

## **SAGE Day 9 -- The Sage Math Software System**

# **The Sage Project: Creating A Viable Free Open Source Alternative to Magma, Maple, Mathematica and Matlab**

**William Stein**

**Sage Days 9**



**irmacs**



---

# Acknowledgement:

**Thank Organizers:** Thanks for the generous support from ACCELERATE BC, Irmacs, etc., and to the other organizers -- Nils Bruin, Bill Casselman, David Austin, and Robert Bradshaw.

---

# What is the History of Sage?



1. **Berkeley 1995-1998:** Doing my Ph.D. in Number Theory (Modular Forms) under Hendrik Lenstra, and wrote a massive amount of open source C++ code; Allan Steel and David Kohel convinced me in 1998 to **switch to Magma**.
2. **Harvard 1998-2003:** I developed a lot of code in

Magma, and eagerly convinced many people in Number Theory to use Magma, since Magma is much better at almost everything I needed than any other software on the planet.

... (interlude) ...

---

# Linus Torvalds on Open Source

I think, fundamentally, open source does tend to be more stable software. It's the right way to do things. I compare it to science versus witchcraft. In science, the whole system builds on people looking at other people's results and building on top of them. In witchcraft, somebody had a small secret and guarded it -- but never allowed others to really understand it and build on it.

**Traditional software is like witchcraft.** In history, witchcraft just died out. The same will happen in software. When problems get serious enough, you can't have one person or one company guarding their secrets. You have to have everybody share in knowledge.

-- Linus Torvalds

---

## **Some Timings Pages for Magma**

[\*\*NEW: Magma V2.14 Finite Field Computations Timings \(October 2007\)\*\*](#)

[\*\*Hermite Normal Form Timings Page \(June 2006\)\*\*](#)

[\*\*Magma V2.12 is apparently "Overall Best in the World at Polynomial GCD" :-\)\*\*](#) (May 2005)

[\*\*Conquering Inseparability \(March 2005\)\*\*](#)

[\*\*Allan Steel's Gröbner Basis Timings Page \(October 2004\)\*\*](#)

---

---

## J. Neubuser on Open Source

"You can read Sylow's Theorem and its proof in Huppert's book in the library [...] 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 [...]. You 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?"

– J. Neubuser in 1993, who started GAP, which won last night's Jenk's prize that "recognizes outstanding software engineering contributions in the field of computer algebra and encourages future excellence in such research and development".

# Wolfram: Viable Non-commercial Math Software is Impossible

**Wolfram:** There's another thing, quite honestly, that that community has a hard time with. They sort of hate one aspect of what I have done, which is to take intellectual developments and make a company out of them and sell things to people.

**DDJ:** Probably not surprising, if mathematicians are the most puristic of scientists.

**Wolfram:** My own view of that, which has hardened over the years, is, my god, that's the right thing to do. If you look at what's happened with TeX, for example, which went in the other direction...well, Mathematica could not have been brought to where it is today if it had not been done as a commercial effort. The amount of money that has to be spent to do all the details of development, you just **can't support that in any other way than this unique American idea of the entrepreneurial company.**



-- Stephen Wolfram, 1993, Doctor Dobbs  
Journal Interview

---

[Mathematica Tutorial](#): "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 one might at first suppose. [...]"

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, 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.

A typical problem is that Mathematica has many internal optimizations..."

---

# John Cannon's: Viable Non-commercial Math Software *is Possible*

John Cannon (the 2006 Jenk's prize winner) proved Stephen Wolfram wrong with Magma, which is *not* developed via "this unique American idea of the entrepreneurial company."

# MAGMA

COMPUTER • ALGEBRA

```
To: pari-dev@list.cr.yp.to
* Subject: Magma and Pari
* From: john@maths.usyd.edu.au (John Cannon)
* Date: Sun, 12 Mar 2000 15:35:37 +1100 (EST)
```

```
>then spent quite a lot of time implementing more efficient
>algorithms wherever they could. [ that's actually what you pay
> Magma for: so that they can pay their developpers... ]
```

The implementation of almost all modules in Magma is paid for by research grants. The license charge covers the costs of maintaining and distributing Magma. To get research grants we have to show that research funds wont be spent on such support activities.

-- John Cannon

Magma -- both the software and the development model -- was a major inspiration for Sage...

---

## Sage is Essentially Doomed

Richard Fateman in December 2005 posted his opinion of Sage to sci.math.symbolic:

"By avoiding applications (say, to engineering design, finance, education, scientific visualization, etc etc) the activity [Sage] is essentially doomed. Why? Government funding for people or projects will be a small percentage of the funding for pure mathematics. That's not much. And the future is pretty grim."

-- Richard Fateman

---

# What is the History of Sage?



1. **Berkeley 1995-2000:** I did my Ph.D. in Number Theory (Modular Forms) under Hendrik Lenstra, and wrote a massive amount of C++ code, then moved entirely to Magma.
2. **Harvard 2000-2003:** I developed a lot of code in Magma, and convinced many people in my area to use Magma.
3. **Harvard 2003-2005:** I realized that I had made a **major mistake** in embracing Magma as my core research tool: Magma is proprietary, the development model is closed, no way to write compiled code for Magma, Magma is expensive which is bad for students and profs, and the language itself lacked many features (e.g., user defined classes) that I had requested repeatedly for over 5 years; and yet the **design** of much of Magma is frickin' brilliant...
4. **Harvard late 2004:** I get tenure at UCSD and start investigating nontrivial options.
5. **Harvard, January 2005:** I release Sage-0.1 and a long lonely year of very hard work follows.
6. **UC San Diego, February 2006:** Sage-1.0 and Sage Days 1. Get some help from David Joyner and David Kohel.
7. **University of Washington, October 2006:** Sage Days 2; Start build a developer community that now numbers well over 100 and is writing **massive amounts of new very high quality code**.

8. **November 2007:** Sage wins first prize in Scientific Category of Trophees du Libre (3000 euros, a laptop, etc.)

---

## What is Sage?



1. A **completely free distribution** of the world's best open source mathematical software that builds easily from source on Linux and OS X (and soon on Solaris and Microsoft Windows). Basically Sage **includes nearly everything** you need standard.
2. Over three hundred thousand lines of **new code and documentation** that implements large amounts of new functionality and provides a graphical interface. A massive automated test suite.
3. **Unified interfaces to Magma, Mathematica, Maple, Matlab, Axiom, Mupad, Singular, Macaulay2, etc.**, so it is easier for you to use all these programs together (for example, when migrating to Sage or making sure your new Sage code is faster than anything else out there).



---

## Some Examples

```
2 + 3
```

```
5
```

```
4/5 + 7/8 + x^(pi + e + e)
```

```
x^(pi + 2*e) + 67/40
```

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

```
plot3d(sin(x*cos(y)), (x,-2,5),(y,-2,5), adaptive=True, spin=True,  
opacity=0.7)
```

```
# Yoda! -- over 50,000 triangles.
```

```

from scipy import io
x = io.loadmat(DATA + 'yodapose.mat')
from sage.plot.plot3d.index_face_set import IndexFaceSet
V = x['V']; F3=x['F3']-1; F4=x['F4']-1
Y = IndexFaceSet(F3,V,color=Color('#00aa00')) +
IndexFaceSet(F4,V,color=Color('#00aa00'))
Y = Y.rotateX(-1)

```

```
Y.show(aspect_ratio=[1,1,1], frame=False, figsize=4)
```

```

var('x')
show(integrate(sin(x)*cos(pi*x) + pi^3))

```

$$\frac{-\cos((\pi+1)x)}{2(\pi+1)} - \frac{\cos((1-\pi)x)}{2(1-\pi)} + \pi^3 x$$

```

R.<x> = PolynomialRing(IntegerModRing(389*997))
p = x^20 + 387832*x^19 + 3*x^2 + 2*x + 387828
p.small_roots(beta=0.1)

```

```
[1]
```

```
p.small_roots??
```

```

%hide
# Requires code not yet in Sage-3.0.6
def hmm_predict(x, training_window, verbose, n):
    # parameters in the training window.
    mu = x[-training_window:].mean()
    std = x.standard_deviation()

    # Define a Gaussian HMM with n states:
    A = [[1/n]*n]*n # n x n matrix of 1/4th's
    eps = 4*std/(n-1)
    B = [(-2*std + i*eps + mu, std) for i in range(n)]
    C = [1/n]*n

    H = hmm.GaussianHiddenMarkovModel(A, B, C)
    for i in range(n): H.fix_emissions(i)

    try:
        # Train using all data ever known
        H.baum_welch(x[-4*training_window:], max_iter=1)
        # Then train using just the training window
        H.baum_welch(x[-training_window:], max_iter=1)
    except RuntimeError:
        print "Error running Baum-Welch."
        return 0

```

```

# find current most likely state
viterbi = H.viterbi(x[-training_window:])[0]
state = viterbi[-1]
if verbose:
    print "-"*20
    print "state = ", state

# find next expected value, which is
# (prob next state = 0)*(mean 0) + (prob of next state =
1)*(mean 1)
A = H.transition_matrix()
B = H.emission_parameters()
z = max(A[state])
if verbose:
    html(str(A))
    html(str(B))
    print "Prob of prediction = ", z
if B[list(A[state]).index(z)][0] < 0:
    return -1
else:
    return 1

def hmm_strategy(start, end, training_window, verbose, strategy,
number_of_states):
    x = w[start: end]
    y = []
    for i in range(len(x)):

        if strategy == 'hmm':
            pr = hmm_predict(w[:start] + x[:i], training_window,
verbose, number_of_states)
        elif strategy == 'psychic':
            if w[start+i] < 0:
                pr = -1
            else:
                pr = 1
        elif strategy == 'contrarian':
            if w[start+i-1] < 0:
                pr = 1
            else:
                pr = -1
        elif strategy == 'momentum':
            if w[start+i-1] < 0:
                pr = -1
            else:
                pr = 1
        elif strategy == 'random':

```

```

        if random() < 0.5:
            pr = -1
        else:
            pr = 1
    elif strategy == 'buy&hold':
        pr = 1
    elif strategy == 'moving average':
        # program around that stupid exponential moving average
        ignores last term
        # in input -- i consider this a bug. When bug is fixed,
        this will have
        # to be changed!
        avg =
(P[:start+i]+finance.TimeSeries([0])).exponential_moving_average(0.3)[-1:
        if P[start+i-1] < avg:
            pr = 1
        else:
            pr = -1

    if verbose:
        print "pred = ", pr, "  prev = ", w[i-1], "  next = ", w[i]
    if pr < 0:
        y.append(-1)
    else:
        y.append(1)
y = finance.TimeSeries(y)
try:
    return x.correlation(y), y
except ZeroDivisionError:
    print "Correlation not defined; returning 0"
    return 0, y

def shift_plot(v, s, draw_points=True, **kwds):
    L = line([(i,v[i-s]) for i in range(s,s+len(v))], **kwds)
    if draw_points: L += points([(i,v[i-s]) for i in
range(s,s+len(v))], **kwds)
    L.ymin(min(v)); L.ymax(max(v))
    return L

stock = 0; P =0; w = 0; csv=0; sigma = 0; mu = 0; stock_symbol0 = ''
@interact
def f(stock_symbol = 'ivv',
    history=range_slider(-300,0,default=(-10,0)),
    training_window=(100,(1..500)),
    verbose=False,
    strategy = selector(['hmm', 'contrarian', 'momentum',
                        'buy&hold', 'random', 'moving average',
'psychic'],

```

```

        buttons=True),
    number_of_states = selector([2..6], buttons=True),
    draw_points = True
):
global stock, P, w, csv, sigma, mu, stock_symbol0
if stock_symbol != stock_symbol0:
    print "Stock symbol: ", stock_symbol
    stock = finance.Stock(stock_symbol)
    csv = stock.historical('Jan+1,+1990')
    P = stock.close('Jan+1,+1990')
    P = P + P[-1:] # one extra value so we can make predictions
about tomorrow
    w = P.log().diffs()
    stock_symbol0 = stock_symbol

print "Overall Hurst Exponent: ", P.hurst_exponent()

last = csv[history[1]].date if history[1] < 0 else '(Day after
%s)'%csv[-1].date

print "From %s to %s (%s days)"%(csv[history[0]].date, last,
history[1]-history[0])
n = len(P)

try:
    c, y = hmm_strategy(n+history[0], n+history[1], training_window,
        verbose, strategy, number_of_states)
    print "Correlation ----> ", c
except ZeroDivisionError:
    print "unable to compute correlation because of zero division
error."

if history[1] == 0:
    print "\n** %s's forecast is %s ** \n"%(last, 'up' if y[-1] ==
1 else 'down')
    history = [history[0], history[1]-1]

reality = P[history[0]:history[1]]
print "Hurst Exponent of Window: ", reality.hurst_exponent()

# Compute how much the strategy makes at any given time.
# This is tricky because of shorting.
strategy = [reality[0]]
for i in range(1, len(reality)):
    if y[i-1] == 1: # long
        strategy.append(strategy[i-1]*(reality[i]/reality[i-1]))
    else: # short

```

```
strategy.append(strategy[i-1]*((reality[i-1] +
(reality[i-1] - reality[i])) / reality[i-1] ))

print "Strategy/Index ---> ", strategy[-1] / reality[-1]
print "Profit ---> ", strategy[-1] / reality[0]
G = shift_plot(reality, history[0], draw_points=draw_points) +
shift_plot(strategy, history[0], rgbcolor='red',
draw_points=draw_points)
G.show(figsize=[8,3],frame=True, xmin=history[0], xmax=history[1],
axes=False)
```

stock\_symbol

history

training\_window

verbose

strategy

number\_of\_states

draw\_points

---

# Every Single Copy of Sage Includes...

1. [ATLAS](#): Automatically Tuned Linear Algebra Software
2. [BLAS](#): Basic Fortran 77 linear algebra routines
3. [Bzip2](#): High-quality data compressor
4. [Cddlib](#): Double Description Method of Motzkin
5. [Common Lisp](#): Multiparadigm and general-purpose programming language
6. [CVXOPT](#): Convex optimization, linear programming, least squares, etc.
7. [Cython](#): C-Extensions for Python
8. [F2c](#): Converts Fortran 77 to C code
9. [Flint](#): Fast Library for Number Theory
10. [FpLLL](#): Euclidian lattice reduction
11. [FreeType](#): A Free, High-Quality, and Portable Font Engine
12. [G95](#): Open source Fortran 95 compiler
13. [GAP](#): Groups, Algorithms, Programming
14. [GD](#): Dynamic graphics generation tool
15. [Genus2reduction](#): Curve data computation
16. [Gfan](#): Gröbner fans and tropical varieties

17. [Givaro](#): C++ library for arithmetic and algebra
18. [GMP](#): GNU Multiple Precision Arithmetic Library
19. [GMP-ECM](#): Elliptic Curve Method for Integer Factorization
20. [GNU TLS](#): Secure networking
21. [GSL](#): Gnu Scientific Library
22. [JsMath](#): JavaScript implementation of LaTeX
23. [IML](#): Integer Matrix Library
24. [IPython](#): Interactive Python shell
25. [LAPACK](#): Fortran 77 linear algebra library
26. [Lcalc](#): L-functions calculator
27. [Libgcrypt](#): General purpose cryptographic library
28. [Libpgp-error](#): Common error values for GnuPG components
29. [Linbox](#): C++ linear algebra library
30. [M4RI](#): Linear Algebra over GF(2)
31. [Matplotlib](#): Python plotting library
32. [Maxima](#): computer algebra system
33. [Mercurial](#): Revision control system
34. [MoinMoin Wiki](#)
35. [MPFI](#): Multiple Precision Floating-point Interval library
36. [MPFR](#): C library for multiple-precision floating-point computations with correct rounding
37. [ECLib](#): Cremona's Programs for Elliptic curves
38. [NetworkX](#): Graph theory
39. [NTL](#): Number theory C++ library
40. [Numpy](#): Numerical linear algebra
41. [OpenCDK](#): Open Crypto Development Kit
42. [PALP](#): A Package for Analyzing Lattice Polytopes
43. [PARI/GP](#): Number theory calculator
44. [Pexpect](#): Pseudo-tty control for Python
45. [PNG](#): Bitmap image support
46. [PolyBoRi](#): Polynomials Over Boolean Rings
47. [PyCrypto](#): Python Cryptography Toolkit
48. [Python](#): Interpreted language
49. [Qd](#): Quad-double/Double-double Computation Package
50. [R](#): Statistical Computing
51. [Readline](#): Line-editing
52. [Rpy](#): Python interface to R



53. [Scipy](#): Python library for scientific computation
54. [Singular](#): fast commutative and noncommutative algebra
55. [Scons](#): Software construction tool
56. [SQLite](#): Relation database
57. [Sympow](#): L-function calculator
58. [Symmetrica](#): Representation theory
59. [SymPy](#): Python library for symbolic computation
60. [Tachyon](#): lightweight 3d ray tracer
61. [Termcap](#): Simplifies the process of writing portable text mode applications
62. [Twisted](#): Python networking library
63. [Weave](#): Tools for including C/C++ code within Python
64. [Zlib](#): Data compression library
65. [ZODB](#): Object-oriented database

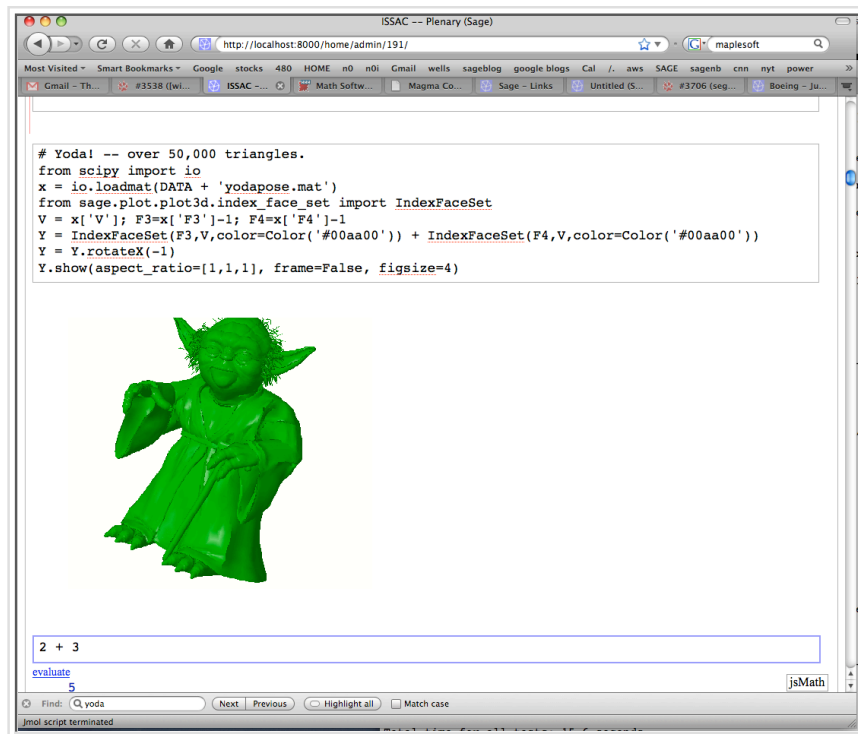
This is over **over five million** lines of source code. **You** can build this on almost any Linux or OS X box out there as follows:

```
@ tar xf sage-3.0.5.tar
@ cd sage-3.0.5
@ make
...
real    105m8.635s
SAGE build/upgrade complete!
```

```
@ ./sage
```

```
-----
| SAGE Version 3.0.5, Release Date: 2008-07-11
| Type notebook() for the GUI, and license() for infor
-----
```

```
sage: notebook()
```



The screenshot shows a web browser window titled "ISSAC -- Plenary (Sage)" with the URL "http://localhost:8000/home/admin/191/". The browser's address bar and tabs are visible. The main content area displays a SageMath code cell with the following Python code:

```
# Yoda! -- over 50,000 triangles.
from scipy import io
x = io.loadmat(DATA + 'yodapose.mat')
from sage.plot.plot3d.index_face_set import IndexFaceSet
V = x['V']; F3=x['F3']-1; F4=x['F4']-1
Y = IndexFaceSet(F3,V,color=Color('#00aa00')) + IndexFaceSet(F4,V,color=Color('#00aa00'))
Y = Y.rotateX(-1)
Y.show(aspect_ratio=[1,1,1], frame=False, figsize=4)
```

Below the code, a 3D plot of Yoda's head is displayed, rendered in a bright green color. The plot is centered on a white background. At the bottom of the code cell, there is an input field containing the expression "2 + 3", an "evaluate" button, and the result "5". The bottom status bar of the SageMath interface shows "Find: yoda", "Next", "Previous", "Highlight all", "Match case", and "jsMath".

---

# Debianization of Sage

I want `apt-get install sagemath!`

From: Tim Abbot

Date: Tue, Jul 22, 2008 at 4:44 AM

I figured I'd give everyone an update on how things are going with the Sage packages. I believe (but am not certain) that all of the Sage dependencies that I want to get into Lenny will make it, though I'm still waiting on final review for 5 of them that had copyright problems in the past.

On the other hand, Debian is freezing everything starting in around 5 or 6 days. So, I need to have a presentable Sage package in the very near future.

There are currently a few showstopper problems:

- \* [M4Ri problems...]
- \* [Linbox problems...]

-Tim Abbott

Sage has got tons of math software into Debian that might *never* otherwise get in.

---

# Questions?

Ask me anything!

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

[x](#)

```
f = integrate(sin(x^2)); show(f)
```



---

# Who is Sage?

**Sage is an international project with over 100  
developers across the globe!**



---

# Sage Days Workshops

1. February 2006: Sage Days 1 at UC San Diego
2. October 2006: Sage Days 2 at University of Washington
3. February 2007: Sage Days 3 at UC Los Angeles (IPAM)
4. June 2007: Sage Days 4 at University of Washington
5. October 2007: Sage Days 5 at the Clay Mathematics Institute
6. November 2007: Sage Days 6 at Bristol, UK
7. February 2008: Sage Days 7 at UC Los Angeles (IPAM)
8. March 2008: Sage Days 8 at Austin, Texas (Enthought)
9. June 2008: Sage Days 8.5 at University of Washington
10. August 2008: Sage Days 9 in Vancouver (SFU)
11. October 2008: Sage Days 10 in Nancy, France (INRIA)



12. November 2008: Sage Days 11 in Austin, Texas (UT Austin)
13. January 2009: Sage Days 12 in San Diego, CA
14. March 2009: Sage Days 13 in Athens, Georgia (UGA)



---

# Funding

We have received substantial funding from NSF, Microsoft, Google, DoD, several universities and institutes.



**There are several funded summer employees right now:**

1. Elliott Brossard -- (funded by NASA/Sage Foundation) interactive plotting, calc 101
2. Timothy Clemans -- (funded by Sage Foundation) Sage notebook
3. Gary Furnish -- (funded by Google/NSF) symbolic

### Calculus

4. Chris Gorecki -- (funded by Microsoft) port of Sage to Windows
5. Mike Hansen -- (funded by Google) combinatorial species
6. Emily Kirkman -- (funded by Google) graph theory, 2d graphics
7. Robert Miller -- (funded by Google) backtracking algorithms
8. Brett Nakashima -- (funded by VIGRE/NSF) quantitative finance
9. Yi Qiang -- (funded by Google) distributed Sage
10. Chris Swierczewski -- (funded by Glenn Tarbox) quantitative finance
11. Igor Tolkov -- (funded by Google Summer of Code) improving @interact

## Fulltime Employees

1. Michael Abshoff -- (funded by Microsoft and DoD); porting sage to Solaris and Windows; the Sage release manager
2. Clement Pernet -- (funded by NSF); The Sage Postdoc; lead Linbox developer

Also, *many* researchers use Sage to support their research and

implement new algorithms, and they contribute their code to Sage.

---

# Python



Instead of using a custom language like all other major math software systems, Sage uses the mainstream object oriented programming language Python, which is considered one of the worlds top 6 most popular languages.

- "Python is fast enough for our site and allows us to *produce maintainable features in record times*, with a minimum of developers," said Cuong Do, Software Architect, [YouTube.com](#).
- "Google has made no secret of the fact they use Python a lot for a number of internal projects. Even knowing that, once *I was an employee, I was amazed at how much Python code there actually is in the Google source code system.*", said Guido van Rossum, [Google](#), creator of Python.
- "Python is everywhere at ILM. It's used to extend the capabilities of our applications, as well as providing the glue between them. Every CG image we create has involved Python somewhere in the process," said Philip Peterson, Principal Engineer, Research & Development, [Industrial Light & Magic](#).

---

# Cython: Compiled Python



1. In order for Sage to be very fast and make reuse of existing C/C++ libraries easy, we use Cython, which is a Python-to-C compiler and language for writing fast compiled extensions to Python.
2. Cython is a language that makes writing C extensions for the Python language as easy as Python itself.
3. The Cython language is very close to the Python language, but Cython additionally supports calling C functions and declaring C types on variables and class attributes. This allows the compiler to generate very efficient C code from Cython code.
4. This makes Cython the ideal language for wrapping for external C libraries, and for fast C modules that speed up the execution of Python code.

```
def mymean(n):  
    s = float(0)  
    for m in range(n):  
        s += m  
    return float(s) / n
```

```
time mymean(10^6)
```

```
%cython
def mymean2(int n):
    cdef double s = 0
    cdef int m
    for m from 0 <= m < n:
        s += m
    return s / n
```

```
time mymean2(10^6)
```

```
timeit('mymean2(10^6)')
```

```
0.16/(1.19*10^(-3))
```

---

# Sage Interfaces

Sage has pseudo-tty based interfaces to everything that make it is possible to make extensive use of Maple, Mathematica, Magma, Matlab, GAP, Maxima, Singular, PARI, and many other systems from Sage.

```
x = var('x')
f = sin(x^2) + sqrt(3)/pi^e
show(f)
```

$$\sin(x^2) + \frac{\sqrt{3}}{\pi^e}$$

```
m = mathematica(f) # requires mathematica!!
m
```

```
Sqrt[3]/Pi^E + Sin[x^2]
```

Note: It's officially a license violation to use Mathematica from Sage via the notebook, since the EULA for Mathematica explicitly disallows any use of Mathematica via http.

```
os.system('ps ax |grep Mathematica.app')
1749 s006 S+ 0:00.01 sh -c ps ax |grep Mathematica.app
1751 s006 S+ 0:00.00 grep Mathematica.app
1747 s009 S+ 0:00.05
/Applications/Mathematica.app/Contents/MacOS/MathKernel
0
```

```
type(m)
```

```
<class 'sage.interfaces.mathematica.MathematicaElement'>
```

```
m.name()
```



```
'sage0'
```

```
m.Integrate(x)
```

```
(Sqrt[3]*x)/Pi^E + Sqrt[Pi/2]*FresnelS[Sqrt[2/Pi]*x]
```

```
# there is a bug in the maple interface where it never works
# at Simon Fraser University! I haven't been able to debug it
# until this workshop, since I haven't been here in a while.
m = maple(f); show(m)
```

```
sage3
```

```
m.integrate(x)
```

```
m = maxima(f); show(m)
```

$$\sin x^2 + \frac{\sqrt{3}}{\pi^e}$$

```
show(m.integrate())
```

$$\sqrt{\pi} \left( \left( \sqrt{2i + \sqrt{2}} \right) \operatorname{erf} \left( \frac{(\sqrt{2i + \sqrt{2}}) x}{2} \right) + \left( \sqrt{2i - \sqrt{2}} \right) \operatorname{erf} \left( \frac{(\sqrt{2i - \sqrt{2}}) x}{2} \right) \right) + \frac{\sqrt{3} x}{\pi^e}$$

```
r = octave('rand(4)'); r # I do not have matlab on laptop, but it's
the same
```

```
0.558769 0.261113 0.177883 0.989534
0.604335 0.0213031 0.908614 0.970141
0.206653 0.744935 0.0509978 0.563303
0.329074 0.948453 0.296885 0.691544
```

```
r.eig()
```

```
2.14615
-0.84845
0.146992
-0.12208
```

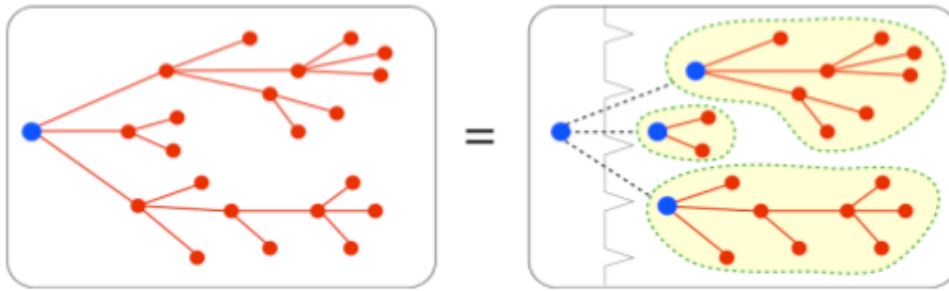
---

# Questions?

Ask me anything!

---

# Combinatorics



1. Sage is very strong in algebraic combinatorics. **The MuPAD-Combinat project** -- founded by Nicolas Thiery and Florent Hivert -- has officially decided to switch over to Sage, and have already ported much of their code over. Mike Hansen has done a massive amount of work on this.
2. Mike Hansen is porting Aldor-Combinat to Sage.
3. Sage has a **massive amount of graph theory code**, including the only open source implementation of computation of automorphism groups (Robert Miller's), a large database (Jason Grout and Emily Kirkman), and much much more.
4. Sage includes Gap, Gfan, and NetworkX.

```
G = graphs.CubeGraph(4); G
```

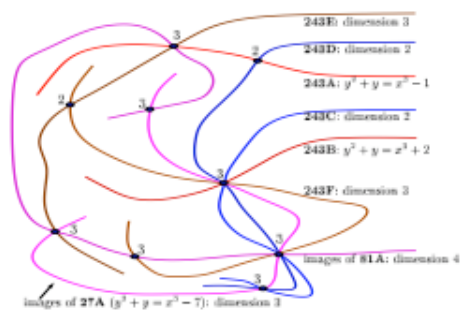
```
G.automorphism_group()
```

```
show(G)
```

```
G.plot3d(spin=True)
```

---

# Number Theory



1. Sage has a wide range of number theory, including algebraic number fields, modular forms, special functions, way more

elliptic curve functionality than any other open source program, and many elliptic and hyperelliptic curve functions that are available in no other programs.

2. Sage tightly builds on PARI, NTL, lcalc, Dokchiter, all of Cremona's code (both for curves and modular symbols), sympow, genus2reduction, and other number theory programs and libraries.

```
show(factor(29308203480928340982340820934820983402348))
```

```
E = EllipticCurve('5077a')
show(E)
print "Integral Points:"
print E.integral_points() # will be in sage-3.1 (thanks Nagel, Mardaus, and
Cremona!!)
plot(E, thickness=3, rgbcolor='orange').show()
```

---

# Exact Linear Algebra

1. Sage has optimized exact dense and sparse linear algebra over the rationals, integers, finite fields (especially  $\text{GF}(2)$ ).
2. Sage contains a large amount of new linear algebra code tightly integrated with Linbox, IML, NTL, Numpy, and ATLAS.

```
a = random_matrix(ZZ,200, x=-2^16,y=2^16) # 16-bit entries
set_verbose(2)
s = cputime()
d = a.determinant()      # Uses Linbox, ATLAS, IML, and new Sage code.
print d
print "time: ", cputime(s)
set_verbose(0)
```

```
magma.quit()
#b = magma('MatrixAlgebra(IntegerRing(),300)!Random(-2^16, 2^16) : i in [1..200^2]')
b = magma(a)
t = magma.cputime()
d2 = b.Determinant()
print 'Time: ', magma.cputime(t)
```

```
d == d2
```

```
b = mathematica(a)
```

```
mathematica.eval('Timing[Det[%s];]'%b.name())
```

---

# Cryptography

1. Sage is well suited for cryptography research because of its exact linear algebra (Linbox, M4RI), commutative algebra (PolyBori), support for crypto protocols (PyCrypto), number theory, and general extendability.

```
# Let's crack a small-scale AES variant!
```

```
sr = mq.SR(2,1,1,4, gf2=True); sr
```

```
F, s = sr.polynomial_system()
```

```
F, s
```



```
show(F.groebner_basis())
```

---

# Numerical Computation and Statistics

# 1. Sage includes R (and rpy), scipy.stats, and some new code for quantative finance.

```
import scipy.stats
scipy.stats.std([1..1000])
```

```
finance.TimeSeries([1..1000]).standard_deviation()
```

```
v = finance.TimeSeries(10^6).randomize('normal')
v.plot_histogram()
```

```
v.sums().plot()
```

---

# Get Your Code Into Sage!

1. Sage has an *extremely open and friendly* development model.
2. Every single patch that goes into the Sage library is *formally refereed*, like is done with a journal. Doctests and comments must be up to snuff.
3. All bugs are **publicly listed**, the referee process is public, the latest version of Sage is available. In fact, you can browse the developers home directories from the web here!  
<http://sage.math.washington.edu/home>

---

# Questions?

Ask me anything!

```
R.<x,y,z> = QQ[ ]
```

```
f = (x+y+z+1)^20; f
```

WARNING: Output truncated!  
[full output.txt](#)

```
x^20 + 20*x^19*y + 190*x^18*y^2 + 1140*x^17*y^3 + 4845*x^16*y^4 +  
15504*x^15*y^5 + 38760*x^14*y^6 + 77520*x^13*y^7 + 125970*x^12*y^8 +  
167960*x^11*y^9 + 184756*x^10*y^10 + 167960*x^9*y^11 +  
125970*x^8*y^12 + 77520*x^7*y^13 + 38760*x^6*y^14 + 15504*x^5*y^15 +  
4845*x^4*y^16 + 1140*x^3*y^17 + 190*x^2*y^18 + 20*x*y^19 + y^20 +  
20*x^19*z + 380*x^18*y*z + 3420*x^17*y^2*z + 19380*x^16*y^3*z +
```

```

77520*x^15*y^4*z + 232560*x^14*y^5*z + 542640*x^13*y^6*z +
1007760*x^12*y^7*z + 1511640*x^11*y^8*z + 1847560*x^10*y^9*z +
1847560*x^9*y^10*z + 1511640*x^8*y^11*z + 1007760*x^7*y^12*z +
542640*x^6*y^13*z + 232560*x^5*y^14*z + 77520*x^4*y^15*z +
19380*x^3*y^16*z + 3420*x^2*y^17*z + 380*x*y^18*z + 20*y^19*z +
190*x^18*z^2 + 3420*x^17*y*z^2 + 29070*x^16*y^2*z^2 +
155040*x^15*y^3*z^2 + 581400*x^14*y^4*z^2 + 1627920*x^13*y^5*z^2 +
3527160*x^12*y^6*z^2 + 6046560*x^11*y^7*z^2 + 8314020*x^10*y^8*z^2 +
9237800*x^9*y^9*z^2 + 8314020*x^8*y^10*z^2 + 6046560*x^7*y^11*z^2 +
3527160*x^6*y^12*z^2 + 1627920*x^5*y^13*z^2 + 581400*x^4*y^14*z^2 +
155040*x^3*y^15*z^2 + 29070*x^2*y^16*z^2 + 3420*x*y^17*z^2 +
190*y^18*z^2 + 1140*x^17*z^3 + 19380*x^16*y*z^3 +
155040*x^15*y^2*z^3 + 775200*x^14*y^3*z^3 + 2713200*x^13*y^4*z^3 +
7054320*x^12*y^5*z^3 + 14108640*x^11*y^6*z^3 + 22170720*x^10*y^7*z^3
+ 27713400*x^9*y^8*z^3 + 27713400*x^8*y^9*z^3 +
22170720*x^7*y^10*z^3 + 14108640*x^6*y^11*z^3 + 7054320*x^5*y^12*z^3
+ 2713200*x^4*y^13*z^3 + 775200*x^3*y^14*z^3 + 155040*x^2*y^15*z^3 +
19380*x*y^16*z^3 + 1140*y^17*z^3 + 4845*x^16*z^4 + 77520*x^15*y*z^4
+ 581400*x^14*y^2*z^4 + 2713200*x^13*y^3*z^4 + 8817900*x^12*y^4*z^4
+ 21162960*x^11*y^5*z^4 + 38798760*x^10*y^6*z^4 +
55426800*x^9*y^7*z^4 + 62355150*x^8*y^8*z^4 + 55426800*x^7*y^9*z^4 +
38798760*x^6*y^10*z^4 + 21162960*x^5*y^11*z^4 + 8817900*x^4*y^12*z^4
+ 2713200*x^3*y^13*z^4 + 581400*x^2*y^14*z^4 + 77520*x*y^15*z^4 +
4845*y^16*z^4 + 15504*x^15*z^5 + 232560*x^14*y*z^5 +
1627920*x^13*y^2*z^5 + 7054320*x^12*y^3*z^5 + 21162960*x^11*y^4*z^5
+ 46558512*x^10*y^5*z^5 + 77597520*x^9*y^6*z^5 +
99768240*x^8*y^7*z^5 + 99768240*x^7*y^8*z^5 + 77597520*x^6*y^9*z^5 +
46558512*x^5*y^10*z^5 + 21162960*x^4*y^11*z^5 + 7054320*x^3*y^12*z^5
+ 1627920*x^2*y^13*z^5 + 232560*x*y^14*z^5 + 15504*y^15*z^5 +
38760*x^14*z^6 + 542640*x^13*y*z^6 + 3527160*x^12*y^2*z^6 +
14108640*x^11*y^3*z^6 + 38798760*x^10*y^4*z^6 + 77597520*x^9*y^5*z^6
+ 116396280*x^8*y^6*z^6 + 133024320*x^7*y^7*z^6 +
116396280*x^6*y^8*z^6 + 77597520*x^5*y^9*z^6 + 38798760*x^4*y^10*z^6
+ 14108640*x^3*y^11*z^6 + 3527160*x^2*y^12*z^6 + 542640*x*y^13*z^6 +
38760*y^14*z^6 + 77520*x^13*z^7 + 1007760*x^12*y*z^7 +
6046560*x^11*y^2*z^7 + 22170720*x^10*y^3*z^7 + 55426800*x^9*y^4*z^7
+ 99768240*x^8*y^5*z^7 + 133024320*x^7*y^6*z^7 +
133024320*x^6*y^7*z^7 + 99768240*x^5*y^8*z^7 + 55426800*x^4*y^9*z^7
+ 22170720*x^3*y^10*z^7 + 6046560*x^2*y^11*z^7 + 1007760*x*y^12*z^7
+ 77520*y^13*z^7 + 125970*x^12*z^8 + 1511640*x^11*y*z^8 +
8314020*x^10*y^2*z^8 + 27713400*x^9*y^3*z^8 + 62355150*x^8*y^4*z^8 +
99768240*x^7*y^5*z^8 + 116396280*x^6*y^6*z^8 + 99768240*x^5*y^7*z^8
+ 62355150*x^4*y^8*z^8 + 27713400*x^3*y^9*z^8 + 8314020*x^2*y^10*z^8
+ 1511640*x*y^11*z^8 + 125970*y^12*z^8 + 167960*x^11*z^9 +
1847560*x^10*y*z^9 + 9237800*x^9*y^2*z^9 + 27713400*x^8*y^3*z^9 +
55426800*x^7*y^4*z^9 + 77597520*x^6*y^5*z^9 + 77597520*x^5*y^6*z^9 +
55426800*x^4*y^7*z^9 + 27713400*x^3*y^8*z^9 + 9237800*x^2*y^9*z^9 +
1847560*x*y^10*z^9 + 167960*y^11*z^9 + 184756*x^10*z^10 +
1847560*x^9*y*z^10 + 8314020*x^8*y^2*z^10 + 22170720*x^7*y^3*z^10 +
38798760*x^6*y^4*z^10 + 46558512*x^5*y^5*z^10 +
38798760*x^4*y^6*z^10 + 22170720*x^3*y^7*z^10 + 8314020*x^2*y^8*z^10
+ 1847560*x*y^9*z^10 + 184756*y^10*z^10 + 167960*x^9*z^11 +
1511640*x^8*y*z^11 + 6046560*x^7*y^2*z^11 + 14108640*x^6*y^3*z^11 +
21162960*x^5*y^4*z^11 + 21162960*x^4*y^5*z^11 +

```

```

14108640*x^3*y^6*z^11 + 6046560*x^2*y^7*z^11 + 1511640*x*y^8*z^11 +
167960*y^9*z^11 + 125970*x^8*z^12 + 1007760*x^7*y*z^12 +
3527160*x^6*y^2*z^12 + 7054320*x^5*y^3*z^12 + 8817900*x^4*y^4*z^12 +
7054320*x^3*y^5*z^12 + 3527160*x^2*y^6*z^12 + 1007760*x*y^7*z^12 +
125970*y^8*z^12 + 77520*x^7*z^13 + 542640*x^6*y*z^13 +
1627920*x^5*y^2*z^13 + 2713200*x^4*y^3*z^13 + 2713200*x^3*y^4*z^13 +
1627920*x^2*y^5*z^13 + 542640*x*y^6*z^13 + 77520*y^7*z^13 +
38760*x^6*z^14 + 232560*x^5*y*z^14 + 581400*x^4*y^2*z^14 +
775200*x^3*y^3*z^14 + 581400*x^2*y^4*z^14 + 232560*x*y^5*z^14 +
38760*y^6*z^14 + 15504*x^5*z^15 + 77520*x^4*y*z^15 +
155040*x^3*y^2*z^15 + 155040*x^2*y^3*z^15 + 77520*x*y^4*z^15 +
15504*y^5*z^15 + 4845*x^4*z^16 + 19380*x^3*y*z^16 +
29070*x^2*y^2*z^16 + 19380*x*y^3*z^16 + 4845*y^4*z^16 +
1140*x^3*z^17 + 3420*x^2*y*z^17 + 3420*x*y^2*z^17 + 1140*y^3*z^17 +
190*x^2*z^18 + 380*x*y*z^18 + 190*y^2*z^18 + 20*x*z^19 + 20*y*z^19 +
z^20 + 20*x^19 + 380*x^18*y + 3420*x^17*y^2 + 19380*x^16*y^3 +
77520*x^15*y^4 + 232560*x^14*y^5 + 542640*x^13*y^6 +
1007760*x^12*y^7 + 1511640*x^11*y^8 + 1847560*x^10*y^9 +
1847560*x^9*y^10 + 1511640*x^8*y^11 + 1007760*x^7*y^12 +
542640*x^6*y^13 + 232560*x^5*y^14 + 77520*x^4*y^15 + 19380*x^3*y^16
+ 3420*x^2*y^17 + 380*x*y^18 + 20*y^19 + 380*x^18*z + 6840*x^17*y*z
+ 58140*x^16*y^2*z + 310080*x^15*y^3*z + 1162800*x^14*y^4*z +
3255840*x^13*y^5*z + 7054320*x^12*y^6*z + 12093120*x^11*y^7*z +
16628040*x^10*y^8*z + 18475600*x^9*y^9*z + 16628040*x^8*y^10*z +
12093120*x^7*y^11*z + 7054320*x^6*y^12*z + 3255840*x^5*y^13*z +
1162800*x^4*y^14*z + 310080*x^3*y^15*z + 58140*x^2*y^16*z +
6840*x*y^17*z + 380*y^18*z + 3420*x^17*z^2 + 58140*x^16*y*z^2 +
465120*x^15*y^2*z^2 + 2325600*x^14*y^3*z^2 + 8139600*x^13*y^4*z^2 +
21162960*x^12*y^5*z^2 + 42325920*x^11*y^6*z^2 +
66512160*x^10*y^7*z^2 + 83140200*x^9*y^8*z^2 + 83140200*x^8*y^9*z^2
+ 66512160*x^7*y^10*z^2 + 42325920*x^6*y^11*z^2 +
21162960*x^5*y^12*z^2 + 8139600*x^4*y^13*z^2 + 2325600*x^3*y^14*z^2
+ 465120*x^2*y^15*z^2 + 58140*x*y^16*z^2 + 3420*y^17*z^2 +
19380*x^16*z^3 + 310080*x^15*y*z^3 + 2325600*x^14*y^2*z^3 +
10852800*x^13*y^3*z^3 + 35271600*x^12*y^4*z^3 +
84651840*x^11*y^5*z^3 + 155195040*x^10*y^6*z^3 +
221707200*x^9*y^7*z^3 + 249420600*x^8*y^8*z^3 +
221707200*x^7*y^9*z^3 + 155195040*x^6*y^10*z^3 +
84651840*x^5*y^11*z^3 + 35271600*x^4*y^12*z^3 +
10852800*x^3*y^13*z^3 + 2325600*x^2*y^14*z^3 + 310080*x*y^15*z^3 +
19380*y^16*z^3 + 77520*x^15*z^4 + 1162800*x^14*y*z^4 +
8139600*x^13*y^2*z^4 + 35271600*x^12*y^3*z^4 +
105814800*x^11*y^4*z^4 + 232792560*x^10*y^5*z^4 +
387987600*x^9*y^6*z^4 + 498841200*x^8*y^7*z^4 +
498841200*x^7*y^8*z^4 + 387987600*x^6*y^9*z^4 +
232792560*x^5*y^10*z^4 + 105814800*x^4*y^11*z^4 +
35271600*x^3*y^12*z^4 + 8139600*x^2*y^13*z^4 + 1162800*x*y^14*z^4 +
77520*y^15*z^4 + 232560*x^14*z^5 + 3255840*x^13*y*z^5 +
21162960*x^12*y^2*z^5 + 84651840*x^11*y^3*z^5 +
232792560*x^10*y^4*z^5 + 465585120*x^9*y^5*z^5 +
698377680*x^8*y^6*z^5 + 798145920*x^7*y^7*z^5 +
698377680*x^6*y^8*z^5 + 465585120*x^5*y^9*z^5 +
232792560*x^4*y^10*z^5 + 84651840*x^3*y^11*z^5 +
21162960*x^2*y^12*z^5 + 3255840*x*y^13*z^5 + 232560*y^14*z^5 +

```

```

542640*x^13*z^6 + 7054320*x^12*y*z^6 + 42325920*x^11*y^2*z^6 +
155195040*x^10*y^3*z^6 + 387987600*x^9*y^4*z^6 +
698377680*x^8*y^5*z^6 + 931170240*x^7*y^6*z^6 +
931170240*x^6*y^7*z^6 + 698377680*x^5*y^8*z^6 +
387987600*x^4*y^9*z^6 + 155195040*x^3*y^10*z^6 +
42325920*x^2*y^11*z^6 + 7054320*x*y^12*z^6 + 542640*y^13*z^6 +
1007760*x^12*z^7 + 12093120*x^11*y*z^7 + 66512160*x^10*y^2*z^7 +
221707200*x^9*y^3*z^7 + 498841200*x^8*y^4*z^7 +
798145920*x^7*y^5*z^7 + 931170240*x^6*y^6*z^7 +
798145920*x^5*y^7*z^7 + 498841200*x^4*y^8*z^7 +
221707200*x^3*y^9*z^7 + 66512160*x^2*y^10*z^7 + 12093120*x*y^11*z^7
+ 1007760*y^12*z^7 + 1511640*x^11*z^8 + 16628040*x^10*y*z^8 +
83140200*x^9*y^2*z^8 + 249420600*x^8*y^3*z^8 + 498841200*x^7*y^4*z^8
+ 698377680*x^6*y^5*z^8 + 698377680*x^5*y^6*z^8 +
498841200*x^4*y^7*z^8 + 249420600*x^3*y^8*z^8 + 83140200*x^2*y^9*z^8
+ 16628040*x*y^10*z^8 + 1511640*y^11*z^8 + 1847560*x^10*z^9 +
18475600*x^9*y*z^9 + 83140200*x^8*y^2*z^9 + 221707200*x^7*y^3*z^9 +
387987600*x^6*y^4*z^9 + 465585120*x^5*y^5*z^9 +
387987600*x^4*y^6*z^9 + 221707200*x^3*y^7*z^9 + 83140200*x^2*y^8*z^9
+ 18475600*x*y^9*z^9 + 1847560*y^10*z^9 + 1847560*x^9*z^10 +
16628040*x^8*y*z^10 + 66512160*x^7*y^2*z^10 + 155195040*x^6*y^3*z^10
+ 232792560*x^5*y^4*z^10 + 232792560*x^4*y^5*z^10 +
155195040*x^3*y^6*z^10 + 66512160*x^2*y^7*z^10 + 16628040*x*y^8*z^10
+ 1847560*y^9*z^10 + 1511640*x^8*z^11 + 12093120*x^7*y*z^11 +
42325920*x^6*y^2*z^11 + 84651840*x^5*y^3*z^11 +
105814800*x^4*y^4*z^11 + 84651840*x^3*y^5*z^11 +
42325920*x^2*y^6*z^11 + 12093120*x*y^7*z^11 + 1511640*y^8*z^11 +
1007760*x^7*z^12 + 7054320*x^6*y*z^12 + 21162960*x^5*y^2*z^12 +
35271600*x^4*y^3*z^12 + 35271600*x^3*y^4*z^12 +
21162960*x^2*y^5*z^12 + 7054320*x*y^6*z^12 + 1007760*y^7*z^12 +
542640*x^6*z^13 + 3255840*x^5*y*z^13 + 8139600*x^4*y^2*z^13 +
10852800*x^3*y^3*z^13 + 8139600*x^2*y^4*z^13 + 3255840*x*y^5*z^13 +
542640*y^6*z^13 + 232560*x^5*z^14 + 1162800*x^4*y*z^14 +
2325600*x^3*y^2*z^14 + 2325600*x^2*y^3*z^14 + 1162800*x*y^4*z^14 +
232560*y^5*z^14 + 77520*x^4*z^15 + 310080*x^3*y*z^15 +
465120*x^2*y^2*z^15 + 310080*x*y^3*z^15 + 77520*y^4*z^15 +
19380*x^3*z^16 + 58140*x^2*y*z^16 + 58140*x*y^2*z^16 +
19380*y^3*z^16 + 3420*x^2*z^17 + 6840*x*y*z^17 + 3420*y^2*z^17 +
380*x*z^18 + 380*y*z^18 + 20*z^19 + 190*x^18 + 3420*x^17*y +
29070*x^16*y^2 + 155040*x^15*y^3 + 581400*x^14*y^4 +
1627920*x^13*y^5 + 3527160*x^12*y^6 + 6046560*x^11*y^7 +
8314020*x^10*y^8 + 9237800*x^9*y^9 + 8314020*x^8*y^10 +
6046560*x^7*y^11 + 3527160*x^6*y^12 + 1627920*x^5*y^13 +
581400*x^4*y^14 + 155040*x^3*y^15 + 29070*x^2*y^16 + 3420*x*y^17 +
190*y^18 + 3420*x^17*z + 58140*x^16*y*z + 465120*x^15*y^2*z +
2325600*x^14*y^3*z + 8139600*x^13*y^4*z + 21162960*x^12*y^5*z +
42325920*x^11*y^6*z + 66512160*x^10*y^7*z + 83140200*x^9*y^8*z +
83140200*x^8*y^9*z + 66512160*x^7*y^10*z + 42325920*x^6*y^11*z +
21162960*x^5*y^12*z + 8139600*x^4*y^13*z + 2325600*x^3*y^14*z +
465120*x^2*y^15*z + 58140*x*y^16*z + 3420*y^17*z + 29070*x^16*z^2 +
465120*x^15*y*z^2 + 3488400*x^14*y^2*z^2 + 16279200*x^13*y^3*z^2 +
52907400*x^12*y^4*z^2 + 126977760*x^11*y^5*z^2 +
232792560*x^10*y^6*z^2 + 332560800*x^9*y^7*z^2 +
374130900*x^8*y^8*z^2 + 332560800*x^7*y^9*z^2 +

```

```

232792560*x^6*y^10*z^2 + 126977760*x^5*y^11*z^2 +
52907400*x^4*y^12*z^2 + 16279200*x^3*y^13*z^2 + 3488400*x^2*y^14*z^2
+ 465120*x*y^15*z^2 + 29070*y^16*z^2 + 155040*x^15*z^3 +
2325600*x^14*y*z^3 + 16279200*x^13*y^2*z^3 + 70543200*x^12*y^3*z^3 +
211629600*x^11*y^4*z^3 + 465585120*x^10*y^5*z^3 +
775975200*x^9*y^6*z^3 + 997682400*x^8*y^7*z^3 +
997682400*x^7*y^8*z^3 + 775975200*x^6*y^9*z^3 +
465585120*x^5*y^10*z^3 + 211629600*x^4*y^11*z^3 +
70543200*x^3*y^12*z^3 + 16279200*x^2*y^13*z^3 + 2325600*x*y^14*z^3 +
155040*y^15*z^3 + 581400*x^14*z^4 + 8139600*x^13*y*z^4 +
52907400*x^12*y^2*z^4 + 211629600*x^11*y^3*z^4 +
581981400*x^10*y^4*z^4 + 1163962800*x^9*y^5*z^4 +
1745944200*x^8*y^6*z^4 + 1995364800*x^7*y^7*z^4 +
1745944200*x^6*y^8*z^4 + 1163962800*x^5*y^9*z^4 +
581981400*x^4*y^10*z^4 + 211629600*x^3*y^11*z^4 +
52907400*x^2*y^12*z^4 + 8139600*x*y^13*z^4 + 581400*y^14*z^4 +
1627920*x^13*z^5 + 21162960*x^12*y*z^5 + 126977760*x^11*y^2*z^5 +
465585120*x^10*y^3*z^5 + 1163962800*x^9*y^4*z^5 +
2095133040*x^8*y^5*z^5 + 2793510720*x^7*y^6*z^5 +
2793510720*x^6*y^7*z^5 + 2095133040*x^5*y^8*z^5 +
1163962800*x^4*y^9*z^5 + 465585120*x^3*y^10*z^5 +
126977760*x^2*y^11*z^5 + 21162960*x*y^12*z^5 + 1627920*y^13*z^5 +
3527160*x^12*z^6 + 42325920*x^11*y*z^6 + 232792560*x^10*y^2*z^6 +
775975200*x^9*y^3*z^6 + 1745944200*x^8*y^4*z^6 +
2793510720*x^7*y^5*z^6 + 3259095840*x^6*y^6*z^6 +
2793510720*x^5*y^7*z^6 + 1745944200*x^4*y^8*z^6 +
775975200*x^3*y^9*z^6 + 232792560*x^2*y^10*z^6 + 42325920*x*y^11*z^6
+ 3527160*y^12*z^6 + 6046560*x^11*z^7 + 66512160*x^10*y*z^7 +
332560800*x^9*y^2*z^7 + 997682400*x^8*y^3*z^7 +
1995364800*x^7*y^4*z^7 + 2793510720*x^6*y^5*z^7 +
2793510720*x^5*y^6*z^7 + 1995364800*x^4*y^7*z^7 +
997682400*x^3*y^8*z^7 + 332560800*x^2*y^9*z^7 + 66512160*x*y^10*z^7
+ 6046560*y^11*z^7 + 8314020*x^10*z^8 + 83140200*x^9*y*z^8 +
374130900*x^8*y^2*z^8 + 997682400*x^7*y^3*z^8 +
1745944200*x^6*y^4*z^8 + 2095133040*x^5*y^5*z^8 +
1745944200*x^4*y^6*z^8 + 997682400*x^3*y^7*z^8 +
374130900*x^2*y^8*z^8 + 83140200*x*y^9*z^8 + 8314020*y^10*z^8 +
9237800*x^9*z^9 + 83140200*x^8*y*z^9 + 332560800*x^7*y^2*z^9 +
775975200*x^6*y^3*z^9 + 1163962800*x^5*y^4*z^9 +
1163962800*x^4*y^5*z^9 + 775975200*x^3*y^6*z^9 +
332560800*x^2*y^7*z^9 + 83140200*x*y^8*z^9 + 9237800*y^9*z^9 +
8314020*x^8*z^10 + 66512160*x^7*y*z^10 + 232792560*x^6*y^2*z^10 +
465585120*x^5*y^3*z^10 + 581981400*x^4*y^4*z^10 +
465585120*x^3*y^5*z^10 + 232792560*x^2*y^6*z^10 +
66512160*x*y^7*z^10 + 8314020*y^8*z^10 + 6046560*x^7*z^11 +
42325920*x^6*y*z^11 + 126977760*x^5*y^2*z^11 +
211629600*x^4*y^3*z^11 + 211629600*x^3*y^4*z^11 +
126977760*x^2*y^5*z^11 + 42325920*x*y^6*z^11 + 6046560*y^7*z^11 +
3527160*x^6*z^12 + 21162960*x^5*y*z^12 + 52907400*x^4*y^2*z^12 +
70543200*x^3*y^3*z^12 + 52907400*x^2*y^4*z^12 + 21162960*x*y^5*z^12
+ 3527160*y^6*z^12 + 1627920*x^5*z^13 + 8139600*x^4*y*z^13 +
16279200*x^3*y^2*z^13 + 16279200*x^2*y^3*z^13 + 8139600*x*y^4*z^13 +
1627920*y^5*z^13 + 581400*x^4*z^14 + 2325600*x^3*y*z^14 +
3488400*x^2*y^2*z^14 + 2325600*x*y^3*z^14 + 581400*y^4*z^14 +

```



```

155040*x^3*z^15 + 465120*x^2*y*z^15 + 465120*x*y^2*z^15 +
155040*y^3*z^15 + 29070*x^2*z^16 + 58140*x*y*z^16 + 29070*y^2*z^16 +
3420*x*z^17 + 3420*y*z^17 + 190*z^18 + 1140*x^17 + 19380*x^16*y +
155040*x^15*y^2 + 775200*x^14*y^3 + 2713200*x^13*y^4 +
7054320*x^12*y^5 + 14108640*x^11*y^6 + 22170720*x^10*y^7 +
27713400*x^9*y^8 + 27713400*x^8*y^9 + 22170720*x^7*y^10 +
14108640*x^6*y^11 + 7054320*x^5*y^12 + 2713200*x^4*y^13 +
775200*x^3*y^14 + 155040*x^2*y^15 + 19380*x*y^16 + 1140*y^17 +
19380*x^16*z + 310080*x^15*y*z + 2325600*x^14*y^2*z +
10852800*x^13*y^3*z + 35271600*x^12*y^4*z + 84651840*x^11*y^5*z +
155195040*x^10*y^6*z + 221707200*x^9*y^7*z + 249420600*x^8*y^8*z +
221707200*x^7*y^9*z + 155195040*x^6*y^10*z + 84651840*x^5*y^11*z +
35271600*x^4*y^12*z + 10852800*x^3*y^13*z + 2325600*x^2*y^14*z +
310080*x*y^15*z + 19380*y^16*z + 155040*x^15*z^2 +
2325600*x^14*y*z^2 + 16279200*x^13*y^2*z^2 + 70543200*x^12*y^3*z^2 +
211629600*x^11*y^4*z^2 + 465585120*x^10*y^5*z^2 +
775975200*x^9*y^6*z^2 + 997682400*x^8*y^7*z^2 +
997682400*x^7*y^8*z^2 + 775975200*x^6*y^9*z^2 +
465585120*x^5*y^10*z^2 + 211629600*x^4*y^11*z^2 +
70543200*x^3*y^12*z^2 + 16279200*x^2*y^13*z^2 + 2325600*x*y^14*z^2 +
155040*y^15*z^2 + 775200*x^14*z^3 + 10852800*x^13*y*z^3 +
70543200*x^12*y^2*z^3 + 282172800*x^11*y^3*z^3 +
775975200*x^10*y^4*z^3 + 1551950400*x^9*y^5*z^3 +
2327925600*x^8*y^6*z^3 + 2660486400*x^7*y^7*z^3 +
2327925600*x^6*y^8*z^3 + 1551950400*x^5*y^9*z^3 +
775975200*x^4*y^10*z^3 + 282172800*x^3*y^11*z^3 +
70543200*x^2*y^12*z^3 + 10852800*x*y^13*z^3 + 775200*y^14*z^3 +
2713200*x^13*z^4 + 35271600*x^12*y*z^4 + 211629600*x^11*y^2*z^4 +
775975200*x^10*y^3*z^4 + 1939938000*x^9*y^4*z^4 +
3491888400*x^8*y^5*z^4 + 4655851200*x^7*y^6*z^4 +
4655851200*x^6*y^7*z^4 + 3491888400*x^5*y^8*z^4 +
1939938000*x^4*y^9*z^4 + 775975200*x^3*y^10*z^4 +
211629600*x^2*y^11*z^4 + 35271600*x*y^12*z^4 + 2713200*y^13*z^4 +
7054320*x^12*z^5 + 84651840*x^11*y*z^5 + 465585120*x^10*y^2*z^5 +
1551950400*x^9*y^3*z^5 + 3491888400*x^8*y^4*z^5 +
5587021440*x^7*y^5*z^5 + 6518191680*x^6*y^6*z^5 +
5587021440*x^5*y^7*z^5 + 3491888400*x^4*y^8*z^5 +
1551950400*x^3*y^9*z^5 + 465585120*x^2*y^10*z^5 +
84651840*x*y^11*z^5 + 7054320*y^12*z^5 + 14108640*x^11*z^6 +
155195040*x^10*y*z^6 + 775975200*x^9*y^2*z^6 +
2327925600*x^8*y^3*z^6 + 4655851200*x^7*y^4*z^6 +
6518191680*x^6*y^5*z^6 + 6518191680*x^5*y^6*z^6 +
4655851200*x^4*y^7*z^6 + 2327925600*x^3*y^8*z^6 +
775975200*x^2*y^9*z^6 + 155195040*x*y^10*z^6 + 14108640*y^11*z^6 +
22170720*x^10*z^7 + 221707200*x^9*y*z^7 + 997682400*x^8*y^2*z^7 +
2660486400*x^7*y^3*z^7 + 4655851200*x^6*y^4*z^7 +
5587021440*x^5*y^5*z^7 + 4655851200*x^4*y^6*z^7 +
2660486400*x^3*y^7*z^7 + 997682400*x^2*y^8*z^7 + 221707200*x*y^9*z^7
+ 22170720*y^10*z^7 + 27713400*x^9*z^8 + 249420600*x^8*y*z^8 +
997682400*x^7*y^2*z^8 + 2327925600*x^6*y^3*z^8 +
3491888400*x^5*y^4*z^8 +

```

...

```

2560*y^14*z + 1627920*x^13*z^2 + 21162960*x^12*y*z^2 +
126977760*x^11*y^2*z^2 + 465585120*x^10*y^3*z^2 +
1163962800*x^9*y^4*z^2 + 2095133040*x^8*y^5*z^2 +
2793510720*x^7*y^6*z^2 + 2793510720*x^6*y^7*z^2 +
2095133040*x^5*y^8*z^2 + 1163962800*x^4*y^9*z^2 +
465585120*x^3*y^10*z^2 + 126977760*x^2*y^11*z^2 +
21162960*x*y^12*z^2 + 1627920*y^13*z^2 + 7054320*x^12*z^3 +
84651840*x^11*y*z^3 + 465585120*x^10*y^2*z^3 +
1551950400*x^9*y^3*z^3 + 3491888400*x^8*y^4*z^3 +
5587021440*x^7*y^5*z^3 + 6518191680*x^6*y^6*z^3 +
5587021440*x^5*y^7*z^3 + 3491888400*x^4*y^8*z^3 +
1551950400*x^3*y^9*z^3 + 465585120*x^2*y^10*z^3 +
84651840*x*y^11*z^3 + 7054320*y^12*z^3 + 21162960*x^11*z^4 +
232792560*x^10*y*z^4 + 1163962800*x^9*y^2*z^4 +
3491888400*x^8*y^3*z^4 + 6983776800*x^7*y^4*z^4 +
9777287520*x^6*y^5*z^4 + 9777287520*x^5*y^6*z^4 +
6983776800*x^4*y^7*z^4 + 3491888400*x^3*y^8*z^4 +
1163962800*x^2*y^9*z^4 + 232792560*x*y^10*z^4 + 21162960*y^11*z^4 +
46558512*x^10*z^5 + 465585120*x^9*y*z^5 + 2095133040*x^8*y^2*z^5 +
5587021440*x^7*y^3*z^5 + 9777287520*x^6*y^4*z^5 +
11732745024*x^5*y^5*z^5 + 9777287520*x^4*y^6*z^5 +
5587021440*x^3*y^7*z^5 + 2095133040*x^2*y^8*z^5 +
465585120*x*y^9*z^5 + 46558512*y^10*z^5 + 77597520*x^9*z^6 +
698377680*x^8*y*z^6 + 2793510720*x^7*y^2*z^6 +
6518191680*x^6*y^3*z^6 + 9777287520*x^5*y^4*z^6 +
9777287520*x^4*y^5*z^6 + 6518191680*x^3*y^6*z^6 +
2793510720*x^2*y^7*z^6 + 698377680*x*y^8*z^6 + 77597520*y^9*z^6 +
99768240*x^8*z^7 + 798145920*x^7*y*z^7 + 2793510720*x^6*y^2*z^7 +
5587021440*x^5*y^3*z^7 + 6983776800*x^4*y^4*z^7 +
5587021440*x^3*y^5*z^7 + 2793510720*x^2*y^6*z^7 +
798145920*x*y^7*z^7 + 99768240*y^8*z^7 + 99768240*x^7*z^8 +
698377680*x^6*y*z^8 + 2095133040*x^5*y^2*z^8 +
3491888400*x^4*y^3*z^8 + 3491888400*x^3*y^4*z^8 +
2095133040*x^2*y^5*z^8 + 698377680*x*y^6*z^8 + 99768240*y^7*z^8 +
77597520*x^6*z^9 + 465585120*x^5*y*z^9 + 1163962800*x^4*y^2*z^9 +
1551950400*x^3*y^3*z^9 + 1163962800*x^2*y^4*z^9 +
465585120*x*y^5*z^9 + 77597520*y^6*z^9 + 46558512*x^5*z^10 +
232792560*x^4*y*z^10 + 465585120*x^3*y^2*z^10 +
465585120*x^2*y^3*z^10 + 232792560*x*y^4*z^10 + 46558512*y^5*z^10 +
21162960*x^4*z^11 + 84651840*x^3*y*z^11 + 126977760*x^2*y^2*z^11 +
84651840*x*y^3*z^11 + 21162960*y^4*z^11 + 7054320*x^3*z^12 +
21162960*x^2*y*z^12 + 21162960*x*y^2*z^12 + 7054320*y^3*z^12 +
1627920*x^2*z^13 + 3255840*x*y*z^13 + 1627920*y^2*z^13 +
232560*x*x*z^14 + 232560*y*z^14 + 15504*z^15 + 38760*x^14 +
542640*x^13*y + 3527160*x^12*y^2 + 14108640*x^11*y^3 +
38798760*x^10*y^4 + 77597520*x^9*y^5 + 116396280*x^8*y^6 +
133024320*x^7*y^7 + 116396280*x^6*y^8 + 77597520*x^5*y^9 +
38798760*x^4*y^10 + 14108640*x^3*y^11 + 3527160*x^2*y^12 +
542640*x*y^13 + 38760*y^14 + 542640*x^13*z + 7054320*x^12*y*z +
42325920*x^11*y^2*z + 155195040*x^10*y^3*z + 387987600*x^9*y^4*z +
698377680*x^8*y^5*z + 931170240*x^7*y^6*z + 931170240*x^6*y^7*z +
698377680*x^5*y^8*z + 387987600*x^4*y^9*z + 155195040*x^3*y^10*z +
42325920*x^2*y^11*z + 7054320*x*y^12*z + 542640*y^13*z +
3527160*x^12*z^2 + 42325920*x^11*y*z^2 + 232792560*x^10*y^2*z^2 +

```

```

775975200*x^9*y^3*z^2 + 1745944200*x^8*y^4*z^2 +
2793510720*x^7*y^5*z^2 + 3259095840*x^6*y^6*z^2 +
2793510720*x^5*y^7*z^2 + 1745944200*x^4*y^8*z^2 +
775975200*x^3*y^9*z^2 + 232792560*x^2*y^10*z^2 + 42325920*x*y^11*z^2
+ 3527160*y^12*z^2 + 14108640*x^11*z^3 + 155195040*x^10*y*z^3 +
775975200*x^9*y^2*z^3 + 2327925600*x^8*y^3*z^3 +
4655851200*x^7*y^4*z^3 + 6518191680*x^6*y^5*z^3 +
6518191680*x^5*y^6*z^3 + 4655851200*x^4*y^7*z^3 +
2327925600*x^3*y^8*z^3 + 775975200*x^2*y^9*z^3 +
155195040*x*y^10*z^3 + 14108640*y^11*z^3 + 38798760*x^10*z^4 +
387987600*x^9*y*z^4 + 1745944200*x^8*y^2*z^4 +
4655851200*x^7*y^3*z^4 + 8147739600*x^6*y^4*z^4 +
9777287520*x^5*y^5*z^4 + 8147739600*x^4*y^6*z^4 +
4655851200*x^3*y^7*z^4 + 1745944200*x^2*y^8*z^4 +
387987600*x*y^9*z^4 + 38798760*y^10*z^4 + 77597520*x^9*z^5 +
698377680*x^8*y*z^5 + 2793510720*x^7*y^2*z^5 +
6518191680*x^6*y^3*z^5 + 9777287520*x^5*y^4*z^5 +
9777287520*x^4*y^5*z^5 + 6518191680*x^3*y^6*z^5 +
2793510720*x^2*y^7*z^5 + 698377680*x*y^8*z^5 + 77597520*y^9*z^5 +
116396280*x^8*z^6 + 931170240*x^7*y*z^6 + 3259095840*x^6*y^2*z^6 +
6518191680*x^5*y^3*z^6 + 8147739600*x^4*y^4*z^6 +
6518191680*x^3*y^5*z^6 + 3259095840*x^2*y^6*z^6 +
931170240*x*y^7*z^6 + 116396280*y^8*z^6 + 133024320*x^7*z^7 +
931170240*x^6*y*z^7 + 2793510720*x^5*y^2*z^7 +
4655851200*x^4*y^3*z^7 + 4655851200*x^3*y^4*z^7 +
2793510720*x^2*y^5*z^7 + 931170240*x*y^6*z^7 + 133024320*y^7*z^7 +
116396280*x^6*z^8 + 698377680*x^5*y*z^8 + 1745944200*x^4*y^2*z^8 +
2327925600*x^3*y^3*z^8 + 1745944200*x^2*y^4*z^8 +
698377680*x*y^5*z^8 + 116396280*y^6*z^8 + 77597520*x^5*z^9 +
387987600*x^4*y*z^9 + 775975200*x^3*y^2*z^9 + 775975200*x^2*y^3*z^9
+ 387987600*x*y^4*z^9 + 77597520*y^5*z^9 + 38798760*x^4*z^10 +
155195040*x^3*y*z^10 + 232792560*x^2*y^2*z^10 + 155195040*x*y^3*z^10
+ 38798760*y^4*z^10 + 14108640*x^3*z^11 + 42325920*x^2*y*z^11 +
42325920*x*y^2*z^11 + 14108640*y^3*z^11 + 3527160*x^2*z^12 +
7054320*x*y*z^12 + 3527160*y^2*z^12 + 542640*x*z^13 + 542640*y*z^13
+ 38760*z^14 + 77520*x^13 + 1007760*x^12*y + 6046560*x^11*y^2 +
22170720*x^10*y^3 + 55426800*x^9*y^4 + 99768240*x^8*y^5 +
133024320*x^7*y^6 + 133024320*x^6*y^7 + 99768240*x^5*y^8 +
55426800*x^4*y^9 + 22170720*x^3*y^10 + 6046560*x^2*y^11 +
1007760*x*y^12 + 77520*y^13 + 1007760*x^12*z + 12093120*x^11*y*z +
66512160*x^10*y^2*z + 221707200*x^9*y^3*z + 498841200*x^8*y^4*z +
798145920*x^7*y^5*z + 931170240*x^6*y^6*z + 798145920*x^5*y^7*z +
498841200*x^4*y^8*z + 221707200*x^3*y^9*z + 66512160*x^2*y^10*z +
12093120*x*y^11*z + 1007760*y^12*z + 6046560*x^11*z^2 +
66512160*x^10*y*z^2 + 332560800*x^9*y^2*z^2 + 997682400*x^8*y^3*z^2
+ 1995364800*x^7*y^4*z^2 + 2793510720*x^6*y^5*z^2 +
2793510720*x^5*y^6*z^2 + 1995364800*x^4*y^7*z^2 +
997682400*x^3*y^8*z^2 + 332560800*x^2*y^9*z^2 + 66512160*x*y^10*z^2
+ 6046560*y^11*z^2 + 22170720*x^10*z^3 + 221707200*x^9*y*z^3 +
997682400*x^8*y^2*z^3 + 2660486400*x^7*y^3*z^3 +
4655851200*x^6*y^4*z^3 + 5587021440*x^5*y^5*z^3 +
4655851200*x^4*y^6*z^3 + 2660486400*x^3*y^7*z^3 +
997682400*x^2*y^8*z^3 + 221707200*x*y^9*z^3 + 22170720*y^10*z^3 +
55426800*x^9*z^4 + 498841200*x^8*y*z^4 + 1995364800*x^7*y^2*z^4 +

```

```

4655851200*x^6*y^3*z^4 + 6983776800*x^5*y^4*z^4 +
6983776800*x^4*y^5*z^4 + 4655851200*x^3*y^6*z^4 +
1995364800*x^2*y^7*z^4 + 498841200*x*y^8*z^4 + 55426800*y^9*z^4 +
99768240*x^8*z^5 + 798145920*x^7*y*z^5 + 2793510720*x^6*y^2*z^5 +
5587021440*x^5*y^3*z^5 + 6983776800*x^4*y^4*z^5 +
5587021440*x^3*y^5*z^5 + 2793510720*x^2*y^6*z^5 +
798145920*x*y^7*z^5 + 99768240*y^8*z^5 + 133024320*x^7*z^6 +
931170240*x^6*y*z^6 + 2793510720*x^5*y^2*z^6 +
4655851200*x^4*y^3*z^6 + 4655851200*x^3*y^4*z^6 +
2793510720*x^2*y^5*z^6 + 931170240*x*y^6*z^6 + 133024320*y^7*z^6 +
133024320*x^6*z^7 + 798145920*x^5*y*z^7 + 1995364800*x^4*y^2*z^7 +
2660486400*x^3*y^3*z^7 + 1995364800*x^2*y^4*z^7 +
798145920*x*y^5*z^7 + 133024320*y^6*z^7 + 99768240*x^5*z^8 +
498841200*x^4*y*z^8 + 997682400*x^3*y^2*z^8 + 997682400*x^2*y^3*z^8 +
498841200*x*y^4*z^8 + 99768240*y^5*z^8 + 55426800*x^4*z^9 +
221707200*x^3*y*z^9 + 332560800*x^2*y^2*z^9 + 221707200*x*y^3*z^9 +
55426800*y^4*z^9 + 22170720*x^3*z^10 + 66512160*x^2*y*z^10 +
66512160*x*y^2*z^10 + 22170720*y^3*z^10 + 6046560*x^2*z^11 +
12093120*x*y*z^11 + 6046560*y^2*z^11 + 1007760*x*z^12 +
1007760*y*z^12 + 77520*z^13 + 125970*x^12 + 1511640*x^11*y +
8314020*x^10*y^2 + 27713400*x^9*y^3 + 62355150*x^8*y^4 +
99768240*x^7*y^5 + 116396280*x^6*y^6 + 99768240*x^5*y^7 +
62355150*x^4*y^8 + 27713400*x^3*y^9 + 8314020*x^2*y^10 +
1511640*x*y^11 + 125970*y^12 + 1511640*x^11*z + 16628040*x^10*y*z +
83140200*x^9*y^2*z + 249420600*x^8*y^3*z + 498841200*x^7*y^4*z +
698377680*x^6*y^5*z + 698377680*x^5*y^6*z + 498841200*x^4*y^7*z +
249420600*x^3*y^8*z + 83140200*x^2*y^9*z + 16628040*x*y^10*z +
1511640*y^11*z + 8314020*x^10*z^2 + 83140200*x^9*y*z^2 +
374130900*x^8*y^2*z^2 + 997682400*x^7*y^3*z^2 +
1745944200*x^6*y^4*z^2 + 2095133040*x^5*y^5*z^2 +
1745944200*x^4*y^6*z^2 + 997682400*x^3*y^7*z^2 +
374130900*x^2*y^8*z^2 + 83140200*x*y^9*z^2 + 8314020*y^10*z^2 +
27713400*x^9*z^3 + 249420600*x^8*y*z^3 + 997682400*x^7*y^2*z^3 +
2327925600*x^6*y^3*z^3 + 3491888400*x^5*y^4*z^3 +
3491888400*x^4*y^5*z^3 + 2327925600*x^3*y^6*z^3 +
997682400*x^2*y^7*z^3 + 249420600*x*y^8*z^3 + 27713400*y^9*z^3 +
62355150*x^8*z^4 + 498841200*x^7*y*z^4 + 1745944200*x^6*y^2*z^4 +
3491888400*x^5*y^3*z^4 + 4364860500*x^4*y^4*z^4 +
3491888400*x^3*y^5*z^4 + 1745944200*x^2*y^6*z^4 +
498841200*x*y^7*z^4 + 62355150*y^8*z^4 + 99768240*x^7*z^5 +
698377680*x^6*y*z^5 + 2095133040*x^5*y^2*z^5 +
3491888400*x^4*y^3*z^5 + 3491888400*x^3*y^4*z^5 +
2095133040*x^2*y^5*z^5 + 698377680*x*y^6*z^5 + 99768240*y^7*z^5 +
116396280*x^6*z^6 + 698377680*x^5*y*z^6 + 1745944200*x^4*y^2*z^6 +
2327925600*x^3*y^3*z^6 + 1745944200*x^2*y^4*z^6 +
698377680*x*y^5*z^6 + 116396280*y^6*z^6 + 99768240*x^5*z^7 +
498841200*x^4*y*z^7 + 997682400*x^3*y^2*z^7 + 997682400*x^2*y^3*z^7 +
498841200*x*y^4*z^7 + 99768240*y^5*z^7 + 62355150*x^4*z^8 +
249420600*x^3*y*z^8 + 374130900*x^2*y^2*z^8 + 249420600*x*y^3*z^8 +
62355150*y^4*z^8 + 27713400*x^3*z^9 + 83140200*x^2*y*z^9 +
83140200*x*y^2*z^9 + 27713400*y^3*z^9 + 8314020*x^2*z^10 +
16628040*x*y*z^10 + 8314020*y^2*z^10 + 1511640*x*z^11 +
1511640*y*z^11 + 125970*z^12 + 167960*x^11 + 1847560*x^10*y +
9237800*x^9*y^2 + 27713400*x^8*y^3 + 55426800*x^7*y^4 +

```

```

77597520*x^6*y^5 + 77597520*x^5*y^6 + 55426800*x^4*y^7 +
27713400*x^3*y^8 + 9237800*x^2*y^9 + 1847560*x*y^10 + 167960*y^11 +
1847560*x^10*z + 18475600*x^9*y*z + 83140200*x^8*y^2*z +
221707200*x^7*y^3*z + 387987600*x^6*y^4*z + 465585120*x^5*y^5*z +
387987600*x^4*y^6*z + 221707200*x^3*y^7*z + 83140200*x^2*y^8*z +
18475600*x*y^9*z + 1847560*y^10*z + 9237800*x^9*z^2 +
83140200*x^8*y*z^2 + 332560800*x^7*y^2*z^2 + 775975200*x^6*y^3*z^2 +
1163962800*x^5*y^4*z^2 + 1163962800*x^4*y^5*z^2 +
775975200*x^3*y^6*z^2 + 332560800*x^2*y^7*z^2 + 83140200*x*y^8*z^2 +
9237800*y^9*z^2 + 27713400*x^8*z^3 + 221707200*x^7*y*z^3 +
775975200*x^6*y^2*z^3 + 1551950400*x^5*y^3*z^3 +
1939938000*x^4*y^4*z^3 + 1551950400*x^3*y^5*z^3 +
775975200*x^2*y^6*z^3 + 221707200*x*y^7*z^3 + 27713400*y^8*z^3 +
55426800*x^7*z^4 + 387987600*x^6*y*z^4 + 1163962800*x^5*y^2*z^4 +
1939938000*x^4*y^3*z^4 + 1939938000*x^3*y^4*z^4 +
1163962800*x^2*y^5*z^4 + 387987600*x*y^6*z^4 + 55426800*y^7*z^4 +
77597520*x^6*z^5 + 465585120*x^5*y*z^5 + 1163962800*x^4*y^2*z^5 +
1551950400*x^3*y^3*z^5 + 1163962800*x^2*y^4*z^5 +
465585120*x*y^5*z^5 + 77597520*y^6*z^5 + 77597520*x^5*z^6 +
387987600*x^4*y*z^6 + 775975200*x^3*y^2*z^6 + 775975200*x^2*y^3*z^6 +
+ 387987600*x*y^4*z^6 + 77597520*y^5*z^6 + 55426800*x^4*z^7 +
221707200*x^3*y*z^7 + 332560800*x^2*y^2*z^7 + 221707200*x*y^3*z^7 +
55426800*y^4*z^7 + 27713400*x^3*z^8 + 83140200*x^2*y*z^8 +
83140200*x*y^2*z^8 + 27713400*y^3*z^8 + 9237800*x^2*z^9 +
18475600*x*y*z^9 + 9237800*y^2*z^9 + 1847560*x*z^10 + 1847560*y*z^10
+ 167960*z^11 + 184756*x^10 + 1847560*x^9*y + 8314020*x^8*y^2 +
22170720*x^7*y^3 + 38798760*x^6*y^4 + 46558512*x^5*y^5 +
38798760*x^4*y^6 + 22170720*x^3*y^7 + 8314020*x^2*y^8 +
1847560*x*y^9 + 184756*y^10 + 1847560*x^9*z + 16628040*x^8*y*z +
66512160*x^7*y^2*z + 155195040*x^6*y^3*z + 232792560*x^5*y^4*z +
232792560*x^4*y^5*z + 155195040*x^3*y^6*z + 66512160*x^2*y^7*z +
16628040*x*y^8*z + 1847560*y^9*z + 8314020*x^8*z^2 +
66512160*x^7*y*z^2 + 232792560*x^6*y^2*z^2 + 465585120*x^5*y^3*z^2 +
581981400*x^4*y^4*z^2 + 465585120*x^3*y^5*z^2 +
232792560*x^2*y^6*z^2 + 66512160*x*y^7*z^2 + 8314020*y^8*z^2 +
22170720*x^7*z^3 + 155195040*x^6*y*z^3 + 465585120*x^5*y^2*z^3 +
775975200*x^4*y^3*z^3 + 775975200*x^3*y^4*z^3 +
465585120*x^2*y^5*z^3 + 155195040*x*y^6*z^3 + 22170720*y^7*z^3 +
38798760*x^6*z^4 + 232792560*x^5*y*z^4 + 581981400*x^4*y^2*z^4 +
775975200*x^3*y^3*z^4 + 581981400*x^2*y^4*z^4 + 232792560*x*y^5*z^4 +
+ 38798760*y^6*z^4 + 46558512*x^5*z^5 + 232792560*x^4*y*z^5 +
465585120*x^3*y^2*z^5 + 465585120*x^2*y^3*z^5 + 232792560*x*y^4*z^5 +
+ 46558512*y^5*z^5 + 38798760*x^4*z^6 + 155195040*x^3*y*z^6 +
232792560*x^2*y^2*z^6 + 155195040*x*y^3*z^6 + 38798760*y^4*z^6 +
22170720*x^3*z^7 + 66512160*x^2*y*z^7 + 66512160*x*y^2*z^7 +
22170720*y^3*z^7 + 8314020*x^2*z^8 + 16628040*x*y*z^8 +
8314020*y^2*z^8 + 1847560*x*z^9 + 1847560*y*z^9 + 184756*z^10 +
167960*x^9 + 1511640*x^8*y + 6046560*x^7*y^2 + 14108640*x^6*y^3 +
21162960*x^5*y^4 + 21162960*x^4*y^5 + 14108640*x^3*y^6 +
6046560*x^2*y^7 + 1511640*x*y^8 + 167960*y^9 + 1511640*x^8*z +
12093120*x^7*y*z + 42325920*x^6*y^2*z + 84651840*x^5*y^3*z +
105814800*x^4*y^4*z + 84651840*x^3*y^5*z + 42325920*x^2*y^6*z +
12093120*x*y^7*z + 1511640*y^8*z + 6046560*x^7*z^2 +
42325920*x^6*y*z^2 + 126977760*x^5*y^2*z^2 + 211629600*x^4*y^3*z^2 +

```

```

211629600*x^3*y^4*z^2 + 126977760*x^2*y^5*z^2 + 42325920*x*y^6*z^2 +
6046560*y^7*z^2 + 14108640*x^6*z^3 + 84651840*x^5*y*z^3 +
211629600*x^4*y^2*z^3 + 282172800*x^3*y^3*z^3 +
211629600*x^2*y^4*z^3 + 84651840*x*y^5*z^3 + 14108640*y^6*z^3 +
21162960*x^5*z^4 + 105814800*x^4*y*z^4 + 211629600*x^3*y^2*z^4 +
211629600*x^2*y^3*z^4 + 105814800*x*y^4*z^4 + 21162960*y^5*z^4 +
21162960*x^4*z^5 + 84651840*x^3*y*z^5 + 126977760*x^2*y^2*z^5 +
84651840*x*y^3*z^5 + 21162960*y^4*z^5 + 14108640*x^3*z^6 +
42325920*x^2*y*z^6 + 42325920*x*y^2*z^6 + 14108640*y^3*z^6 +
6046560*x^2*z^7 + 12093120*x*y*z^7 + 6046560*y^2*z^7 + 1511640*x*z^8
+ 1511640*y*z^8 + 167960*z^9 + 125970*x^8 + 1007760*x^7*y +
3527160*x^6*y^2 + 7054320*x^5*y^3 + 8817900*x^4*y^4 +
7054320*x^3*y^5 + 3527160*x^2*y^6 + 1007760*x*y^7 + 125970*y^8 +
1007760*x^7*z + 7054320*x^6*y*z + 21162960*x^5*y^2*z +
35271600*x^4*y^3*z + 35271600*x^3*y^4*z + 21162960*x^2*y^5*z +
7054320*x*y^6*z + 1007760*y^7*z + 3527160*x^6*z^2 +
21162960*x^5*y*z^2 + 52907400*x^4*y^2*z^2 + 70543200*x^3*y^3*z^2 +
52907400*x^2*y^4*z^2 + 21162960*x*y^5*z^2 + 3527160*y^6*z^2 +
7054320*x^5*z^3 + 35271600*x^4*y*z^3 + 70543200*x^3*y^2*z^3 +
70543200*x^2*y^3*z^3 + 35271600*x*y^4*z^3 + 7054320*y^5*z^3 +
8817900*x^4*z^4 + 35271600*x^3*y*z^4 + 52907400*x^2*y^2*z^4 +
35271600*x*y^3*z^4 + 8817900*y^4*z^4 + 7054320*x^3*z^5 +
21162960*x^2*y*z^5 + 21162960*x*y^2*z^5 + 7054320*y^3*z^5 +
3527160*x^2*z^6 + 7054320*x*y*z^6 + 3527160*y^2*z^6 + 1007760*x*z^7
+ 1007760*y*z^7 + 125970*z^8 + 77520*x^7 + 542640*x^6*y +
1627920*x^5*y^2 + 2713200*x^4*y^3 + 2713200*x^3*y^4 +
1627920*x^2*y^5 + 542640*x*y^6 + 77520*y^7 + 542640*x^6*z +
3255840*x^5*y*z + 8139600*x^4*y^2*z + 10852800*x^3*y^3*z +
8139600*x^2*y^4*z + 3255840*x*y^5*z + 542640*y^6*z + 1627920*x^5*z^2
+ 8139600*x^4*y*z^2 + 16279200*x^3*y^2*z^2 + 16279200*x^2*y^3*z^2 +
8139600*x*y^4*z^2 + 1627920*y^5*z^2 + 2713200*x^4*z^3 +
10852800*x^3*y*z^3 + 16279200*x^2*y^2*z^3 + 10852800*x*y^3*z^3 +
2713200*y^4*z^3 + 2713200*x^3*z^4 + 8139600*x^2*y*z^4 +
8139600*x*y^2*z^4 + 2713200*y^3*z^4 + 1627920*x^2*z^5 +
3255840*x*y*z^5 + 1627920*y^2*z^5 + 542640*x*z^6 + 542640*y*z^6 +
77520*z^7 + 38760*x^6 + 232560*x^5*y + 581400*x^4*y^2 +
775200*x^3*y^3 + 581400*x^2*y^4 + 232560*x*y^5 + 38760*y^6 +
232560*x^5*z + 1162800*x^4*y*z + 2325600*x^3*y^2*z +
2325600*x^2*y^3*z + 1162800*x*y^4*z + 232560*y^5*z + 581400*x^4*z^2
+ 2325600*x^3*y*z^2 + 3488400*x^2*y^2*z^2 + 2325600*x*y^3*z^2 +
581400*y^4*z^2 + 775200*x^3*z^3 + 2325600*x^2*y*z^3 +
2325600*x*y^2*z^3 + 775200*y^3*z^3 + 581400*x^2*z^4 +
1162800*x*y*z^4 + 581400*y^2*z^4 + 232560*x*z^5 + 232560*y*z^5 +
38760*z^6 + 15504*x^5 + 77520*x^4*y + 155040*x^3*y^2 +
155040*x^2*y^3 + 77520*x*y^4 + 15504*y^5 + 77520*x^4*z +
310080*x^3*y*z + 465120*x^2*y^2*z + 310080*x*y^3*z + 77520*y^4*z +
155040*x^3*z^2 + 465120*x^2*y*z^2 + 465120*x*y^2*z^2 +
155040*y^3*z^2 + 155040*x^2*z^3 + 310080*x*y*z^3 + 155040*y^2*z^3 +
77520*x*z^4 + 77520*y*z^4 + 15504*z^5 + 4845*x^4 + 19380*x^3*y +
29070*x^2*y^2 + 19380*x*y^3 + 4845*y^4 + 19380*x^3*z + 58140*x^2*y*z
+ 58140*x*y^2*z + 19380*y^3*z + 29070*x^2*z^2 + 58140*x*y*z^2 +
29070*y^2*z^2 + 19380*x*z^3 + 19380*y*z^3 + 4845*z^4 + 1140*x^3 +
3420*x^2*y + 3420*x*y^2 + 1140*y^3 + 3420*x^2*z + 6840*x*y*z +
3420*y^2*z + 3420*x*z^2 + 3420*y*z^2 + 1140*z^3 + 190*x^2 + 380*x*y

```

```
+ 190*y^2 + 380*x*z + 380*y*z + 190*z^2 + 20*x + 20*y + 20*z + 1
```

```
type(f)
```

```
<type
'sage.rings.polynomial.multi_polynomial_libsingular.MPolynomial_libs\
ingular'>
```

```
f = x+y+z+1
```

```
timeit('f*f')
```

```
625 loops, best of 3: 1.33  $\mu$ s per loop
```

```
g = singular(f); g
```

```
x+y+z+1
```

```
type(g)
```

```
<class 'sage.interfaces.singular.SingularElement'>
```

```
g.name()
```

```
'sage11'
```

```
timeit('g*g')
```

```
625 loops, best of 3: 392  $\mu$ s per loop
```

```
392/1.33
```

```
294.736842105263
```

```
k.<a> = GF(37^2)
```

```
type(k)
```

```
<type 'sage.rings.finite_field_givaro.FiniteField_givaro'>
```

```
timeit('a*a')
```

```
625 loops, best of 3: 320 ns per loop
```

```
625 loops, best of 3: 1.32  $\mu$ s per loop
```

```
I= sage.rings.ideal.Cyclic(R)
```

```
I
```

```
Ideal (x + y + z + w + m, x*y + y*z + z*w + x*m + w*m, x*y*z + y*z*w
+ x*y*m + x*w*m + z*w*m, x*y*z*w + x*y*z*m + x*y*w*m + x*z*w*m +
y*z*w*m, x*y*z*w*m - 1) of Multivariate Polynomial Ring in x, y, z,
w, m over Rational Field
```

```
search_src('groebner_basis')
```

## Search Source Code: groebner\_basis

1. [crypto/mq/mpolynomialssystem.py](#)
2. [crypto/mq/sbox.py](#)
3. [crypto/mq/sr.py](#)
4. [rings/ideal.py](#)
5. [rings/polynomial/groebner\\_fan.py](#)
6. [rings/polynomial/multi\\_polynomial\\_ideal.py](#)
7. [rings/polynomial/multi\\_polynomial\\_ideal\\_libsingular.pyx](#)
8. [rings/polynomial/pbori.pyx](#)
9. [rings/polynomial/toy\\_buchberger.py](#)
10. [schemes/generic/affine\\_space.py](#)
11. [schemes/generic/projective\\_space.py](#)