic_patch(square_cone)
patch
```

Closed subscheme of Affine Space of dimension 4 over Rational Field
defined by:
 $z_0z_2 - z_1z_3$

The conifold is not smooth because the hypersurface equation is not transverse at $\tilde{x} = (0, 0, 0, 0)$. More precisely, jsMath

```
conifold = toric_varieties.Conifold()
conifold.is_smooth()
```

evaluate

False

```
conifold.is_orbifold()
```

False

```
square_cone = conifold.fan().generating_cone(0)
square_cone.rays()
```

$(N(0, 0, 1), N(0, 1, 1), N(1, 0, 1), N(1, 1, 1))$

```
patch = conifold.affine_algebraic_patch(square_cone)
patch
```

Closed subscheme of Affine Space of dimension 4 over Rational Field
defined by:
 $z_0*z_2 - z_1*z_3$

The conifold is not smooth because the hypersurface equation is not transverse at $\tilde{x} = (0, 0, 0, 0)$. More precisely, jsMath


```
patch = conifold.affine_algebraic_patch(square_cone)
patch
```

Closed subscheme of Affine Space of dimension 4 over Rational Field
defined by:
 $z_0 \cdot z_2 - z_1 \cdot z_3$

The conifold is not smooth because the hypersurface equation is not transverse at $\vec{z} = (0, 0, 0, 0)$. More precisely, the singularities are the variety of the Jacobian ideal

$$Jac(f) = \left\langle f, \frac{\partial f}{\partial z_0}, \dots, \frac{\partial f}{\partial z_3} \right\rangle \quad (1)$$

For the conifold, it is

```
Jac = patch.Jacobian()
Jac
```

Ideal ($z_0 \cdot z_2 - z_1 \cdot z_3, z_2, -z_3, z_0, -z_1$) of Multivariate Polynomial Ring
in z_0, z_1, z_2, z_3 over Rational Field

The conifold is not smooth because the hypersurface equation is not transverse at $\bar{z} = (0, 0, 0, 0)$. More precisely, the singularities are the variety of the Jacobian ideal

$$Jac(f) = \left\langle f, \frac{\partial f}{\partial z_0}, \dots, \frac{\partial f}{\partial z_3} \right\rangle \quad (1)$$

For the conifold, it is

```
Jac = patch.Jacobian()
Jac
```

```
Ideal (z0*z2 - z1*z3, z2, -z3, z0, -z1) of Multivariate Polynomial Ring
in z0, z1, z2, z3 over Rational Field
```

```
A4 = patch.ambient_space()
origin = A4.subscheme(A4.gens()) # the origin (0,0,0,0)
A4.subscheme(Jac) == origin
```

```
True
```

The conifold is not smooth because the hypersurface equation is not transverse at $\bar{z} = (0, 0, 0, 0)$. More precisely, the singularities are the variety of the Jacobian ideal

$$Jac(f) = \left\langle f, \frac{\partial f}{\partial z_0}, \dots, \frac{\partial f}{\partial z_3} \right\rangle \quad (1)$$

For the conifold, it is

```
Jac = patch.Jacobian()
Jac
```

```
Ideal (z0*z2 - z1*z3, z2, -z3, z0, -z1) of Multivariate Polynomial Ring
in z0, z1, z2, z3 over Rational Field
```

```
A4 = patch.ambient_space()
origin = A4.subscheme(A4.gens()) # the origin (0,0,0,0)
A4.subscheme(Jac) == origin
```

```
True
```

The most basic invariant of an isolated hypersurface singularity is its *Milnor number*, which is the vector space dimension of $\mathbb{C}[\bar{x}]/Jac(f(\bar{x}))$. For the conifold, it is one:

For the conifold, it is

```
Jac = patch.Jacobian()
```

```
Jac
```

```
Ideal (z0*z2 - z1*z3, z2, -z3, z0, -z1) of Multivariate Polynomial Ring
in z0, z1, z2, z3 over Rational Field
```

```
A4 = patch.ambient_space()
```

```
origin = A4.subscheme(A4.gens()) # the origin (0,0,0,0)
```

```
A4.subscheme(Jac) == origin
```

```
True
```

The most basic invariant of an isolated hypersurface singularity is its *Milnor number*, which is the vector space dimension of $\mathbb{C}[\bar{x}]/Jac(f(\bar{x}))$. For the conifold, it is one:

```
Jac.vector_space_dimension()
```

```
1
```

In fact, the converse is also true: A 3-dimensional isolated singularity of Milnor number one is a conifold. Note, however, that higher Milnor numbers no longer uniquely determine the singularity.

in z_0, z_1, z_2, z_3 over Rational Field

```
A4 = patch.ambient_space()
origin = A4.subscheme(A4.gens()) # the origin (0,0,0,0)
A4.subscheme(Jac) == origin
```

True

The most basic invariant of an isolated hypersurface singularity is its *Milnor number*, which is the vector space dimension of $\mathbb{C}[\bar{x}]/\text{Jac}(f(\bar{x}))$. For the conifold, it is one:

```
Jac.vector_space_dimension()
```

1

In fact, the converse is also true: A 3-dimensional isolated singularity of Milnor number one is a conifold. Note, however, that higher Milnor numbers no longer uniquely determine the singularity.

Exercise 9. Use Sage to compute the Milnor number of the singularity $\mathbb{C}^2/\mathbb{Z}_n$ for $n \in \{2, 3, \dots, 10\}$.

12 Parallel Processing

Parallel "decorator" converts function to an iterator

• 2 cores, 4 threads

```
@parallel('multiprocessing', 4)
def summation(multiple):
    sum = 0
    for i in xrange(0, 1000000*multiple, multiple):
        sum = sum + i
    print('Finished {}'.format(multiple))
    return sum
```

• Now pass in a *list* of inputs to create the "jobs."

```
summer = summation([20, 5, 100, 57])
```

• Ask for a list of the jobs (inputs and outputs, in new order).

```
list(summer)
```


12 Parallel Processing

- Parallel "decorator" converts function to an iterator
- This example uses the Python multiprocessing library, can also fork, or...
- 2 cores, 4 threads

```
@parallel('multiprocessing', 4)
def summation(multiple):
    sum = 0
    for i in xrange(0, 1000000*multiple, multiple):
        sum = sum + i
    print('Finished {0}'.format(multiple))
    return sum
```

- Now pass in a *list* of inputs to create the "jobs."

```
summer = summation([20, 5, 100, 57])
```

- Ask for a list of the jobs (inputs and outputs in new order)

• 2 cores, 4 threads

```
@parallel('multiprocessing', 4)
def summation(multiple):
    sum = 0
    for i in xrange(0, 1000000*multiple, multiple):
        sum = sum + i
    print('Finished {}'.format(multiple))
    return sum
```

• Now pass in a *list* of inputs to create the "jobs."

```
summer = summation([20, 5, 100, 57])
```

[evaluate](#)

• Ask for a list of the jobs (inputs and outputs, in new order).

```
list(summer)
```

• 2 cores, 4 threads

```
@parallel('multiprocessing', 4)
def summation(multiple):
    sum = 0
    for i in xrange(0, 1000000*multiple, multiple):
        sum = sum + i
    print('Finished {}'.format(multiple))
    return sum
```

• Now pass in a *list* of inputs to create the "jobs."

```
summer = summation([20, 5, 100, 57])
```

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

```
...
NameError: name 'summation' is not defined
```

• Ask for a list of the jobs (inputs and outputs, in new order).

```
list(summer)
```

evaluate

• 2 cores, 4 threads

```
@parallel('multiprocessing', 4)
def summation(multiple):
    sum = 0
    for i in xrange(0, 1000000*multiple, multiple):
        sum = sum + i
    print('Finished {}'.format(multiple))
    return sum
```

• Now pass in a *list* of inputs to create the "jobs."

```
summer = summation([20, 5, 100, 57])
```

• Ask for a list of the jobs (inputs and outputs, in new order).

```
list(summer)
```

[evaluate](#)

- Now pass in a *list* of inputs to create the "jobs."

```
summer = summation([20, 5, 100, 57])
```

- Ask for a list of the jobs (inputs and outputs, in new order).

```
list(summer)
```

```
Finished 100
Finished 5
Finished 57
Finished 20
[[((100,), {}), 499999500000000), ((5,), {}), 2499997500000), ((57,), {}), 28499971500000), ((20,), {}), 9999990000000)]
```

- A larger list of inputs. Watch in htop.

```
long_summer = summation(range(100))
```

[evaluate](#)

```
list(long_summer)
```

- Now pass in a *list* of inputs to create the "jobs."

```
summer = summation([20, 5, 100, 57])
```

- Ask for a list of the jobs (inputs and outputs, in new order).

```
list(summer)
```

```
Finished 100
Finished 5
Finished 57
Finished 20
[[((100,), {}), 49999950000000), ((5,), {}), 2499997500000), ((57,), {}), 28499971500000), ((20,), {}), 9999990000000)]
```

- A larger list of inputs. Watch in htop.

```
long_summer = summation(range(100))
```

[evaluate](#)


```
Finished 100
Finished 5
Finished 57
Finished 20
[(((100,), {}), 49999950000000), (((5,), {}), 2499997500000), (((57,), {}), 28499971500000), (((20,), {}), 9999990000000)]
```

- A larger list of inputs. Watch in htop.

```
long_summer = summation(range(100))
```

[evaluate](#)

```
list(long_summer)
```

13 Cython

- Convert Python to compiled C
- Originally Pyrex, forked by Sage to Cython

```
def summation_python():
```

```
Finished 100
Finished 5
Finished 57
Finished 20
[[((100,), {}), 49999950000000), ((5,), {}), 2499997500000), ((57,), {}), 28499971500000), ((20,), {}), 99999900000000)]
```

• A larger list of inputs. Watch in htop.

```
long_summer = summation(range(100))
```

```
list(long_summer)
```

```
Finished 0
Finished 3
Finished 2
Finished 1
Finished 4
Finished 5
Finished 6
```

13 Cython

• Convert Python to compiled C

Pirsa: 11050014

Page 279/394

Find: def jordan

Previous Next Highlight all Match case

dev : bash

File Edit View Scrollback Bookmarks Settings Help

```
rob@tiger:/home/sage/abstract$ cd ..
```

```
rob@tiger:/home/sage$ cd dev
```

```
rob@tiger:/home/sage/dev$ ./sage
```

```
-----  
Sage Version 4.7.rc0, Release Date: 2011-04-22
```

```
Type notebook() for the GUI, and license() for information.  
-----
```

```
*****
```

```
Warning: this is a prerelease version, and it may be unstable.  
*****
```

```
sage: █
```

Google

abstract/deve... X + v

jsMath

rob : htop

File Edit View Scrollback Bookmarks Settings Help

```

1  [|||||] 100.0%
2  [|||||] 100.0%
3  [|||||] 99.4%
4  [|||||] 100.0%
Mem [|||||] 2100/3887MB
Swp [|||||] 0/2047MB

```

Tasks: 260 total, 5 running
Load average: 1.29 0.32 0.10
Uptime: 01:23:47

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
6702	rob	20	0	1147M	240M	2432	R	97.0	6.2	0:05.36	/sage/abstract/loc
6705	rob	20	0	1147M	240M	2432	R	95.0	6.2	0:05.33	/sage/abstract/loc
6703	rob	20	0	1147M	240M	2432	R	94.0	6.2	0:05.00	/sage/abstract/loc
6704	rob	20	0	1147M	240M	2432	R	81.0	6.2	0:05.21	/sage/abstract/loc
4523	root	20	0	112M	31320	11044	S	16.0	0.8	0:45.78	/usr/bin/X :0 -br
6321	rob	20	0	906M	158M	41080	S	4.0	4.1	1:07.68	/usr/lib/firefox-3
6291	rob	20	0	288M	25128	15308	S	3.0	0.6	0:05.92	konsole
6178	rob	20	0	239M	14024	10748	S	1.0	0.4	0:01.61	/usr/lib/gnome-pan
6133	rob	20	0	184M	12200	9272	S	1.0	0.3	0:02.53	metacity --replace
6499	rob	20	0	716M	99M	23088	S	1.0	2.6	0:07.12	python /sage/abstr
6310	rob	20	0	19788	1596	1108	R	1.0	0.0	0:41.45	htop
6152	rob	20	0	253M	18500	12896	S	0.0	0.5	0:01.02	gnome-panel
6553	rob	20	0	1147M	272M	35864	S	0.0	7.0	0:31.03	/sage/abstract/loc
4540	root	20	0	11280	632	492	S	0.0	0.0	0:00.26	/usr/sbin/inrqbala
4375	root	20	0	85652	4428	3428	S	0.0	0.1	0:00.85	NetworkManager
6390	rob	20	0	906M	158M	41080	S	0.0	4.1	0:03.06	/usr/lib/firefox-3
4816	root	20	0	57112	6148	2504	S	0.0	0.2	0:01.51	/usr/lib/upower/up
6120	rob	20	0	46196	5748	2388	S	0.0	0.1	0:00.61	/usr/lib/libgconf2
6059	rob	20	0	43120	31560	1512	S	0.0	0.8	0:00.23	/sage/abstract/loc
4306	messageb	20	0	24412	1976	776	S	0.0	0.0	0:01.48	dbus-daemon --syst
6326	rob	20	0	906M	158M	41080	S	0.0	4.1	0:01.27	/usr/lib/firefox-3
6333	rob	20	0	906M	158M	41080	S	0.0	4.1	0:01.97	/usr/lib/firefox-3
6589	rob	20	0	1435M	471M	12692	S	0.0	12.1	0:04.67	/usr/lib/jvm/java-

Help F2Setup F3Search F4Invert F5Tree F6SortBy F7Nice F8Nice +F9Kill F10Quit

Google

abstract/deve... X

jsMath

```

1  [|||||] 100.0%
2  [|||||] 100.0%
3  [|||||] 100.0%
4  [|||||] 100.0%
Mem [|||||] 2100/3887MB
Swp [|||||] 0/2047MB

```

Tasks: 260 total, 5 running
Load average: 1.29 0.32 0.10
Uptime: 01:23:48

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
6705	rob	20	0	1147M	240M	2432	R	98.0	6.2	0:06.83	/sage/abstract/loc
6703	rob	20	0	1147M	240M	2432	R	97.0	6.2	0:06.50	/sage/abstract/loc
6702	rob	20	0	1147M	240M	2432	R	95.0	6.2	0:06.83	/sage/abstract/loc
6704	rob	20	0	1147M	240M	2432	R	94.0	6.2	0:06.67	/sage/abstract/loc
4523	root	20	0	112M	31320	11044	S	5.0	0.8	0:45.85	/usr/bin/X :0 -br
6321	rob	20	0	906M	158M	41080	S	3.0	4.1	1:07.73	/usr/lib/firefox-3
6133	rob	20	0	184M	12200	9272	S	1.0	0.3	0:02.56	metacity --replace
6291	rob	20	0	288M	25128	15308	S	0.0	0.6	0:05.92	konsole
6178	rob	20	0	239M	14024	10748	S	0.0	0.4	0:01.62	/usr/lib/gnome-pan
6499	rob	20	0	716M	99M	23088	S	0.0	2.6	0:07.12	python /sage/abstr
6310	rob	20	0	19788	1596	1108	R	0.0	0.0	0:41.46	htop
6152	rob	20	0	253M	18500	12896	S	0.0	0.5	0:01.02	gnome-panel
6553	rob	20	0	1147M	272M	35864	S	0.0	7.0	0:31.03	/sage/abstract/loc
4540	root	20	0	11280	632	492	S	0.0	0.0	0:00.26	/usr/sbin/inrqbala
4375	root	20	0	85652	4428	3428	S	0.0	0.1	0:00.85	NetworkManager
6390	rob	20	0	906M	158M	41080	S	0.0	4.1	0:03.06	/usr/lib/firefox-3
4816	root	20	0	57112	6148	2504	S	0.0	0.2	0:01.51	/usr/lib/upower/up
6120	rob	20	0	46196	5748	2388	S	0.0	0.1	0:00.62	/usr/lib/libgconf2
6059	rob	20	0	43120	31560	1512	S	0.0	0.8	0:00.23	/sage/abstract/loc
4306	messageb	20	0	24412	1976	776	S	0.0	0.0	0:01.48	dbus-daemon --syst
6326	rob	20	0	906M	158M	41080	S	0.0	4.1	0:01.27	/usr/lib/firefox-3
6333	rob	20	0	906M	158M	41080	S	0.0	4.1	0:01.97	/usr/lib/firefox-3
6588	rob	20	0	1435M	471M	12692	S	0.0	12.1	0:04.67	/usr/lib/jvm/java-

Google

abstract/deve... X

jsMath

rob : htop

File Edit View Scrollback Bookmarks Settings Help

```

1  [|||||] 100.0%
2  [|||||] 100.0%
3  [|||||] 100.0%
4  [|||||] 100.0%
Mem[|||||] 2100/3887MB
Swp[|||||] 0/2047MB

```

Tasks: 260 total, 5 running
Load average: 1.29 0.32 0.10
Uptime: 01:23:50

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
6705	rob	20	0	1147M	240M	2432	R	95.0	6.2	0:08.30	/sage/abstract/loc
6704	rob	20	0	1147M	240M	2432	R	95.0	6.2	0:08.13	/sage/abstract/loc
6702	rob	20	0	1147M	240M	2432	R	93.0	6.2	0:08.25	/sage/abstract/loc
6703	rob	20	0	1147M	240M	2432	R	92.0	6.2	0:07.92	/sage/abstract/loc
6390	rob	20	0	906M	158M	41080	S	7.0	4.1	0:03.16	/usr/lib/firefox-3
4523	root	20	0	112M	31320	11044	S	5.0	0.8	0:45.93	/usr/bin/X :0 -br
6321	rob	20	0	906M	158M	41080	S	3.0	4.1	1:07.78	/usr/lib/firefox-3
6310	rob	20	0	19788	1596	1108	R	1.0	0.0	0:41.48	htop
6133	rob	20	0	184M	12200	9272	S	0.0	0.3	0:02.56	metacity --replace
6291	rob	20	0	288M	25128	15308	S	0.0	0.6	0:05.92	konsole
6178	rob	20	0	239M	14024	10748	S	0.0	0.4	0:01.63	/usr/lib/gnome-pan
6499	rob	20	0	716M	99M	23088	S	0.0	2.6	0:07.14	python /sage/abstr
6152	rob	20	0	253M	18500	12896	S	0.0	0.5	0:01.03	gnome-panel
6553	rob	20	0	1147M	272M	35864	S	0.0	7.0	0:31.03	/sage/abstract/loc
4540	root	20	0	11280	632	492	S	0.0	0.0	0:00.26	/usr/sbin/irqbalan
4375	root	20	0	85652	4428	3428	S	0.0	0.1	0:00.85	NetworkManager
4816	root	20	0	57112	6148	2504	S	0.0	0.2	0:01.51	/usr/lib/upower/up
6120	rob	20	0	46196	5748	2388	S	0.0	0.1	0:00.62	/usr/lib/libgconf2
6059	rob	20	0	43120	31560	1512	S	0.0	0.8	0:00.23	/sage/abstract/loc
4306	messageb	20	0	24412	1976	776	S	0.0	0.0	0:01.48	dbus-daemon --syst
6326	rob	20	0	906M	158M	41080	S	0.0	4.1	0:01.28	/usr/lib/firefox-3
6333	rob	20	0	906M	158M	41080	S	0.0	4.1	0:01.97	/usr/lib/firefox-3
6588	rob	20	0	1435M	471M	12692	S	0.0	12.1	0:04.67	/usr/lib/jvm/java-


```

1  [|||||] 100.0%
2  [|||||] 100.0%
3  [|||||] 100.0%
4  [|||||] 100.0%
Mem [|||||] 2100/3887MB
Swp [|||||] 0/2047MB

```

Tasks: 260 total, 5 running
Load average: 1.29 0.32 0.10
Uptime: 01:23:51

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
6705	rob	20	0	1147M	240M	2432	R	98.0	6.2	0:09.80	/sage/abstract/loc
6702	rob	20	0	1147M	240M	2432	R	98.0	6.2	0:09.75	/sage/abstract/loc
6704	rob	20	0	1147M	240M	2432	R	91.0	6.2	0:09.52	/sage/abstract/loc
6703	rob	20	0	1147M	240M	2432	R	90.0	6.2	0:09.31	/sage/abstract/loc
4523	root	20	0	112M	31320	11044	S	7.0	0.8	0:46.05	/usr/bin/X :0 -br
6321	rob	20	0	906M	158M	41080	S	5.0	4.1	1:07.87	/usr/lib/firefox-3
6310	rob	20	0	19788	1596	1108	R	1.0	0.0	0:41.50	htop
6178	rob	20	0	239M	14024	10748	S	1.0	0.4	0:01.65	/usr/lib/gnome-pan
6499	rob	20	0	716M	99M	23088	S	1.0	2.6	0:07.16	python /sage/abstr
6390	rob	20	0	906M	158M	41080	S	0.0	4.1	0:03.16	/usr/lib/firefox-3
6133	rob	20	0	184M	12200	9272	S	0.0	0.3	0:02.58	metacity --replace
6291	rob	20	0	288M	25128	15308	S	0.0	0.6	0:05.92	konsole
6152	rob	20	0	253M	18500	12896	S	0.0	0.5	0:01.03	gnome-panel
6553	rob	20	0	1147M	272M	35864	S	0.0	7.0	0:31.03	/sage/abstract/loc
4540	root	20	0	11280	632	492	S	0.0	0.0	0:00.26	/usr/sbin/iqrbalan
4375	root	20	0	85652	4428	3428	S	0.0	0.1	0:00.85	NetworkManager
4816	root	20	0	57112	6148	2504	S	0.0	0.2	0:01.51	/usr/lib/upower/up
6120	rob	20	0	46196	5748	2388	S	0.0	0.1	0:00.62	/usr/lib/libgconf2
6059	rob	20	0	43120	31560	1512	S	0.0	0.8	0:00.23	/sage/abstract/loc
4306	messageb	20	0	24412	1976	776	S	0.0	0.0	0:01.48	dbus-daemon --syst
6326	rob	20	0	906M	158M	41080	S	0.0	4.1	0:01.29	/usr/lib/firefox-3
6333	rob	20	0	906M	158M	41080	S	0.0	4.1	0:01.97	/usr/lib/firefox-3
6589	rob	20	0	1435M	471M	12692	S	0.0	12.1	0:04.67	/usr/lib/jvm/java-

Google

abstract/deve... X

jsMath

rob : htop

File Edit View Scrollback Bookmarks Settings Help

```

1  [|||||] 100.0%
2  [|||||] 100.0%
3  [|||||] 100.0%
4  [|||||] 100.0%
Mem [|||||] 2101/3887MB
Swp [|||||] 0/2047MB
    
```

Tasks: 260 total, 5 running
Load average: 1.51 0.38 0.12
Uptime: 01:23:53

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
6703	rob	20	0	1147M	240M	2432	R	99.0	6.2	0:10.82	/sage/abstract/loc
6705	rob	20	0	1147M	240M	2432	R	97.0	6.2	0:11.29	/sage/abstract/loc
6702	rob	20	0	1147M	240M	2432	R	94.0	6.2	0:11.20	/sage/abstract/loc
6704	rob	20	0	1147M	240M	2432	R	93.0	6.2	0:10.96	/sage/abstract/loc
4523	root	20	0	112M	31320	11044	S	7.0	0.8	0:46.16	/usr/bin/X :0 -br
6321	rob	20	0	906M	158M	41080	S	3.0	4.1	1:07.92	/usr/lib/firefox-3
6133	rob	20	0	184M	12200	9272	S	1.0	0.3	0:02.60	metacity --replace
6291	rob	20	0	288M	25128	15308	S	1.0	0.6	0:05.95	konsole
6310	rob	20	0	19788	1596	1108	R	0.0	0.0	0:41.52	htop
6178	rob	20	0	239M	14024	10748	S	0.0	0.4	0:01.65	/usr/lib/gnome-pan
6499	rob	20	0	716M	99M	23088	S	0.0	2.6	0:07.16	python /sage/abstr
6390	rob	20	0	906M	158M	41080	S	0.0	4.1	0:03.16	/usr/lib/firefox-3
6152	rob	20	0	253M	18500	12896	S	0.0	0.5	0:01.03	gnome-panel
6553	rob	20	0	1147M	272M	35864	S	0.0	7.0	0:31.03	/sage/abstract/loc
4540	root	20	0	11280	632	492	S	0.0	0.0	0:00.26	/usr/sbin/iqbalan
4375	root	20	0	85652	4428	3428	S	0.0	0.1	0:00.85	NetworkManager
4816	root	20	0	57112	6148	2504	S	0.0	0.2	0:01.51	/usr/lib/upower/up
6120	rob	20	0	46196	5748	2388	S	0.0	0.1	0:00.62	/usr/lib/libgconf2
6059	rob	20	0	43120	31560	1512	S	0.0	0.8	0:00.23	/sage/abstract/loc
4306	messageb	20	0	24412	1976	776	S	0.0	0.0	0:01.48	dbus-daemon --syst
6326	rob	20	0	906M	158M	41080	S	0.0	4.1	0:01.29	/usr/lib/firefox-3
6333	rob	20	0	906M	158M	41080	S	0.0	4.1	0:01.97	/usr/lib/firefox-3
6588	rob	20	0	1435M	471M	12692	S	0.0	12.1	0:04.67	/usr/lib/jvm/java-

Rob 0050014

Help F2Setup F3Search F4Invert F5Tree F6SortBy F7Nice F8Nice +F9Kill F10Quit

Google

abstract/deve... X

jsMath


```

1  [|||||] 30.5% Tasks: 254 total, 1 running
2  [|||] 11.1% Load average: 1.51 0.38 0.12
3  [|||||] 25.7% Uptime: 01:23:54
4  [||||] 13.0%
Mem [|||||] 1584/3887MB
Swp [|||||] 0/2047MB

```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
523	root	20	0	112M	31320	11044	S	7.0	0.8	0:46.27	/usr/bin/X :0 -br
5321	rob	20	0	906M	159M	41080	S	1.0	4.1	1:07.95	/usr/lib/firefox-3
5310	rob	20	0	19788	1596	1108	R	1.0	0.0	0:41.53	htop
5178	rob	20	0	239M	14024	10748	S	1.0	0.4	0:01.67	/usr/lib/gnome-pan
5133	rob	20	0	184M	12200	9272	S	0.0	0.3	0:02.60	metacity --replace
5291	rob	20	0	288M	25128	15308	S	0.0	0.6	0:05.95	konsole
5499	rob	20	0	716M	99M	23088	S	0.0	2.6	0:07.17	python /sage/abstr
5390	rob	20	0	906M	159M	41080	S	0.0	4.1	0:03.16	/usr/lib/firefox-3
5152	rob	20	0	253M	18500	12896	S	0.0	0.5	0:01.03	gnome-panel
5553	rob	20	0	1147M	272M	35864	S	0.0	7.0	0:31.03	/sage/abstract/loc
5440	root	20	0	11280	632	492	S	0.0	0.0	0:00.26	/usr/sbin/irqbalan
5375	root	20	0	85652	4428	3428	S	0.0	0.1	0:00.85	NetworkManager
5816	root	20	0	57112	6148	2504	S	0.0	0.2	0:01.51	/usr/lib/upower/up
5120	rob	20	0	46196	5748	2388	S	0.0	0.1	0:00.62	/usr/lib/libgconf2
5059	rob	20	0	43120	31560	1512	S	0.0	0.8	0:00.23	/sage/abstract/loc
5306	messageb	20	0	24412	1976	776	S	0.0	0.0	0:01.48	dbus-daemon --syst
5326	rob	20	0	906M	159M	41080	S	0.0	4.1	0:01.29	/usr/lib/firefox-3
5333	rob	20	0	906M	159M	41080	S	0.0	4.1	0:01.97	/usr/lib/firefox-3
5588	rob	20	0	1435M	471M	12692	S	0.0	12.1	0:04.67	/usr/lib/jvm/java-
5663	rob	20	0	1435M	471M	12692	S	0.0	12.1	0:00.04	/usr/lib/jvm/java-
5592	rob	20	0	1435M	471M	12692	S	0.0	12.1	0:03.48	/usr/lib/jvm/java-
5584	rob	20	0	1435M	471M	12692	S	0.0	12.1	0:00.25	/usr/lib/jvm/java-
5585	rob	20	0	1435M	471M	12692	S	0.0	12.1	0:00.24	/usr/lib/jvm/java-

Google

/abstract/deve... X

jsMath


```

1  [||]          1.3%]      Tasks: 254 total, 1 running
2  [          0.0%]      Load average: 1.51 0.38 0.12
3  [||]          4.7%]      Uptime: 01:23:56
4  [|]          0.7%]
Mem[|||||||||||||1584/3887MB]
Swp[                0/2047MB]

```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
523	root	20	0	112M	31320	11044	S	4.0	0.8	0:46.34	/usr/bin/X :0 -br
5321	rob	20	0	906M	159M	41080	S	0.0	4.1	1:07.95	/usr/lib/firefox-3
5310	rob	20	0	19788	1596	1108	R	0.0	0.0	0:41.55	htop
5178	rob	20	0	239M	14024	10748	S	0.0	0.4	0:01.67	/usr/lib/gnome-pan
5133	rob	20	0	184M	12200	9272	S	0.0	0.3	0:02.60	metacity --replace
5291	rob	20	0	288M	25128	15308	S	0.0	0.6	0:05.96	konsole
5499	rob	20	0	716M	99M	23088	S	0.0	2.6	0:07.17	python /sage/abstr
5390	rob	20	0	906M	159M	41080	S	0.0	4.1	0:03.16	/usr/lib/firefox-3
5152	rob	20	0	253M	18500	12896	S	0.0	0.5	0:01.03	gnome-panel
5553	rob	20	0	1147M	272M	35864	S	0.0	7.0	0:31.03	/sage/abstract/loc
5440	root	20	0	11280	632	492	S	0.0	0.0	0:00.26	/usr/sbin/irqbalan
5375	root	20	0	85652	4428	3428	S	0.0	0.1	0:00.85	NetworkManager
5816	root	20	0	57112	6148	2504	S	0.0	0.2	0:01.51	/usr/lib/upower/up
5120	rob	20	0	46196	5748	2388	S	0.0	0.1	0:00.62	/usr/lib/libgconf2
5059	rob	20	0	43120	31560	1512	S	0.0	0.8	0:00.23	/sage/abstract/loc
5306	messageb	20	0	24412	1976	776	S	0.0	0.0	0:01.48	dbus-daemon --syst
5326	rob	20	0	906M	159M	41080	S	0.0	4.1	0:01.29	/usr/lib/firefox-3
5333	rob	20	0	906M	159M	41080	S	0.0	4.1	0:01.97	/usr/lib/firefox-3
5588	rob	20	0	1435M	471M	12692	S	0.0	12.1	0:04.67	/usr/lib/jvm/java-
5663	rob	20	0	1435M	471M	12692	S	0.0	12.1	0:00.04	/usr/lib/jvm/java-
5592	rob	20	0	1435M	471M	12692	S	0.0	12.1	0:03.48	/usr/lib/jvm/java-
5584	rob	20	0	1435M	471M	12692	S	0.0	12.1	0:00.25	/usr/lib/jvm/java-
5585	rob	20	0	1435M	471M	12692	S	0.0	12.1	0:00.24	/usr/lib/jvm/java-

Google

abstract/deve... X

jsMath

```
{}, 20499971300000, [(20, 1, 15), 555550000000, 1
```

A larger list of inputs. Watch in htop.

```
long_summer = summation(range(100))
```

```
list(long_summer)
```

```
Finished 0
Finished 3
Finished 2
Finished 1
Finished 4
Finished 5
Finished 6
Finished 7
Finished 8
Finished 9
Finished 10
Finished 11
Finished 12
Finished 13
Finished 14
Finished 15
Finished 16
Finished 17
```



```
46499953500000), ((94,), {}), 46999953000000), ((95,), {}),
47499952500000), ((96,), {}), 47999952000000), ((97,), {}),
48499951500000), ((98,), {}), 48999951000000), ((99,), {}),
49499950500000)]
```

13 Cython

- Convert Python to compiled C
- Originally Pyrex, forked by Sage to Cython

```
def summation_python():
    sum = 0
    for i in range(10000000):
        sum = sum + i
    return sum
```

[evaluate](#)

```
timeit("summation_python()")
```

- Declare integer variables

13 Cython

- Convert Python to compiled C
- Originally Pyrex, forked by Sage to Cython

```
def summation_python():  
    sum = 0  
    for i in range(10000000):  
        sum = sum + i  
    return sum
```

[evaluate](#)

```
timeit("summation_python()")
```

- Declare integer variables
- Modify the for-loop

```
%cython  
def summation_cython():  
    cdef i, sum
```

13 Cython

- Convert Python to compiled C
- Originally Pyrex, forked by Sage to Cython

```
def summation_python():  
    sum = 0  
    for i in range(1000000):  
        sum = sum + i  
    return sum
```

[evaluate](#)

```
timeit("summation_python()")
```

- Declare integer variables
- Modify the for-loop

```
%cython  
def summation_cython():  
    cdef i, sum
```

13 Cython

- Convert Python to compiled C
- Originally Pyrex, forked by Sage to Cython

```
def summation_python():  
    sum = 0  
    for i in range(1000000):  
        sum = sum + i  
    return sum
```

[evaluate](#)

```
timeit("summation_python()")
```

- Declare integer variables
- Modify the for-loop

```
%cython  
def summation_cython():  
    cdef i, sum
```


13 Cython

- Convert Python to compiled C
- Originally Pyrex, forked by Sage to Cython

```
def summation_python():  
    sum = 0  
    for i in range(10000000):  
        sum = sum + i  
    return sum
```

```
timeit("summation_python()")
```

- Declare integer variables
- Modify the for-loop

```
%cython  
def summation_cython():  
    cdef i, sum  
    sum = 0
```

13 Cython

- Convert Python to compiled C
- Originally Pyrex, forked by Sage to Cython

```
def summation_python():  
    sum = 0  
    for i in range(1000000):  
        sum = sum + i  
    return sum
```

```
timeit("summation_python()")
```

5 loops, best of 3: 111 ms per loop

- Declare integer variables
- Modify the for-loop

```
%cython  
def summation_cython():  
    cdef i, sum  
    sum = 0
```

```
timeit("summation_python()")
```

5 loops, best of 3: 111 ms per loop

- Declare integer variables
- Modify the for-loop

```
%cython
def summation_cython():
    cdef i, sum
    sum = 0
    for i in range(1000000):
        sum = sum + i
    return sum
```

[evaluate](#)

```
timeit("summation_cython()")
```

- Speedups by factors of hundreds and thousands are not uncommon

- Declare integer variables
- Modify the for-loop

```
%cython
def summation_cython():
    cdef i, sum
    sum = 0
    for i in range(1000000):
        sum = sum + i
    return sum
```

[evaluate](#)

```
timeit("summation_cython()")
```

- Speedups by factors of hundreds and thousands are not uncommon
- f2c (Fortran to C) is also included

14 SageTeX

- Include Sage instructions in a L^AT_EX document

- Declare integer variables
- Modify the for-loop

```
%cython
def summation_cython():
    cdef i, sum
    sum = 0
    for i in range(1000000):
        sum = sum + i
    return sum
```

[evaluate](#)

```
timeit("summation_cython()")
```

- Speedups by factors of hundreds and thousands are not uncommon
- f2c (Fortran to C) is also included

14 SageTeX

- Include Sage instructions in a \LaTeX document

- Declare integer variables
- Modify the for-loop

```
%cython
def summation_cython():
    cdef int, sum
    sum = 0
    for i in range(1000000):
        sum = sum + i
    return sum
```

[evaluate](#)

```
timeit("summation_cython()")
```

- Speedups by factors of hundreds and thousands are not uncommon
- f2c (Fortran to C) is also included

14 SageTeX

- Include Sage instructions in a L^AT_EX document

- Declare integer variables
- Modify the for-loop

```
%cython
def summation_cython():
    cdef int i, sum
    sum = 0
    for i in range(1000000):
        sum = sum + i
    return sum
```

[evaluate](#)

```
timeit("summation_cython()")
```

- Speedups by factors of hundreds and thousands are not uncommon
- f2c (Fortran to C) is also included

14 SageTeX

- Include Sage instructions in a L^AT_EX document

- Declare integer variables
- Modify the for-loop

```
%cython
def summation_cython():
    cdef int i, sum
    sum = 0
    for 0 <= i < 1000000:
        sum = sum + i
    return sum
```

```
timeit("summation_cython()")
```

evaluate

- Speedups by factors of hundreds and thousands are not uncommon
- f2c (Fortran to C) is also included

14 SageTeX

- Include Sage instructions in a L^AT_EX document

- Declare integer variables
- Modify the for-loop

```
%cython
def summation_cython():
    cdef int i, sum
    sum = 0
    for i in range(1000000):
        sum = sum + i
    return sum
```

[_home_rob...8_code_sage72_spyx.c](#)

[_home_rob...ode_sage72_spyx.html](#)

```
timeit("summation_cython()")
```

evaluate

- Speedups by factors of hundreds and thousands are not uncommon
- f2c (Fortran to C) is also included

14 SageTeX

- Include Sage instructions in a L^AT_EX document

- Declare integer variables
- Modify the for-loop

```
%cython
def summation_cython():
    cdef int i, sum
    sum = 0
    for i in range(1000000):
        sum = sum + i
    return sum
```

[home_rob...8_code_sage72_spyx.c](#)

[home_rob...ode_sage72_spyx.html](#)

```
timeit("summation_cython()")
```

[evaluate](#)

- Speedups by factors of hundreds and thousands are not uncommon
- f2c (Fortran to C) is also included

14 SageTeX

- Include Sage instructions in a L^AT_EX document

Generated by Cython 0.13 on Tue May 10 07:28:45 2011

new output: [_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.c](#)

```
1: include "interrupt.pxi" # ctrl-c interrupt block support
2: include "stdsage.pxi" # ctrl-c interrupt block support
3:
4: include "cdefs.pxi"
5: def summation_cython():
6:     cdef int i, sum
7:     sum = 0
8:     for 0 <= i < 10000000:
9:         sum = sum + i
10:
11:     return sum
```

Generated by Cython 0.13 on Tue May 10 07:28:45 2011

new output: [_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.c](#)

```
include "interrupt.pxi" # ctrl-c interrupt block support
```

```
/* "/home/rob/.sage/temp/tiger/5553/spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.pyx":2
 *
 * include "interrupt.pxi" # ctrl-c interrupt block support          # <-----
 * include "stdsage.pxi" # ctrl-c interrupt block support
 *
 */
__pyx_t_1 = PyDict_New(); if (unlikely(!_pyx_t_1)) {__pyx_filename = __pyx_f[0]; __pyx_lineno = 2; __pyx_clineno = __LINE__; goto __pyx_li_error;}
__Pyx_DECREF(((PyObject *)__pyx_t_1));
if (PyObject_SetAttr(__pyx_m, __pyx_n_s_test, ((PyObject *)__pyx_t_1)) < 0) {__pyx_filename = __pyx_f[0]; __pyx_lineno = 2; __pyx_clineno = __LINE__; goto __pyx_li_error;}
__Pyx_DECREF(((PyObject *)__pyx_t_1)); __pyx_t_1 = 0;
```

```
include "stdsage.pxi" # ctrl-c interrupt block support
```

```
include "cdefs.pxi"
```

```
def summation_cython():
```

```
    cdef int i, sum
```

```
    sum = 0
```

```
    for 0 <= i < 1000000:
```

```
        sum = sum + i
```

```
    return sum
```


Generated by Cython 0.13 on Tue May 10 07:28:45 2011

new output: [_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.c](#)

```
include "interrupt.pxi" # ctrl-c interrupt block support
```

```
/* "/home/rob/.sage/temp/tiger/5553/spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.pyx":2
 *
 * include "interrupt.pxi" # ctrl-c interrupt block support          # <oooooooooooo
 * include "stdsage.pxi" # ctrl-c interrupt block support
 *
 */
__pyx_t_1 = PyDict_New(); if (unlikely(!__pyx_t_1)) {__pyx_filename = __pyx_f[0]; __pyx_lineno = 2; __pyx_clineno = __LINE__; goto __pyx_l1_error;}
__Pyx_DECREF(((PyObject *)__pyx_t_1));
if (PyObject_SetAttr(__pyx_m, __pyx_n_s_test, ((PyObject *)__pyx_t_1)) < 0) {__pyx_filename = __pyx_f[0]; __pyx_lineno = 2; __pyx_clineno = __LINE__; goto __pyx_l1_error;}
__Pyx_DECREF(((PyObject *)__pyx_t_1)); __pyx_t_1 = 0;
```

```
include "stdsage.pxi" # ctrl-c interrupt block support
```

```
include "cdefs.pxi"
```

```
def summation_cython():
```

```
    cdef int i, sum
```

```
    sum = 0
```

```
/* "/home/rob/.sage/temp/tiger/5553/spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.pyx":8
 * def summation_cython():
 *     cdef int i, sum
 *     sum = 0          # <oooooooooooo
 *     for 0 <= i < 1000000:
 *         sum = sum + i
 *
 */
__pyx_v_sum = 0;
```

```
    for 0 <= i < 1000000:
```

```
        sum = sum + i
```

```
    return sum
```

Generated by Cython 0.13 on Tue May 10 07:28:45 2011

new output: [_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.c](#)

```
include "interrupt.pxi" # ctrl-c interrupt block support
```

```
/* "/home/rob/.sage/temp/tiger/5553/spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.pyx":2
 *
 * include "interrupt.pxi" # ctrl-c interrupt block support
 * include "stdsage.pxi" # ctrl-c interrupt block support
 */
__pyx_t_1 = PyDict_New(); if (unlikely(! __pyx_t_1)) { __pyx_filename = __pyx_f[0], __pyx_lineno = 2; __pyx_clineno = __LINE__; goto __pyx_li_error;}
__Pyx_DECREF(((PyObject *) __pyx_t_1));
if (PyObject_SetAttr(__pyx_m, __pyx_n_s_test, ((PyObject *) __pyx_t_1)) < 0) { __pyx_filename = __pyx_f[0], __pyx_lineno = 2; __pyx_clineno = __LINE__; goto __pyx_li_error;}
__Pyx_DECREF(((PyObject *) __pyx_t_1)); __pyx_t_1 = 0;
```

```
include "stdsage.pxi" # ctrl-c interrupt block support
```

```
include "cdefs.pxi"
def summation_cython():
    cdef int i, sum
    sum = 0
```

```
/* "/home/rob/.sage/temp/tiger/5553/spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.pyx":8
 * def summation_cython():
 *     cdef int i, sum
 *     sum = 0
 *     for 0 <= i < 1000000:
 *         sum = sum + i
 */
__pyx_v_sum = 0;
```

```
for 0 <= i < 1000000:
    sum = sum + i
return sum
```

```
/* "/home/rob/.sage/temp/tiger/5553/spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.pyx":11
 *     for 0 <= i < 1000000:
 *         sum = sum + i
 *     return sum
 */
```



```

include "stdsage.pxi" # ctrl-c interrupt block support

include "cdefs.pxi"
def summation_cython():
    cdef int i, sum
    sum = 0

/* "/home/rob/.sage/temp/tiger/5553/spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.pyx":8
* def summation_cython():
*     cdef int i, sum
*     sum = 0
*     for 0 <= i < 1000000:
*         sum = sum + i
*/
__pyx_v_sum = 0;

for 0 <= i < 1000000:
    sum = sum + i
l: return sum

/* "/home/rob/.sage/temp/tiger/5553/spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.pyx":11
*     for 0 <= i < 1000000:
*         sum = sum + i
*     return sum
*/
__Pyx_DECREF(__pyx_r);
__pyx_t_1 = PyInt_FromLong(__pyx_v_sum); if (unlikely(!__pyx_t_1)) {__pyx_filename = __pyx_f[0]; __pyx_lineno = 11; __pyx_clineno = __LINE__; goto __pyx_ll_error;}
__Pyx_GOTREF(__pyx_t_1);
__pyx_r = __pyx_t_1;
__pyx_t_1 = 0;
goto __pyx_L0;

__pyx_r = Py_None; __Pyx_INCREF(Py_None);
goto __pyx_L0;
__pyx_ll_error:
__Pyx_DECREF(__pyx_t_1);
__Pyx_AddTraceback("_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.summation_cython");
__pyx_r = NULL;
__pyx_L0:
__Pyx_XDECREF(__pyx_r);
__Pyx_RefNannyFinishContext();
return __pyx_r;
}

```


Generated by Cython 0.13 on Tue May 10 07:28:45 2011

Now output: [_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.c](#)

```
include "interrupt.pxi" # ctrl-c interrupt block support
```

```
/* "/home/rob/.sage/temp/tiger/5553/spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.pyx":2
 *
 * include "interrupt.pxi" # ctrl-c interrupt block support # <ooooooooooooo
 * include "stdsage.pxi" # ctrl-c interrupt block support
 *
 */
__pyx_t_1 = PyDict_New(); if (unlikely(!__pyx_t_1)) {__pyx_filename = __pyx_f[0]; __pyx_lineno = 2; __pyx_clineno = __LINE__; goto __pyx_l1_error;}
__Pyx_DECREF(((PyObject *)__pyx_t_1));
if (PyObject_SetAttr(__pyx_m, __pyx_n_s_test, ((PyObject *)__pyx_t_1)) < 0) {__pyx_filename = __pyx_f[0]; __pyx_lineno = 2; __pyx_clineno = __LINE__; goto __pyx_l1_error;}
__Pyx_DECREF(((PyObject *)__pyx_t_1)); __pyx_t_1 = 0;
```

```
include "stdsage.pxi" # ctrl-c interrupt block support
```

```
include "cdefs.pxi"
def summation_cython():
    cdef int i, sum
    sum = 0
```

```
/* "/home/rob/.sage/temp/tiger/5553/spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.pyx":8
 * def summation_cython():
 *     cdef int i, sum
 *     sum = 0 # <ooooooooooooo
 *     for 0 <= i < 1000000:
 *         sum = sum + i
 *
 */
__pyx_v_sum = 0;
for 0 <= i < 1000000:
    sum = sum + i
return sum
```

Generated by Cython 0.13 on Tue May 10 07:28:45 2011

new output: [_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.c](#)

```

1: include "interrupt.pxi" # ctrl-c interrupt block support
2: include "stdsage.pxi" # ctrl-c interrupt block support
3:
4: include "cdefs.pxi"
5: def summation_cython():
6:     cdef int i, sum
7:     sum = 0
8:
9:     /* "/home/rob/.sage/temp/tiger/5553/spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx/_home_rob_sage_sage_notebook_sagenb_home_admin_2388_code_sage72_spyx_0.pyx":8
10: * def summation_cython():
11: *     cdef int i, sum
12: *     sum = 0 # <oooooooooooooo
13: *     for 0 <= i < 1000000:
14: *         sum = sum + i
15: */
16:     __pyx_v_sum = 0;
17:     for 0 <= i < 1000000:
18:         sum = sum + i
19:     return sum

```

- Declare integer variables
- Modify the for-loop

```
%cython
def summation_cython():
    cdef int i, sum
    sum = 0
    for i in range(1000000):
        sum = sum + i
    return sum
```



[home_rob...8_code_sage72_spyx.c](#)

[home_rob...ode_sage72_spyx.html](#)

```
timeit("summation_cython()")
```

[evaluate](#)

- Speedups by factors of hundreds and thousands are not uncommon
- f2c (Fortran to C) is also included

14 SageTeX

- Include Sage instructions in a L^AT_EX document

- Declare integer variables
- Modify the for-loop

```
%cython
def summation_cython():
    cdef int i, sum
    sum = 0
    for i in range(1000000):
        sum = sum + i
    return sum
```

[home_rob...8_code_sage72_spyx.c](#)

[home_rob...ode_sage72_spyx.html](#)

```
timeit("summation_cython()")
```

evaluate

- Speedups by factors of hundreds and thousands are not uncommon
- f2c (Fortran to C) is also included

14 SageTeX

- Include Sage instructions in a L^AT_EX document

```
sage: A4 = patch.ambient_space()
sage: origin = A4.subscheme(A4.gens()) # the origin (0,0,0,0)
sage: A4.subscheme(Jac) == origin
True
```

The most basic invariant of an isolated hypersurface singularity is its *Milnor number*, which is the vector space dimension of $\mathbb{C}[\bar{x}]/\text{Jac}(f(\bar{x}))$. For the conifold, it is one:

```
sage: Jac.vector_space_dimension()
1
```

In fact, the converse is also true: A 3-dimensional isolated singularity of Milnor number one is a conifold. Note, however, that higher Milnor numbers no longer uniquely determine the singularity.

Exercise 9. Use Sage to compute the Milnor number of the singularity $\mathbb{C}^2/\mathbb{Z}_n$ for $n \in \{2, 3, \dots, 10\}$.

```
timeit("summation_python()")
```

5 loops, best of 3: 111 ms per loop

- Declare integer variables
- Modify the for-loop

```
%cython
def summation_cython():
    cdef int i, sum
    sum = 0
    for i in range(1000000):
        sum = sum + i
    return sum
```

I

[_home_rob...8_code_sage72_spyx.c](#) [_home_rob...ode_sage72_spyx.html](#)

```
timeit("summation_cython()")
```

625 loops, best of 3: 627 μ s per loop

- Speedups by factors of hundreds and thousands are not uncommon


```
sum = 0
for 0 <= i < 1000000:
    sum = sum + i
return sum
```

[_home_rob...8_code_sage72_spyx.c](#) [_home_rob...ode_sage72_spyx.html](#)

```
timeit("summation_cython()")
```

625 loops, best of 3: 627 µs per loop

- Speedups by factors of hundreds and thousands are not uncommon
- f2c (Fortran to C) is also included

14 SageTeX

- Include Sage instructions in a \LaTeX document
- SageTeX isolates the commands, evaluates them, pastes in results
- Also creates and inserts plots
- Formats Sage code

3.4 The Conifold

```
sum = 0
for 0 <= i < 1000000:
    sum = sum + i
return sum
```

[_home_rob...8_code_sage72_spyx.c](#) [_home_rob...ode_sage72_spyx.html](#)

`timeit("summation_cython()")`

625 loops, best of 3: 627 μ s per loop

- Speedups by factors of hundreds and thousands are not uncommon
- f2c (Fortran to C) is also included

14 SageTeX

- Include Sage instructions in a \LaTeX document
- SageTeX isolates the commands, evaluates them, pastes in results
- Also creates and inserts plots
- Formats Sage code

3.4 The Conifold

3.4 The Conifold

A singularity that occurs very often is the conifold, which is the simplest non-quotient singularity. In terms of toric geometry, it is defined by the non-simplicial cone over a minimal lattice square at distance 1:

```
sage: conifold = toric_varieties.Conifold() 97
sage: conifold.is_smooth() 98
False 99
sage: conifold.is_orbifold() 100
False 101
sage: square_cone = conifold.fan().generating_cone(0) 102
sage: square_cone.rays() 103
(N(0, 0, 1), N(0, 1, 1), N(1, 0, 1), N(1, 1, 1)) 104
sage: patch = conifold.affine_algebraic_patch(square_cone) 105
sage: patch 106
Closed subscheme of Affine Space of dimension 4 over Rational 107
Field defined by: 108
z0*z2 - z1*z3 109
```

The conifold is not smooth because the hypersurface equation is not transverse at $\bar{z} = (0, 0, 0, 0)$. More precisely, the singularities are the variety of the Jacobian ideal

3.4 The Conifold

A singularity that occurs very often is the conifold, which is the simplest non-quotient singularity. In terms of toric geometry, it is defined by the non-simplicial cone over a minimal lattice square at distance 1:

```
sage: conifold = toric_varieties.Conifold() 97
sage: conifold.is_smooth() 98
False 99
sage: conifold.is_orbifold() 100
False 101
sage: square_cone = conifold.fan().generating_cone(0) 102
sage: square_cone.rays() 103
(N(0, 0, 1), N(0, 1, 1), N(1, 0, 1), N(1, 1, 1)) 104
sage: patch = conifold.affine_algebraic_patch(square_cone) 105
sage: patch 106
Closed subscheme of Affine Space of dimension 4 over Rational 107
Field defined by: 108
z0*z2 - z1*z3 109
```

Minimal lattice square at distance 1.

```
sage: conifold = toric_varieties.Conifold() 97
sage: conifold.is_smooth() 98
False 99
sage: conifold.is_orbifold() 100
False 101
sage: square_cone = conifold.fan().generating_cone(0) 102
sage: square_cone.rays() 103
(N(0, 0, 1), N(0, 1, 1), N(1, 0, 1), N(1, 1, 1)) 104
sage: patch = conifold.affine_algebraic_patch(square_cone) 105
sage: patch 106
Closed subscheme of Affine Space of dimension 4 over Rational 107
Field defined by: 108
z0*z2 - z1*z3 109
```

The conifold is not smooth because the hypersurface equation is not transverse at $\bar{z} = (0, 0, 0, 0)$. More precisely, the singularities are the variety of the Jacobian ideal

$$\text{Jac}(f) = \left\langle f, \frac{\partial f}{\partial z_0}, \dots, \frac{\partial f}{\partial z_3} \right\rangle \quad (37)$$

For the conifold, it is

minimal lattice square at distance 1.

```
sage: conifold = toric_varieties.Conifold() 97
sage: conifold.is_smooth() 98
False 99
sage: conifold.is_orbifold() 100
False 101
sage: square_cone = conifold.fan().generating_cone(0) 102
sage: square_cone.rays() 103
(N(0, 0, 1), N(0, 1, 1), N(1, 0, 1), N(1, 1, 1)) 104
sage: patch = conifold.affine_algebraic_patch(square_cone) 105
sage: patch 106
Closed subscheme of Affine Space of dimension 4 over Rational 107
Field defined by: 108
z0*z2 - z1*z3 109
```

The conifold is not smooth because the hypersurface equation is not transverse at $\bar{z} = (0, 0, 0, 0)$. More precisely, the singularities are the variety of the Jacobian ideal

$$\text{Jac}(f) = \left\langle f, \frac{\partial f}{\partial z_0}, \dots, \frac{\partial f}{\partial z_3} \right\rangle \quad (37)$$

For the conifold, it is


```
sage: Jac.vector_space_dimension()
```

```
1
```

In fact, the converse is also true: A 3-dimensional isolated singularity of Milnor number one is a conifold. Note, however, that higher Milnor numbers no longer uniquely determine the singularity.

Exercise 9. Use Sage to compute the Milnor number of the singularity $\mathbb{C}^2/\mathbb{Z}_n$ for $n \in \{2, 3, \dots, 10\}$.

evaluate

15 Philosophy

evaluate

15 Philosophy

Mathematica: "Why You Do Not Usually Need to Know about Internals"
in the online Mathematica Documentation Center.

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated*.

15 Philosophy

Mathematica: "Why You Do Not Usually Need to Know about Internals"
in the online Mathematica Documentation Center.

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews
Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1994.

15 Philosophy

Mathematica: "Why You Do Not Usually Need to Know about Internals"
in the online Mathematica Documentation Center.

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews
Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1994.

15 Philosophy

Mathematica: "Why You Do Not Usually Need to Know about Internals"
in the online Mathematica Documentation Center.

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews

Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1994.

15 Philosophy

Mathematica: "Why You Do Not Usually Need to Know about Internals"
in the online Mathematica Documentation Center.

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews

Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press

15 Philosophy

Mathematica: "Why You Do Not Usually Need to Know about Internals"
in the online Mathematica Documentation Center.

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. *For the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews

Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1994.

15 Philosophy

Mathematica: "Why You Do Not Usually Need to Know about Internals"
in the online Mathematica Documentation Center.

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews

Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1994.

15 Philosophy

Mathematica: "Why You Do Not Usually Need to Know about Internals"
in the online Mathematica Documentation Center.

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews
Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1994.

15 Philosophy

Mathematica: "Why You Do Not Usually Need to Know about Internals"
in the online Mathematica Documentation Center.

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews
Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1994.

15 Philosophy

Mathematica: "Why You Do Not Usually Need to Know about Internals"
in the online Mathematica Documentation Center.

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews
Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1994.

15 Philosophy

Mathematica: "Why You Do Not Usually Need to Know about Internals"
in the online Mathematica Documentation Center.

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. *For the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
in C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews
Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1994.

15 Philosophy

Mathematica: "Why You Do Not Usually Need to Know about Internals"
in the online Mathematica Documentation Center.

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. *For the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews
Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1994.

15 Philosophy

Mathematica: "Why You Do Not Usually Need to Know about Internals"
in the online Mathematica Documentation Center.

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. *For the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews
Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1994.

15 Philosophy

Mathematica: "Why You Do Not Usually Need to Know about Internals"
in the online Mathematica Documentation Center.

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. *For the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews
Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1994.

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same

could realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be irrelevant.

Similarly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze algorithms in order to predict which way of doing a given computation will be the most efficient. And there indeed occasionally major improvements that you will be able to make in specific computations as a result of analyses.

Most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated* [this is added], and even given a basic description of the algorithm used for a particular purpose, it is usually very difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will behave in particular circumstances.

ser (Founder of GAP): "An invitation to computational group theory."

Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, 1992, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press,

can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way.

could realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be irrelevant.

Similarly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze algorithms in order to predict which way of doing a given computation will be the most efficient. And there indeed occasionally major improvements that you will be able to make in specific computations as a result of analyses.

Most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated* [as added], and even given a basic description of the algorithm used for a particular purpose, it is usually very difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will behave in particular circumstances.

ser (Founder of GAP): "An invitation to computational group theory."

Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, 1992, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press,

can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way.

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same

You should realize at the outset that while knowing about the internals of Mathematica may be of intellectual interest, it is usually much less important in practice than you might at first suppose.

...

Indeed, in almost all practical uses of Mathematica, issues about how Mathematica works inside turn out to be largely irrelevant.

...

Particularly in more advanced applications of Mathematica, it may sometimes seem worthwhile to try to analyze internal algorithms in order to predict which way of doing a given computation will be the most efficient. And there are indeed occasionally major improvements that you will be able to make in specific computations as a result of such analyses.

But most often the analyses will not be worthwhile. For *the internals of Mathematica are quite complicated* [emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same

[emphasis added], and even given a basic description of the algorithm used for a particular purpose, it is usually extremely difficult to reach a reliable conclusion about how the detailed implementation of this algorithm will actually behave in particular circumstances.

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
in C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldova several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

• Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

• Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

• Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
in C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

• Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

• Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

• Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
 In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."

In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>

J. Neubüser (Founder of GAP): "An invitation to computational group theory."
In C. M. Campbell, T. C. Hurley, E. F. Robertson, S. J. Tobin, and J. J. Ward, editors, Groups 93 Galway/St. Andrews, Volume 2, volume 212 of London Mathematical Society Lecture Note Series, pages 457-475. Cambridge University Press, 1995.

You can read Sylow's Theorem and its proof in Huppert's book in the library without even buying the book and then you can use Sylow's Theorem for the rest of your life free of charge, but... for many computer algebra systems license fees have to be paid regularly for the total time of their use. In order to protect what you pay for, you do not get the source, but only an executable, i.e. a black box. You can press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research... means moving in a most undesirable direction. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?

Available at <http://buzzard.ups.edu/talks.html>