[2d Graphics Requests and Plans](#_vvf25zxhktrs)

[Graph Theory](#_8zpu1svgn9ng)

[General 2d Plotting Feature Requests](#_x1mhfyjyy4uf)

[Matplotlib Screenshots](#_uqspx2xipkwi)

[Sage 3D](#_xurcuavklzkp)

[Scope](#_72dpy8yuphn)

[2.5D](#_wfruqubm8ws7)

[3D](#_i7hbkle9xjmj)

[More 3D Issues](#_adla39ck88qn)

[4D](#_u7h0s6xw4wu5)

[Time](#_wm4ob5hxq70y)

[Abstractions which use 3D](#_vv2pn16ja90z)

[Basic Calculus](#_68ciudn6iq3c)

[Design Constraints](#_prtdxw2lxpjn)

[Currently in SAGE](#_x7ugogazoc9x)

[How SAGE currently does calculus](#_kbscaghk4q3p)

[Bugs in the current implementation](#_uamzkqdf0rv)

[Exhibit 1](#_4km07kiphfe1)

[Exhibit 2](#_vnnukc8esoh3)

[Exhibit 3](#_s4p3mypi6d6g)

[Exhibit 4](#_jdoeqham2v2k)

[Exhibit 5](#_qoa58f7b1iu3)

[Exhibit 6](#_gpfguwonuw20)

[Design Ideas](#_pznt6ri3vttn)

[Introduction to the Axiom Library compiler for Python Programmers](#_piiofzbmpbce)

[Bern:](#_t6b5ewhv3sh2)

[algebraic geometry sep](#_68057oq4rau)

[Requests](#_5mxoys91yfma)

[Critical](#_17vbvu6ak17z)

[Dream features](#_3ftp3haak74b)

[Dream code](#_glmunw6vzvb6)

[SEP: Using AutoTools as Build system for Sage](#_tv3kziu3f6lk)

[Requirements](#_rq05s232dg0u)

[Proposal](#_6u91rytge2jc)

[Advantages](#_ms58n6dk64j7)

[Disadvantages](#_y81ok0tudl2q)

[A Short Introduction to Sage](#_c35xuw8ygo9k)

[BuildSystem SEP](#_riqnyceba6n1)

[goals](#_32rnnqd5kacn)

[implementation](#_b6d6onh28i1b)

[open/tricky](#_uz5zifut42s2)

[future](#_dra041p089mf)

[roadmap](#_fjp00v5b1y9c)

[Ticket #1183 is fixed, as of Feb 2017. https://trac.sagemath.org/ticket/1183](#_mxvvdwjavxsm)

[SEP 1: Calculus in SAGE](#_4xktqpj2ue6k)

[Authors](#_ot28f9rxp0ev)

[Preamble](#_8ekhb24x8v2l)

[Abstract](#_1ps5fx3ytoba)

[Copyright](#_l11jagpb72kf)

[Construction of calculus functions](#_216e3y6za32w)

[A proposed class hierarchy](#_7nezzbr6w959)

[Operations on functions](#_votwuzhkfk8w)

[Invoking other systems](#_13slf7l355ji)

[Basic operations](#_6k8bgbs3sdgi)

[Implementation of basic operations](#_m5a5rkmux73d)

[Substitution](#_jtvqqztu6hax)

[Algorithm](#_xclaj5vdrfyf)

[Implementation](#_21dox3mhgr42)

[Preparsing for defining a callable symbolic expression](#_4js9x2hb9cg0)

[Motivation](#_25z4h4ys4kv1)

[Rationale](#_ipnai4u98kts)

[References](#_8onji7lg4aur)

[Cantor](#_g7om8340xznh)

[Cantor](#_njlbkaw7291d)

[Install Cantor at Ubuntu Karmic](#_rcga7kmyizwd)

[Build and Install Cantor](#_2maw5moom4h2)

[Using Sage with Cantor](#_xx2fv2rysyp9)

[Links](#_vqy703p486pu)

[cmake](#_khfqgwksnmth)

[Description](#_xmokij1sds4l)

[License](#_e96ngiya5c1k)

[SPKG Maintainers](#_98dzfy7wxd3f)

[Upstream Contact](#_sh750id201r5)

[Dependencies](#_wxtqwcojpma3)

[Changelog](#_d0mhqdze3pnk)

[Quaternion Algebras](#_qonhkr9oe2bm)

[Combinat/01](#_pf2hsvgc3v99)

[Design discussion for permutations and discrete functions](#_61taaat571ea)

[Potential solutions:](#_rk5maa3n7v3f)

[Extra documentation](#_3cs75g8gu4xy)

[GmoduleSEP](#_bhh839mmhs3q)

[Ideas of things to Implement in SAGE](#_89zterwvfou3)

[Malloc Replacements](#_v1as0wqq3ci)

[Introduction](#_dskpb73x6s6n)

[malloc](#_6yr8hhmlcwn5)

[PyMem\_Malloc](#_hb4jest4myz2)

[OMalloc](#_ouwlrep9v959)

[DLMalloc](#_gsqz7zze5ynt)

[default libCF Memory Manager](#_w7zeugcwm5wa)

[libCF new Memory Manager](#_ywhxp2jygj59)

[GivMM](#_dtbimadehhpk)

[NotebookWiki](#_qf5bv7uk0iad)

[Open Source CAS](#_nmd23ceu6mrr)

[The terrain](#_mwnnxdnixd39)

# 2d Graphics Requests and Plans

## Graph Theory

Plan: Build our own graphics primitive instead of relying on NetworkX.

Requests:

* Multiedges
* Dashed edge option
* Control of arrow width
* Nice tree drawing
* Allow larger labels and account for them in vertex spacing

## General 2d Plotting Feature Requests

* Legends for plots
* Supporting both rgbcolor and color input
* Concatenating graphics primitives (instead of adding) trac #3209
* nice histogram command with axes control (i.e. finance.timeseries)
* ugly tickmarks
* mathematica functionality (with similar interface)
* auto scientific notation for large axis labels
* symbolic axis labels

## Matplotlib Screenshots

* <http://matplotlib.sourceforge.net/screenshots.html>

# Sage 3D

## Scope

3D is 1 more D. More is better.

Full 3D gets nasty very quickly. Actually, full 2D is pretty nasty so most of us stick with 1D projections onto 2D (graphs etc) and occasionally dabbling with extensions like color, flow fields etc.

In the fullness of time, Sage should probably include a range of packages simply because there's a lot of very good work thats been done and it is "relatively" straightforward to incorporate. However, making 3D useful to the Sage community will likely be tricky.

## 2.5D

2.5D spans the trivial to the nasty as well but lets start with the trivial. Generally, one starts with a grid where the grid coordinates map to 2D coordinates and the values are the 3'rd D. One typical projection is color which leads to contour maps and currently exists in sage through matplotlib.

Next is, generally, a surface using the coordinates... And away we go.

3D surfaces are triangles in hardware or are "renderings" onto 2D images. If we do the trivial projection of gridded data to a surface, we're already "deciding" what that means. Perhaps we split along the diagonal and spit triangles to the 3D card. Maybe we "smooth" the surface using NURBS and render.

But we're math guys right? So, lets abstract a bit to consider utility for Sage's user base. A very useful projection is "Parametric" meaning we define a function Z=F(u,v) and display that...

Ugh... turns out this gets nasty quickly... F tends to behave badly and is certainly not uniform. So, how do we wiggle through (u,v) and not miss critical points. Exhaustive searching is computationally expensive (impossible?).

Fortunately, this problem isn't unique and many packages exist which do "pretty well"

more co come...

## 3D

In its most general form, 3D requires a "Scene Graph" which is nothing more than a tree maintaining transformations and descriptions of aggregations of primitives and transformations. Many packages exist (list to be compiled... [OpenSceneGraph](https://wiki.sagemath.org/OpenSceneGraph), Coin3D)

Then we have the "specialized" 3D packages which abstract some of this into a more useful space. VTK is one such package, but one doesn't "dabble" with VTK (nor 3D for that matter)

### More 3D Issues

While managing scenegraphs can be daunting, it gets uglier when considering the non-geometric aspects such as lighting, surface characteristics (specularity, luminosity) etc. As above, there are often ways to pick a reasonable default which makes most happy yet allows for tweaking later.

## 4D

Well, we need a place to stick animation stuff.

## Time

Time is a special case unless you're into subatomic physics... in which case you're probably too "warped" to be reading this... due to causality. Causality is typically handled at the mathematics level so for a 3D system, what we're really interested in is dynamics.

From a design perspective, dynamic 3D rendering used to be hard. In a strange way, its now harder to build systems which aren't 3D at the infrastructure level. The Compiz desktop is a 3D infrastructure with 2D projections of the tools we all know and love... with a few lines of code to nail X-windows to the projections.

## Abstractions which use 3D

* 3D Graphs - a good example that we can't use is: [UBIGRAPH](http://ubietylab.net/ubigraph/)

# Basic Calculus

A page to coordinate the implementation of a unified model for basic calculus that is "insanely easy" (for non-mathematicians) to use.

## Design Constraints

This is a (very incomplete) list of goals that are very important for us to reach, in order for SAGE to be adopted as a mainstream computer algebra system (i.e., a system that is ideal for experts and non-experts alike).

* Must be "insanely easy", à la [David\_Joyner](https://wiki.sagemath.org/David_Joyner)
* Must require no knowledge of mathematics beyond calculus in order to use (this likely means matrices and polynomials with implicit base rings).
* Must require no knowledge of any system outside SAGE. Learning SAGE should require learning one system; not learning several.
* Must interoperate with other parts of SAGE.
* Calculus-related functions (such as, say, differentiate()) should return SAGE objects on which we can do more operations (for example, isolate a variable).
* The system should cooperate in a natural way with other components of SAGE (this may mean providing an interface hiding algebraic notions for modules).
* Transparency between
* Intuitive handling of ODEs

## Currently in SAGE

## How SAGE currently does calculus

Coming soon.

## Bugs in the current implementation

These are thanks to Joel Mohler:

### Exhibit 1

sage: x.parent()  
 Univariate Polynomial Ring in x over Rational Field  
sage: P,y=ZZ['y'].objgen()  
sage: f=cos(x)\*sin(y)  
sage: f(1)  
(cos(1)\*sin(1))

This should have resulted in a syntax error.

### Exhibit 2

What in the world is "sin(x)(1)"?

sage: P,(x,y)=ZZ['x,y'].objgens()  
sage: f=sin(x)  
sage: f(1)  
 sin(x)(1)  
sage: float(f(1))  
# disaster strikes (recursive traceback)

From William: Yes, this is wrong. But it's just a [NotImplementedError](https://wiki.sagemath.org/NotImplementedError), i.e., the \_\_call\_\_ method of sin doesn't do anything with multivariate polynomials, so it just forms the formal evaluation sin(x), which it doesn't know what to do with.

### Exhibit 3

This should result in a type error:

sage: P,x=IntegerModRing(17)['x'].objgen()  
sage: f=sin(x)  
sage: f(18)  
 sin(1)

### Exhibit 4

Constants should have coercion to a function of 1 variable.

sage: show(plot(1))  
---------------------------------------------------------------------------  
<type 'exceptions.ValueError'> Traceback (most recent call last)

### Exhibit 5

If we use a polynomial ring generator as generic function variables, exponentiation must work like so:

sage: P,x=ZZ['x'].objgen()  
sage: x^x  
---------------------------------------------------------------------------  
<type 'exceptions.TypeError'> Traceback (most recent call last)

### Exhibit 6

This worked in 1.4, but does not in 1.5.alpha6. I suspect it's only a simple regression due to the new arithmetic properties.

sage: pi\*x  
---------------------------------------------------------------------------  
<type 'exceptions.NotImplementedError'> Traceback (most recent call last)

## Design Ideas

Some ideas to consider and debate.

* One idea, which I think students will accept, is to have a command like load\_calculus() or something. (The Maple analog is with(student).) This could, for example,
  + load predefined variables,
  + spit out a very brief screen of explanations (perhaps a list of some available commands, an explanation of how to get help in the notebook or on the command-line, ...),
  + load a bunch of teacher-defined \*.sage files in examples/calculus,
  + other stuff?
* Of course, one extremely cool thing about the SAGE notebook is that you can actually design a special calculus control bar if you want. Maybe if you click on help, one of the items says calculus, and there is a link there to explain how to put a calculus help control bar where the normal control bar is .

# Introduction to the Axiom Library compiler for Python Programmers

* SPAD is the 1st generation compiler and Aldor (<http://wiki.axiom-developer.org/Aldor>) is the 2nd generation compiler for the strongly-typed object-oriented high-level language which is used to implement Axiom's library of over 1,300 mathematical domains and categories. The language is similar in many respects to Python which plays the same role in Sage.

Example 1: A prime number sieve to count primes <= n.

#pile  
#include "axiom.as"  
  
import from Boolean, Integer, NonNegativeInteger  
sieve(n: Integer): Integer ==  
 isprime: OneDimensionalArray Boolean := new(n::NonNegativeInteger, true)  
 np:Integer := 0  
 for p in 2..n | isprime p repeat  
 np := np + 1  
 for i in (p+p)..n by p repeat isprime i := false  
 np

# Bern:

Oops, this is already in Sage.

Code to do it...

chi-bernoulli system:sage

From page 656 of Cohen:

{{{id=0| def S(n,chi):

* return sum(chi(r)\*r^n for r in [0..chi.modulus()-1])

def bern(k,chi):

* m = chi.modulus() return sum([binomial(k,j) \* bernoulli(j)\*m^(j-1)\*S(k-j,chi) for j

in [0..k]]) /// }}}

{{{id=1| chi = [DirichletGroup](https://wiki.sagemath.org/DirichletGroup)(33).1 /// }}}

{{{id=2| v = [(n,len(str(bern(2\*n+1,chi)))) for n in [1..50]] /// }}}

{{{id=6| finance.[TimeSeries](https://wiki.sagemath.org/TimeSeries)([a for \_, a in v]).plot() /// }}}

{{{id=3| time chi.bernoulli(97) ///

-1386550730901281655459039291153060660155554359838499623998185825120793577393363680421610680951607498384083291848913360896584699889220142412405950009577924616113546699259811090054434267531175979305388645876564283593414395718/11\*zeta10^3 + 1139018380559996978485536615865491587631736962914990971722815468579216496531837604612620427605357600704292888340343244789306247121665563078738013093738955470092365206329028171332764293574767994686073536165936744981416609102/11\*zeta10^2 - 1815995258580769504023108459745030375976990595454787816812259082650991112133850260010917340017496257156415860250825070730584479213960689975366958760972790104039091614348113102193952661883076923857562553616318209769197351882/11\*zeta10 + 1592208039972106215192346184315475058638993887793107974730549965346512172063570657390267817543848692352055390892551815337023626015369750759903577411594239137472016340616541079239824887795689418707971932300648448020408760118/11 CPU time: 4.17 s, Wall time: 4.35 s }}}

{{{id=4| time bern(97,chi) ///

-1386550730901281655459039291153060660155554359838499623998185825120793577393363680421610680951607498384083291848913360896584699889220142412405950009577924616113546699259811090054434267531175979305388645876564283593414395718/11\*zeta10^3 + 1139018380559996978485536615865491587631736962914990971722815468579216496531837604612620427605357600704292888340343244789306247121665563078738013093738955470092365206329028171332764293574767994686073536165936744981416609102/11\*zeta10^2 - 1815995258580769504023108459745030375976990595454787816812259082650991112133850260010917340017496257156415860250825070730584479213960689975366958760972790104039091614348113102193952661883076923857562553616318209769197351882/11\*zeta10 + 1592208039972106215192346184315475058638993887793107974730549965346512172063570657390267817543848692352055390892551815337023626015369750759903577411594239137472016340616541079239824887795689418707971932300648448020408760118/11 CPU time: 0.22 s, Wall time: 0.22 s }}}

{{{id=5|

/// }}}

# algebraic geometry sep

## Requests

### Critical

* fast R[x\_1,...,x\_n], also with fractional exponents
* GB's, flexible gradings, term orders
* rings (not necessarily commutative)
* modules (not just ideals, not just free)
* sheaves
* homological algebra (free resolutions)
* linear algebra with basis an arbitrary index set I
* very very fast sparse and dense linear algebra over many types of rings and fields

### Dream features

* representation theory for finite groups (char 0 and modular, not just GAP, compare to what MAGMA can do -- and how fast it can do it)
* rings of representations (Grothendieck rings, etc.)
* local rings and global rings: localization, really working (not just M2)
* GB over all rings (e.g. field extensions), even noncommutative when possible
* full functoriality (e.g. preservation of GL\_n-actions, functors, operations on functors, Yoneda product, tensor products)
* full homological algebra (spectral sequences, derived categories, etc.)
* parallelize everything
* a "good clean" programming language (not just M2, e.g. Maple -- having to put things into rings before being able to use them is annoying; work easily with general expressions)
* super fast GB's and syzygies (speed and low memory usage)
* super fast and low memory primary decomposition (e.g. numerical) and integral closure
* sheaves, Chern classes, intersection theory on singular spaces
* algebraic topology on complex and real points on a variety
* etale cohomology
* usable resolution of singularities

## Dream code

[Deformation theory](https://wiki.sagemath.org/DeformationTheory)

[Polytopes](https://wiki.sagemath.org/Polytopes)

# SEP: Using AutoTools as Build system for Sage

This SEP proposes to switch the Sage build system to the standard autotools (automake, autoconf, etc.).

## Requirements

The top level build process needs to

* Verify the needed prerequisites are present
* Setup the build environment
* Unpack and build each spkg (in order) each of which has its own (upstream-dictated) build framework

## Proposal

In keeping with Sage's philosophy of not reinventing the wheel, Sage should use a well established build system. Sage should follow the standard procedure of

./configure && make && sudo make install

in order to install itself system wide and

./configure --prefix=$HOME/sage && make && make install

to install into directory sage.

## Advantages

* Automake generates a number of standard targets that need not be hand-code. (What ones?)
* Makes the Sage build machinery more portable.
  + Can this be made more specific? What in particular is currently not portable?
* Hardcode SAGE\_ROOT into the sage and sage-\* scripts at configure time. (Would this impact relocating sage?)
  + This would break relocation which Sage has been able to do for quite awhile. If we did this, then users who installed binaries in non-standard places would have to set environment variables (like Axiom and FriCAS). It would also be more cumbersome / error-prone when you're trying to work with multiple binary installations of Sage.
* Options to configure would be passed to sub-packages (like, for example, Singular)
* Clear distinction of Sage core (i.e. the bolts) and its third-party parts (NTL, GAP, Pari, Singular, MPFR, Maxima, etc.)
* Checking system features is done at configure time and thus can abort early if something is missing.
  + This is the purpose of the prereq spkg. It basically runs a configure script to make sure that everything needed is installed.

## Disadvantages

* Changing a .py file must be followed by "make install" in order to be visible in an installed Sage session.
* Probably different development workflow for new Sage code.
* SPackages must be downloaded before configure is run. make should never have a need to access the Internet.
  + (spkgs should be installed by a debian like package system, not make.)
  + What portable package systems are there?
* Automake would add another level of complexity to the build process, in particular one which is not understood by many in the community.

# A Short Introduction to Sage

Sage is a mathematics software package for number theory, algebra, and geometry computation. The implementation is due to [William\_Stein](https://wiki.sagemath.org/William_Stein) and over 50 other contributors from [around the world](http://www.sagemath.org/development-map.html). It is open source and freely available under the terms of the GNU General Public License (GPL). Sage is

* A unified distribution of free mathematics software that mere mortals can build from source;
* A new Python library for mathematical computation with graphical and command-line interfaces; and
* An interface to existing mathematics software packages.

It is written in [Python](https://wiki.sagemath.org/Python), C++, and C (via [Pyrex" / "Cython](https://wiki.sagemath.org/Pyrex%22%20/%20%22Cython) <http://www.cython.org/>). Python is an open source object-oriented interpreted language, with a large number of libraries, e.g., for numerical analysis, which are available to users of Sage. Python can also be accessed in library mode from C/C++ programs.

Sage ships with and provides a unified interface to several important open source libraries, including [SINGULAR](https://wiki.sagemath.org/Singular) (for commutative algebra), [SciPy](https://wiki.sagemath.org/scipy) (for applied mathematics), [GAP](https://wiki.sagemath.org/GAP) (for group theory), John Cremona's [MWRANK](https://wiki.sagemath.org/MWRANK) library (for elliptic curves), the [PARI](https://wiki.sagemath.org/PARI) library for number theory, [LinBox](https://wiki.sagemath.org/linbox) (for linear algebra), Victor Shoup's number theory library [NTL](https://wiki.sagemath.org/NTL), and [Maxima](https://wiki.sagemath.org/Maxima) (for symbolic manipulation). There are also interfaces to many commercial or closed source mathematics software packages including Maple, MATLAB, Mathematica, KASH/KANT, MAGMA, Axiom, and many others. For graphics, Sage includes Matplotlib (for 2-D and some 3-D graphics), jmol (3-D graphics) and tachyon (a 3-D ray tracer).

The main long-term goals and guiding principles for Sage are:

* Free and open source: The source code must be freely available and readable, so users can understand what the system is really doing and more easily extend it. Just as mathematicians gain a deeper understanding of a theorem by carefully reading or at least skimming the proof, people who do computations should be able to understand how the calculations work by reading documented source code. All software included in the Sage core distribution must be free and open source, and arbitrary modifications and redistribution of every single line must be allowed.
* Good programming environment: It is hoped that Sage will form a stable environment to begin programming new mathematics software projects. Sage provides a model for the mathematical community of software development with a strong emphasis on openness, community, cooperation, and collaboration.
* Extensible: Be able to define new data types or derive from built-in types, and make code written in a favorite language (including C/C++) part of the system.
* User friendly: The hope is to eventually attain a high level of user support.
* Easy to compile: Sage should be relatively easy to compile from source for Linux and Mac OS X users. This provides more flexibility in modifying the system.
* Cross-platform: Sage runs under Linux, Mac OS X, Windows.
* Comprehensive: Implement enough algorithms to be really useful. Unify free open source mathematics software. Be a comprehensive mainstream high quality open source free mathematics software system.
* Efficient: Be very fast---comparable to or faster than anything else available. This is very difficult, since many systems are closed source, algorithms are sometimes not published, and finding fast algorithms is often extremely difficult (years of work, Ph.D. theses, luck, etc.).
* Tools: Provide robust interfaces to some of the functionality of PARI, GAP, GMP, Maxima, SINGULAR, MWRANK, and NTL. These are all GPL'd and Sage provides a unified interface for using them.
* Well documented: Reference manual, tutorial, API reference with examples for every function, and an extensive "How can ... be constructed in Sage?" document.

Download Sage, an installation guide and a tutorial, from its webpage

<http://www.sagemath.org/>.

Although Sage uses Python, PARI, and other packages, keep in mind that it is not necessary to have those software packages preinstalled on your computer. The installation of Sage is really designed to be relatively painless, but if there are any problems, please ask (the web page has links to the email lists for Sage support). There are installation instructions at <http://www.sagemath.org/doc/html/inst/index.html>. Moreover, if Sage is installed once, upgrading to the newest version is especially easy with the "upgrade" command option (this is described on the website given above and assumes that the command "wget" is installed and you have an Internet connection).

Once Sage is installed into a directory such as "sage-x.y.z" in Linux, cd to this directory and type the command "./sage" to start Sage. Another way to use Sage is to, from within a command-line Sage session, type "notebook(open\_viewer=True)" at the Sage prompt. This (a) starts the Sage server running (loading the previously used worksheet, if any); and (b) opens the Firefox (or your default system) browser (or starts a new tab if it is already open) and displays the Sage notebook webpage.

At the Sage prompt, type "factor(100)" and then press the Return or Enter key. Sage will output the prime factorization of 100. Type "factor??" (and press Return) and the source code of the function "factor" will be shown. Enter Ctrl-d (hold the Ctrl key and d at the same time) or type "quit" (return), and Sage will exit. To do this computation in the notebook, enter "factor(100)" into a "cell" (an empty white window in the web browser) and then press shift-enter. Sage will display the factorization in the space just below the cell. To exit the notebook, go back to the command line and press Ctrl-c. Next, exit the browser. This quits Sage's notebook.

There are thousands of additional examples in the tutorial, constructions, and reference manuals, which are available on the Sage webpage. Please try them out!

Sage is actively growing, and is already usable. There are Sage discussion lists, [bug-tracker](http://trac.sagemath.org/sage_trac), [wiki](http://wiki.sagemath.org/), and [code browser](https://github.com/sagemath/sage). Please visit

<http://www.sagemath.org/> (west coast of USA)

or

[http://sage.apcocoa.org](http://sage.apcocoa.org/) (Germany)

or

<http://echidna.maths.usyd.edu.au/sage/> (Australia)

or

email the Sage group at one of the email lists <http://www.sagemath.org/lists.html>

for more information. Above all, have fun with Sage!

David Joyner wdjoyner@gmail.com

William Stein wstein@gmail.com

Last Updated 1-18-2008.

[DocumentationProject](https://wiki.sagemath.org/DocumentationProject)

# BuildSystem SEP

## goals

* module build systems seperated from toplevel package/module management
* make embedded modules usable outside sage ("the distribution")
* (still) build depends only on (gnu) make and bash
* allow fallback to system-installed modules/packages. not just for gcc
* make embedded modules work with other (gnu/linux) software distributions ([gsoc project](http://www.google-melange.com/gsoc/project/google/gsoc2013/felixs/24001))
* VPATH build
* install/uninstall for all packages within sage
* allow swich between system-installed and sage-shipped packages
* parallel build
* install targets conforming to standard
* avoid patchbomb (see roadmap)

## implementation

* autotools can do most things. (no, this is not just [AutoToolsSEP](https://wiki.sagemath.org/AutoToolsSEP))
* package dependencies remain make recipes
* package management can be reused, must be simplified/extended.
* foreign package build scripts will be turned into makefiles defaulting to "install"
* DESTDIR is used to stage package contents

## open/tricky

* how to upgrade a sage-(local?)-installation?
* DESTDIR is unsuppported in the current foreign packaging system
* distutils is used for the python library. patch or rewrite?

## future

* pick minimal set of foreign packages, depending on what is available
* optionally use portage for package management
* run from builddir without the need to install (simplifies development, YMMV)

## roadmap

* demonstrate proof-of-concept on top of master
* switch embedded module build systems to autotools
* add toplevel configure.ac GNUmakefile.am
* rework configuration parts into configure.ac
* rewire toplevel Makefile, wrap functionality if required
* remove old build system fragments.
* rearrange/regroup embedded modules (?)
* add DESTDIR support to foreign modules
* fix local "installation" and actual installation targets

# Ticket #1183 is fixed, as of Feb 2017. <https://trac.sagemath.org/ticket/1183>

1. (done) implement I.free\_module() and O.free\_module() as ZZ-submodules of K.vector\_space()
2. Compute O/I we instead first compute (O mod p) and (Ibar subset O mod p). Thus we have two F\_p vector spaces and compute the quotient of one by the other -- hey we just finished implementing that nicely.
3. def \_p\_quotient(self, p):  
    """  
    EXAMPLES:  
    sage: K.<i> = NumberField(x^2 + 1); O = K.maximal\_order()  
    sage: I = K.factor\_integer(3)[0][0]  
    sage: I.\_p\_quotient(3)  
    Vector space quotient V/W of dimension 2 over Finite Field of size 3 where  
    V: Vector space of dimension 2 over Finite Field of size 3  
    W: Vector space of degree 2 and dimension 0 over Finite Field of size 3  
    Basis matrix:  
    []  
    sage: I = K.factor\_integer(5)[0][0]  
    sage: I.\_p\_quotient(5)  
    currently broken  
    """  
    return quotient\_char\_p(self, p)  
     
     
   def quotient\_char\_p(I, p):  
    """  
    Given an integral ideal I that contains a prime number p, compute  
    a vector space V = (OK mod p) / (I mod p), along with a  
    homomorphism OK --> V and a section V --> OK.  
    """  
    if not I.is\_integral():  
    raise ValueError, "I must be an integral ideal."  
      
    K = I.number\_field()  
    OK = K.maximal\_order(p) # really only need a p-maximal order.  
    M\_OK = OK.free\_module()  
    M\_I = I.free\_module()  
     
    # Now we have to quite explicitly find a way to compute  
    # with OK / I viewed as a quotient of two F\_p vector space,  
    # and itself viewed as an F\_p vector space.  
     
    # Step 1. Write each basis vector for I (as a ZZ-module)  
    # in terms of the basis for OK.  
    B\_I = M\_I.basis()  
    M\_OK\_change = M\_OK.basis\_matrix()\*\*(-1)  
    B\_I\_in\_terms\_of\_M = M\_I.basis\_matrix() \* M\_OK\_change  
      
    # Step 2. Define "M\_OK mod p" to just be (F\_p)^n and  
    # "M\_I mod p" to be the reduction mod p of the elements  
    # compute in step 1.  
    n = K.degree()  
    k = FiniteField(p)  
    M\_OK\_modp = k\*\*n  
    B\_mod = B\_I\_in\_terms\_of\_M.change\_ring(k)  
    M\_I\_modp = M\_OK\_modp.span(B\_mod.row\_space())  
      
    # Step 3. Compute the quotient of these two F\_p vector space.  
    Q = M\_OK\_modp.quotient(M\_I\_modp)  
      
    # Step 4. Now we get the maps we need from the above data.  
    return Q
4. Choose elements of O (not at random) until we find one whose minpoly generators Obar/Ibar. When computing the minpoly, I think we do that by computing the matrix whose rows are the powers of the reduction of our candidate.

def ResidueField(p, name = None, check = True):  
 """   
 A function that takes in a prime ideal and returns a number field.   
   
 INPUT:   
 p -- a prime integer or prime ideal of an order in a number field.   
 name -- the variable name for the finite field created. Defaults to the name of the number field variable.   
 check -- whether or not to check if p is prime.   
   
 OUTPUT:   
 The residue field at the prime p.   
   
 EXAMPLES:   
 sage: from sage.rings.residue\_field import ResidueField   
 sage: K.<a> = NumberField(x^3-7)   
 sage: P = K.ideal(29).factor()[0][0]   
 sage: k = K.residue\_field(P)   
 sage: k   
 Residue field of Fractional ideal (2\*a^2 + 3\*a - 10)   
 sage: k.order()   
 841   
 """  
 key = (p, name)  
 if residue\_field\_cache.has\_key(key):  
 ans = residue\_field\_cache[key]()  
 if ans is not None:  
 return ans  
 if PY\_TYPE\_CHECK(p, Integer):  
 if check and not p.is\_prime():  
 raise ValueError, "p must be prime"  
 if name is None:  
 name = 'x'  
 ans = ResidueFiniteField\_prime\_modn(p, name)  
 elif is\_NumberFieldIdeal(p):  
 if name is None:  
 name = p.number\_field().variable\_name()  
 if check and not p.is\_prime():  
 raise ValueError, "p must be prime"  
 # Should generalize to allowing residue fields of relative extensions to be extensions of finite fields.   
 characteristic = p.smallest\_integer()  
 K = p.number\_field()  
 OK = K.maximal\_order() # should change to p.order once this works.   
 U, to\_vs, to\_order = p.\_p\_quotient(characteristic)  
 k = U.base\_ring()  
 R = PolynomialRing(k, name)  
 n = p.residue\_class\_degree()  
 gen\_ok = False  
 try:  
 x = K.gen()  
 M = matrix(k, n+1, n, [to\_vs(x\*\*i).list() for i in range(n+1)]).transpose()  
 if M.rank() == n:  
 gen\_ok = True  
 f = K.polynomial().change\_ring(k)  
 except TypeError:  
 pass  
 if not gen\_ok:  
 for u in U: # using this iterator may not be optimal, we may get a long string of non-generators   
 x = to\_order(u)  
 M = matrix(k, n+1, n, [to\_vs(x\*\*i).list() for i in range(n+1)]).transpose()  
 M.echelonize()  
 if M.rank() == n:  
 f = R((-M.column(n)).list() + [1])  
 break  
 if n == 1:  
 ans = ResidueFiniteField\_prime\_modn(p, name, im\_gen = -f[0], intp = p.smallest\_integer())  
 else:  
 q = characteristic\*\*(f.degree())  
 if q < Integer(2)\*\*Integer(16):  
 ans = ResidueFiniteField\_givaro(p, q, name, g, characteristic)  
 else:  
 ans = ResidueFiniteField\_ext\_pari(p, q, name, g, characteristic)  
 else: # Add support for primes in other rings later.   
 raise TypeError, "p must be a prime in the integers or a number field"  
 residue\_field\_cache[key] = weakref.ref(ans)  
 return ans

OR, my clean up of it:

def ResidueField(p, names = None, check = True):  
 """  
 A function that returns the residue class field of a prime ideal p  
 of the ring of integers of a number field.  
  
 INPUT:  
 p -- a prime integer or prime ideal of an order in a number  
 field.  
 names -- the variable name for the finite field created.  
 Defaults to the name of the number field variable but  
 with bar placed after it.  
 check -- whether or not to check if p is prime.  
  
 OUTPUT:  
 -- The residue field at the prime p.  
  
 EXAMPLES:  
 sage: K.<a> = NumberField(x^3-7)  
 sage: P = K.ideal(29).factor()[0][0]  
 sage: ResidueField(P)  
 Residue field in abar of Fractional ideal (2\*a^2 + 3\*a - 10)  
 sage: k = K.residue\_field(P); k  
 Residue field in abar of Fractional ideal (2\*a^2 + 3\*a - 10)  
 sage: k.order()  
 841  
 """  
 if isinstance(names, tuple):  
 if len(names) > 0:  
 names = str(names[0])  
 else:  
 names = None  
 key = (p, names)  
 if residue\_field\_cache.has\_key(key):  
 k = residue\_field\_cache[key]()  
 if k is not None:  
 return k  
 if PY\_TYPE\_CHECK(p, Integer):  
 if check and not p.is\_prime():  
 raise ValueError, "p must be prime"  
 if names is None:  
 names = 'x'  
 k = ResidueFiniteField\_prime\_modn(p, names)  
 elif is\_NumberFieldIdeal(p):  
 if names is None:  
 names = '%sbar'%(p.number\_field().variable\_name())  
 if check and not p.is\_prime():  
 raise ValueError, "p must be prime"  
 # Should generalize to allowing residue fields of relative extensions to be extensions of finite fields.  
 characteristic = p.smallest\_integer()  
  
 K = p.number\_field()  
 OK = K.maximal\_order() # should change to p.order once this works.  
   
 U, to\_vs, to\_order = p.\_p\_quotient(characteristic)  
 k = U.base\_ring()  
 R = PolynomialRing(k, names)  
 n = p.residue\_class\_degree()  
 gen\_ok = False  
 try:  
 x = K.gen()  
 from sage.matrix.constructor import matrix  
 M = matrix(k, n+1, n, [to\_vs(x\*\*i).list() for i in range(n+1)]).transpose()  
 if M.rank() == n:  
 gen\_ok = True  
 f = K.polynomial().change\_ring(k)  
 except TypeError:  
 pass  
 if not gen\_ok:  
 for u in U: # using this iterator may not be optimal, we may get a long string of non-generators   
 x = to\_order(u)  
 M = matrix(k, n+1, n, [to\_vs(x\*\*i).list() for i in range(n+1)]).transpose()  
 M.echelonize()  
 if M.rank() == n:  
 f = R((-M.column(n)).list() + [1])  
 break  
 if n == 1:  
 k = ResidueFiniteField\_prime\_modn(p, names, im\_gen = -f[0], intp = p.smallest\_integer())  
 else:  
 q = characteristic\*\*(f.degree())  
 if q < Integer(2)\*\*Integer(16):  
 k = ResidueFiniteField\_givaro(p, q, names, f, characteristic)  
 else:  
 k = ResidueFiniteField\_ext\_pari(p, q, names, f, characteristic)   
 else: # Add support for primes in other rings later.  
 raise TypeError, "p must be a prime in the integers or a number field"  
 residue\_field\_cache[key] = weakref.ref(k)  
 return k

1. Get isomorphism to a GF(q). Do this by invert that matrix A to write down an explicit isomorphism to a GF(q).

New stuff in for William.

In [ResidueField](https://wiki.sagemath.org/ResidueField):

if n == 1:  
 ans = ResidueFiniteField\_prime\_modn(p, names, x, im\_gen = -f[0], intp = p.smallest\_integer())  
 else:  
 q = characteristic\*\*(f.degree())  
 if q < Integer(2)\*\*Integer(16):  
 ans = ResidueFiniteField\_givaro(p, q, names, x, f, characteristic)  
 else:  
 ans = ResidueFiniteField\_ext\_pari(p, q, names, x, f, characteristic)

NFResidueFieldHomomorphism's init:

def \_\_init\_\_(self, k, p, x, im\_gen):  
 """   
 INPUT:   
 k -- The residue field that is the codomain of this morphism.   
 p -- The prime ideal defining this residue field   
 x -- The element of the order that p belongs to that defined the minimal poly of k   
 im\_gen -- The image of x in k.   
   
   
 EXAMPLES:   
 We create a residue field homomorphism:   
 sage: K.<theta> = CyclotomicField(5)   
 sage: P = K.factor\_integer(7)[0][0]   
 sage: P.residue\_class\_degree()   
 4   
 sage: kk.<a> = P.residue\_field(); kk   
 Residue field in a of Fractional ideal (7)   
 sage: phi = kk.coerce\_map\_from(K.maximal\_order()); phi   
 Ring morphism:   
 From: Maximal Order in Cyclotomic Field of order 5 and degree 4   
 To: Residue field in a of Fractional ideal (7)   
 sage: type(phi)   
 <type 'sage.rings.residue\_field.NFResidueFieldHomomorphism'>   
 """  
 self.im\_gen = im\_gen  
 if not is\_FiniteFieldElement(im\_gen):  
 raise TypeError, "im\_gen must be a finite field element"  
 (<Element>self.im\_gen).\_set\_parent\_c(k)  
 self.p = p  
 self.x = x  
 self.R = PolynomialRing(k, 'x')  
 self.to\_list = x.coordinates\_in\_terms\_of\_powers()  
 ResidueFieldHomomorphism.\_\_init\_\_(self,Hom(p.number\_field().maximal\_order(), k, Rings())) # should eventually change to p.order()

call\_c\_impl:

return self.R(self.to\_list(x))(self.im\_gen)

lift:

return self.domain()(x.polynomial().change\_ring(self.domain().base\_ring())(self.x)) #polynomial should change to absolute\_polynomial?

In [ResidueFiniteField](https://wiki.sagemath.org/ResidueFiniteField)\_prime\_modn:

def \_\_init\_\_(self, p, name, x = None, im\_gen = None, intp = None):  
...  
 self.f = NFResidueFieldHomomorphism(self, p, x, im\_gen)

In [ResidueFiniteField](https://wiki.sagemath.org/ResidueFiniteField)\_ext\_pari:

def \_\_init\_\_(self, p, q, name, x, g, intp):  
...  
 self.f = NFResidueFieldHomomorphism(self, p, x, GF(q, name = name, modulus = g).gen(0))

In [ResidueFiniteField](https://wiki.sagemath.org/ResidueFiniteField)\_givaro:

def \_\_init\_\_(self, p, q, name, x, g, intp):  
...  
 self.f = NFResidueFieldHomomorphism(self, p, x, GF(q, name = name, modulus = g).gen(0)

# SEP 1: Calculus in SAGE

This is SAGE Enhancement Proposal (SEP) number 1.

## Authors

Alex Clemesha, David Joyner, Bobby Moretti, William Stein,

(This is a summary of emails from William Stein, Bobby Moretti, Joel Mohler, Alex Clemesha, Robert Bradshaw, David Joyner, David Harvey, Tom Boothby.)

## Preamble

A system for manipulating analytic functions in SAGE.

## Abstract

A Python class structure is needed to provide SAGE with the ability to perform calculus computations. Simple, intuitive syntax that does not break core Python semantics is a top priority. Basic calculus functions such as polynomials, trig functions, exponential functions, special functions, etc. should be manipulatable in SAGE using standard arithmetic and calculus operations.

## Copyright

Public domain

### Construction of calculus functions

To construct a calculus function, consider the assignment

sage: f = expr

where expr is an symbolic expression, i.e., an element of the set of basic calculus functions closed under the operations +,-,\*,/,^, composition, differentiation, and integration. The objects that generate this set (under the given operations) shall be called primitive functions. The set of primitive functions shall consist of basic analytic functions such as sine, cosine, polynomials, and exponentials. These functions shall be global instances of specific subclasses of the Function class, invoked via the Python \_\_call\_\_() method. When you call a Function it returns a symbolic expression.

There will be a class Function and a class SymbolicExpression.

The output of \_\_call\_\_\_() on an Function is a SymbolicExpression or a Function (in case of composition).

### A proposed class hierarchy

PrimitiveFunction  
 SymbolicExpression  
 SymbolicVariable  
 Constant  
 SymbolicBinaryOperation

### Operations on functions

#### Invoking other systems

In keeping with SAGE's philosophy of not reinventing the wheel, difficult operations may be defered to other systems (MAXIMA!) to which SAGE has an interface. The result of such computations should always be parsed and transformed into a valid SAGE expression. The result should not be a text string. Whenever possible, it should optionally be possible to compute using several systems, so one can compare results. Some operations that will be deferred to other systems in many cases:

* Symbolic integration
* Numerical integration
* Equality testing
* Expansion (of various forms)

#### Basic operations

The operations on basic functions shall include addition, subtraction, multiplication, division, and exponentiation. These will be implemented by overloading the associated Python operator. In addition to these arithmetic operations, functions may be created by composition, differentiation, and integration of Function objects.

Composition shall be invoked when one object of type Function is called as the argument of another function. Differentiation may be implemented in SAGE or passed on to another system (the most likely choice being Maxima), in either case resulting in a simplified expression. The simplification step must return a Function object, i.e., an symbolic expression, and would likely be implemented in Maxima. The simplification can be undone by performing the expand operation, which is defined below. Integration shall be computed entirely within a separate system (again, most likely Maxima) and a simplified result shall be passed back to SAGE in the same manner as described above for differentiation.

Other operations shall include algebraic simplification and various types of expansion are implemented using a separate system as is isolation of one variable.

### Implementation of basic operations

As discussed above, binary operation shall be implemented by overloading the relevant Python operators. Unary operations shall be member methods of the Function class. For example:

sage: f = x \* sin(3\*x^2 + y) + cos(y)  
sage: f.derivative(y)  
2\*x\*cos(3\*x^2 + 2\*y) - sin(y)

## Substitution

Substitution shall be another key operation on functions. By substitution, we mean the mathematical operation of replacing a variable (or set of variables) with either a fixed value, or with another symbolic expression. In this sense, the operation of substitution represents two mathematical concepts: evaluation of a function at a given element of its domain set, and algebraicsubstitution.

### Algorithm

This operation shall be implemented as a member method of the Function class known as substitute. (It is \*key\* that people find the substitute operation quickly). This method shall be implemented as a recursive algorithm. If the Function is built out of some operation, then the algorithm calls itself on each of the operands. The algorithm terminates when the Function object in question is either a primitive variable or a constant. If the termination happens to be on a variable, the variable is substituted with the replacement expression.

### Implementation

The substitute method shall be called in one of the following ways: Let f be an symbolic expression.

* Named parameters. This returns f but with 2 substituted for x and z^2 substitued for w.

f(x=2, w=z^2)

* With a dict. The keys are the variable name, the values are the expressions to be substituted in.

sage: f.subsitute({x: 2, y: z^2})

* With this notation:

sage: f.substitute(x=2, y=z^2)

* Also define the function method:

sage: f.function(\*args)

This method shall return a callable without named parameters version of f, which otherwise works in the same way. Output is result of subs. this just another formal function, but with a call method. Also, function((vars...),expr) makes an evaluatable function.

For example:

g = sin(x)\*cos(x+y+3)  
f = g.function(x,y)

g = f(x=x,y=7); g  
///  
sin(x)\*cos(x+10)

g(x=10)  
///  
sin(10)\*cos(20)

f(x=5,y=7)  
///  
sin(5)\*cos(15)

We could possibly add the following type of function as well:

or dummy = sin(x\*) \* cos(x+y+3) f = dummy.function(x,y) Also, with my proposal one could also already type f = function( (x,y), sin(x)\*cos(x+y+3) ) Here this would be implemented via:

def function( vars, expr):  
 return expr.function(\*vars)

## Preparsing for defining a callable symbolic expression

f(x,y) = sin(x+y) \* x + cos(y^2)\*z

contour\_plot(f, (-1,2), (-1,2))

This would get converted by the preparser to:

f = (sin(x+y) \* x + cos(y^2 + z^3)).function(x,y)

f.integral(z)  
///  
sin(x+y) \* x\*z + cos(y^2)\*(z^2/2)

plot(f(2,5), (z, 0, 5))  
plot(f(y=5), (x, 0, 3), (z, 0, 5))

Mathematica:

Plot[f[2,5,z], {z, 0, 5}]

Another important niceness:

h = x^3 + y^2 + sin(z)

h is a symbolic expression.

This gets preparsed to:

z = h.function(x,y)

or even

h(x,y) = h

h(5,6)   
///  
5^3 + y

## Motivation

For SAGE to have wide-spread acceptance in calculus, ease-of-use is critical. The current situation, illustrated with lambda functions, [ElementaryFunctions](https://wiki.sagemath.org/ElementaryFunctions), or built-in Python functions is inadequate.

## Rationale

Though Maple and Mathematica have functionality with regard to calculus computations which SAGE is currently lacking, something better is needed. The syntax is still frustrating and unnatural for many new users, even for these sophisticated CAS's. Many find these systems very counterintuitive to do basic calculus. Part of the problem is the difference between mathematical functions and computer language functions. This SEP asserts that SAGE can do better and can provide a simpler system.

Backwards compatibility - This will be a new class and will provide new functionality which will not interfere with previous implementations.

## References

* Wiki page: <http://sage.math.washington.edu:9001/BasicCalculus>
* Emails in the "Sage (Fwd)" and "SAGE calculus" threads <http://groups.google.com/group/sage-devel>
  + [More discussion](http://groups.google.com/group/sage-devel/browse_thread/thread/28295fa066dd4e42/88f0c8af548e8a11#88f0c8af548e8a11)
  + [Initial discussion](http://groups.google.com/group/sage-devel/browse_thread/thread/96647186db5aee76/8c60608b9323de5c#8c60608b9323de5c)

# Cantor

goto [Tips](https://wiki.sagemath.org/Tips)

Contents

1. [Cantor](https://wiki.sagemath.org/Cantor#Cantor-1)
2. [Install Cantor at Ubuntu Karmic](https://wiki.sagemath.org/Cantor#Install_Cantor_at_Ubuntu_Karmic)
3. [Build and Install Cantor](https://wiki.sagemath.org/Cantor#Build_and_Install_Cantor)
4. [Using Sage with Cantor](https://wiki.sagemath.org/Cantor#Using_Sage_with_Cantor)
5. [Links](https://wiki.sagemath.org/Cantor#Links)

## Cantor

Cantor is a front-end for many mathematical software applications like Sage, Maxima, R and KAlgebra. Cantor belongs to [The KDE Education Project](http://edu.kde.org/) and lets you use your favorite mathematical applications from within a nice KDE-integrated Worksheet Interface. It offers assistant dialogs for common tasks and allows you to share your worksheets with others.

Cantor's main features are:

* Nice Worksheet view for evaluating expressions
* View of plotting results inside the worksheet or in a separate window
* Easy plugin based structure to add different Backends
* [GetHotNewStuff](http://ghns.freedesktop.org/) integration to upload/download example worksheets
* Typesetting of mathematical formulas using LaTeX
* Backend aware syntax highlighting
* Plugin based assistant dialogs for common tasks (like integrating a function or entering a matrix)

## Install Cantor at Ubuntu Karmic

For ubuntu karmic you just 3 commands in a terminal:

sudo add-apt-repository ppa:kubuntu-ppa/beta

sudo apt-get update

sudo apt-get install cantor

It will update all the KDE-packages to version 4.4, but this is useful for having no troubles with cantor.

## Build and Install Cantor

As Cantor is part of KDE Edu since version 4.4, most Linux distributions will already have a package for it. Either combined with other apps in one kdeedu package or split into a separate cantor package.

If you are using an older distribution, you can build cantor from source:

Cantor depends upon kdelibs, so in order to build Cantor, you must have kdelibs development files.

If you have a Linux box, then you can install the kdelibs-devel package from your distribution. If you have windows, then you can use [The KDE on Windows Project](http://windows.kde.org/), please note that there is already a Cantor.exe ported to windows, so is not necesary build the application. Otherwise, you can [build and install kdelibs from sources](http://techbase.kde.org/Getting_Started/Build/KDE4#kdelibs).

The following steps can be used to build Cantor for a Linux distribution.

* First obtain the sources

svn co svn://anonsvn.kde.org/home/kde/trunk/KDE/kdeedu/cantor

* Replace the the content of cantor/CMakeLists.txt with this:

project(cantor)  
  
# search packages used by KDE  
find\_package(KDE4 4.3.80 REQUIRED)  
if(WIN32)  
 find\_package(KDEWIN32 REQUIRED)  
 # detect oxygen icon dir at configure time based on KDEDIRS - there may be different package installation locations  
 execute\_process(COMMAND "${KDE4\_KDECONFIG\_EXECUTABLE}" --path icon OUTPUT\_VARIABLE \_dir ERROR\_QUIET OUTPUT\_STRIP\_TRAILING\_WHITESPACE)  
 file(TO\_CMAKE\_PATH "${\_dir}" \_\_dir)  
 find\_path(KDE4\_ICON\_DIR oxygen PATHS  
 ${\_\_dir}  
 )  
 message(STATUS "using oxygen application icons from ${KDE4\_ICON\_DIR}")  
else(WIN32)  
 set (KDE4\_ICON\_DIR ${CMAKE\_INSTALL\_PREFIX}/share/icons)  
endif(WIN32)  
  
include(KDE4Defaults)  
include(MacroLibrary)  
  
if(NOT WIN32)  
 macro\_optional\_find\_package(LibSpectre)  
 macro\_log\_feature(LIBSPECTRE\_FOUND "libspectre" "A PostScript rendering library" "http://libspectre.freedesktop.org/wiki/" FALSE "${LIBSPECTRE\_MINIMUM\_VERSION}" "Support for rendering EPS files in Cantor.")  
  
 if(LIBSPECTRE\_FOUND)  
 set( WITH\_EPS On )  
 else(LIBSPECTRE\_FOUND)  
 set( WITH\_EPS Off )  
 endif(LIBSPECTRE\_FOUND)  
else(NOT WIN32)  
 set( WITH\_EPS Off )  
endif(NOT WIN32)  
  
include\_directories( ${KDE4\_INCLUDES} ${QT\_INCLUDES} )  
  
add\_subdirectory( src )  
add\_subdirectory( icons )  
add\_subdirectory( tests )

* Type this 4 commands:

cd cantor  
mkdir build  
cd build  
cmake ../

You can choose where to install with this param DCMAKE\_INSTALL\_PREFIX For example:

cmake -DCMAKE\_INSTALL\_PREFIX=/usr/ ../

* Type this command

make install

## Using Sage with Cantor

After starting Cantor you will be prompted with all the supported backends. If sage is not on the list, you may have to check the settings, and correctly set the path to the sage executable. If you check the "make this the default backend" box, this backend will be preselected in the future.

The Cantor worksheet is quite similar to the one offered by Sage notebook. You can run any Sage commands by entering it in the cell and pressing Shift+Enter.

If you evaluate "? command" the help for the specific command will show up in the help panel on the right, pressing <Tab> will show you possible completions for the current command.

Cantor offers some comfort features, like typesetting your results using [LaTex](https://wiki.sagemath.org/LaTex), or offering dialogs for commonly used functions.

Once you've finished working on your worksheet, you can easily share it with others, by publishing it on kde-files.org. You just select the "Publish Worksheet" entry in File menu, and follow the assistant. Once completed, your new worksheet will show up in the "Download Example" dialog.

## Links

* Official Site - <http://edu.kde.org/cantor/>
* Author's Blog - <http://arieder.wordpress.com/>

# cmake

## Description

cmake is an extensible, open-source system that manages the build process in an operating system and compiler independent manner. Unlike many cross-platform systems, CMake is designed to be used in conjunction with the native build environment. Simple configuration files placed in each source directory (called CMakeLists.txt files) are used to generate standard build files (e.g., makefiles on Unix and projects/workspaces in Windows MSVC) which are used in the usual way. CMake can compile source code, create libraries, generate wrappers, and build executables in arbitrary combinations.

* <http://www.cmake.org/HTML/Features.html>

## License

BSD like:

* <http://www.cmake.org/HTML/Copyright.html>

## SPKG Maintainers

* Josh Kantor?

## Upstream Contact

?

## Dependencies

## Changelog

* First provided as an experimental spkg by Josh Kantor?

# Quaternion Algebras

We introduce quaternion algebras, which are noncommutative analogues of quadratic field extensions and hence are ubiquitous in mathematics. We will give a demonstration of existing algorithms for computing with quaternion algebras, as currently implemented in Magma, and then discuss possible future directions in this area from both the theoretical and practical points of view.

To implement these algorithms in SAGE, we need:

* maximal orders for rings of integers
* pseudo-bases (algorithms for dedekind rings)
* places of a number field
* p-adics
* completions
* linear algebra over Z
* hermite normal forms
* fast algorithms for enumerating elements of a lattice of a given norm (most pressing)
* group theory

To run the Magma demo [magma-quat-demo.m](https://wiki.sagemath.org/Algorithms_for_quaternion_algebras?action=AttachFile&do=view&target=magma-quat-demo.m), you need the latest version of Magma, v.2.13-4 or -5. You can also run them from the Magma notebook, <http://magma.maths.usyd.edu.au/calc/>.

Some quick references for algorithms for quaternion algebras:

* Montserrat Alsina, Pilar Bayer, "Quaternion orders, quadratic forms, and Shimura curves"
* David Kohel, "Endomorphism rings of elliptic curves over finite fields", available at <http://www.maths.usyd.edu.au/u/kohel/res/index.html>
* Marie-France Vigneras, "Arithmetique des algebres de quaternions"
* John Voight, "Quadratic forms and quaternion algebras: Algorithms and arithmetic, available at <http://www.ima.umn.edu/~voight/>
* John Voight, "Algorithms for quaternion algebras", in preparation!

# Combinat/01

## **Design discussion for permutations and discrete functions**

This is about permutations, and more generally about functions between finite sets.

Desirable features:

(1) Python mantra: an object which looks like a list should behave like a list.

**I.e. if the users sees**

* sage: p = ...  
   sage: p  
   [4,1,3,2]

**Then he will expect**

* sage: p[0]  
   4

(2) The user should be able to manipulate permutations (functions) of

**any finite set, and manipulate them as is, without reindexing**

* sage: F = Functions([3,4,8])  
   sage: F.list()  
   [3,3,3]  
   [3,3,4]  
   [3,3,8]  
   ...  
   [8,8,8]  
   sage: p = F([8,3,4])  
   [8,3,4]

**In particular, whatever the syntax is, one want to be able to do**

* sage: p of 3  
   8  
   sage: p of 3 = 4  
   sage: p  
   [4,3,4]

It is important to have short notations for readability of the algorithms. Write access (with surrounding mutability mantra) is necessary as well.

(3) Permutations(n) should be Permutations([1,...,n])

(4) Internally, permutations could be implemented as permutations of

* [0...n-1] for speed (future cythonization)
  + It's in fact a typical design situation: internal implementations
  + using 0...n indexing (cf. matrix, [FreeModule](https://wiki.sagemath.org/FreeModule), dynkin diagrams, Family, ...) + views on them indexed by whatever is convenient for the user.

(5) Backward compatibility

## **Potential solutions:**

**- Current one**

* sage: p = DiscreteFunction([3,1,2])  
   sage: p[0]  
   3  
   sage: p[0] = 1; p[1] = 3 # actually not implemented  
   sage: p  
   [1,3,2]
* Caveat: breaks (2)

**- Use indexed access starting at 1 (or whatever the smallest letter is)**

* sage: p = DiscreteFunction([3,1,2])  
   sage: p[1]  
   3  
   sage: p[1] = 1; p[2] = 3 # actually not implemented  
   sage: p  
   [1,3,2]
* Breaks (1), (5). This also means that any method that works on a list will not work on a permutation. So one would have to define a separate methods for a permutation.

**- Use functional notation**

* sage: p = DiscreteFunction([3,1,2])  
   sage: p(1)  
   3  
   sage: p(1) = 1; p(2) = 3 # actually not implemented  
   sage: p  
   [1,3,2]

**Caveat: requires patching Sage (no {{{\_\_setcall\_\_}}} analoguous to {{{\_\_setitem\_\_}}). In the mean time, we could use p.set(1, 1) (lengthy notation). Another (huge) disadvantage is that {{{\_\_call\_\_\_}}} is much slower than {{{\_\_getitem\_\_}}}**

* sage: p = Permutations(130).random\_element()  
   sage: %timeit p[83]  
   1000000 loops, best of 3: 808 ns per loop  
   sage: %timeit p.\_\_getitem\_\_(83)  
   1000000 loops, best of 3: 561 ns per loop  
   sage: %timeit p(83)  
   100000 loops, best of 3: 3.27 µs per loop

Actually something that I have always wonder about: why don't we have Permutation\_class inheriting from list/tuple? This would speed things up.

* sage: from sage.combinat.permutation import Permutation\_class  
   sage: class MyPermutation(list, Permutation\_class):  
   ... def \_\_repr\_\_(self):  
   ... return '[%s]' % super(MyPermutation,self).\_\_repr\_\_()[1:-1]  
   sage: q = MyPermutation(Permutations(17).random\_element()); q  
   [2, 8, 12, 7, 9, 3, 11, 15, 14, 4, 5, 1, 6, 17, 16, 13, 10]  
   sage: %timeit q[11]  
   10000000 loops, best of 3: 89 ns per loop  
   sage: %timeit q.\_\_getitem\_\_(11)  
   1000000 loops, best of 3: 304 ns per loop  
   sage: %timeit q(11)  
   1000000 loops, best of 3: 1.2 µs per loop

- ...

# Extra documentation

Describe extra documentation (for the standard components of SAGE):

The following docs are included:

* gap-4r4.8
* gnuplot
* ipython
* linbox
* macaulay2
* maxima
* mpfr
* ntl
* numeric
* octave
* pari
* gmp-4.2.1
* kash
* matplotlib
* mwrank
* numpy
* python-2.5
* readline
* singular
* zodb

# GmoduleSEP

= G-module SEP =

AUTHORS: David Joyner, David Roe<<BR>>

=== OBJECTIVE ===

Create

\*1. a Python class data structure representing a finite rank G-module R, where G is a given group (already defined in SAGE),

\*2. a Python class representing a ``group-with-action'' G on a set X, where X is a given set (or list?).

=== PROPERTIES ===

\*1. The G-module class should be able to:

\* interface with the MeatAxe data types (see section 3.1.2 of the MeatAxe manual),

\* structure inherited by FreeModule class,

\* have default construction of ZZ[G] but allow for constructing F[H], where F is any field and H is a group containing G,

\* implements a G-set X by simply constructing the free abelian group ZZ[X] = ZZ^X,

\* interface with the HAP free ZZ[G]-module data type (Here is some info from HAP's manual: "In order to represent a word w in a free ZG-module M of rank n we use a list of integer pairs w=[ [i1,t1], [i2,t2], ..., [ik,tk] ]. The integers ij lie in the range between -n and n and correspond to the free ZZ[G]-generators of M and their additive inverses. The integers tj are positive and correspond to the group element eltsG[tj]. The (partial) listing eltsG may contain duplicate copies of elements of G.")

\* interface with HAP's FpGModule(A,G) construction when G is a p-group and M = GF(p)[G]

\*2. The group-with-action class should be able to:

\* implement group actions of a Galois group acting on a field,

\* if R is an instance of the G-module class then G is a group-with-action on R.

# **Ideas of things to Implement in SAGE**

* ([Tom Boothby](https://wiki.sagemath.org/TomBoothby)) [NotebookWiki](https://wiki.sagemath.org/NotebookWiki): A wiki version of the notebook that could be a robust, distrubuted development enviroment and version control system. Tom's end goal: move all development of SAGE into a wiki-like notebook interface, and make the system such that it could be easily used for any project in Python (and other languages?).
* checksums on downloaded files
* {{{>> It has a file for every single SAGE package that was ever

>> successfully installed. >> Adding an upgrade.a.b.c.d file to that directory would be easy, but >> potentially >> confusing since it has a different meaning than those files. > > Hmmm...to clarify, I have upgraded, successively, beginning with > sage-1.2.3.1 to my current sage-1.3.5.3. There are no "duplicate" > packages (same name, different version) in spkg/installed. Does that > mean that between 1.2.3.1 and 1.3.5.3, there were no package upgrades? >

I think I'm wrong -- it used to be that installed kept everything. Now it doesn't. Sorry for the confusion. I'll probably make an

* spkg/installed/previous

directory where the old ones get put.}}}

done \* Finite fields. Fix it so this isn't needed: {{{sage: from sage.rings.finite\_field import [FiniteField](https://wiki.sagemath.org/FiniteField)\_ext\_pari sage: k = [FiniteField](https://wiki.sagemath.org/FiniteField)\_ext\_pari(9,modulus=x^2+x+2) sage: (k.0).charpoly() x^2 + x + 2}}}

\* Make it so 2d scenes described by SAGE's ploting functions, e.g., line, polygon, point, etc., can be rendered using tachyon somehow. I.e., all the objects would have 3d analogues and lie in a plane.

\* "Port" Helena Verrill's <http://www.math.lsu.edu/~verrill/fundomain/> to SAGE. It's currently written in Java and also MAGMA. It will be incredibly pretty if built on top of SAGE/matplotlib's plotting functionality, and SAGE's tachyon ray tracer (!). (In progress at the MSRI workshop 06 coding sprints...)

\* Make it so all matrices can be coerced to MAGMA matrices (this is pretty good now). Add a flag to the matrix.py file, so echelon forms can be done by MAGMA by default instead of SAGE. This will speed some things up for applications that build on SAGE's linear algebra, so that these apps can be built. These apps will provide good benchmarking coding, etc., to make developing good linear algebra in SAGE easier.

\* (Didier Deshommes:) I would love to see an option where while re-building 1 or more modules with sage -br, you can also verify that your modifications still pass unit tests for pyx, pyxe and py files.

Currently I have a hack in my setup.py ( and a patch here: <http://sage.math.washington.edu/home/dfdeshom/sage/devel/sage/setup-hack.patch.txt> ) that makes this possible for pyx files (and therefore for pyxe files?), but this seems more complicated for native .py files. Does anyone know how to hijack the build\_py process to run doctests on modified .py files?

\* See the more structured project list (somebody should move that list here):

* <http://modular.math.washington.edu/sage/projects/>

\* todo:

* Jones database -- added Richards (?) as co-author in documentation, etc.

\* todo:

* dvipng -- better error about dependency
  + -- don't build if already have.

\* feature request: make an upgrade\_packages() funtion that upgrades

* all installed optional packages to the latest versions. This would, in fact, be very easy to write with just a little Python code using the install\_package and optional\_packages() functions.

\* todo: sage build \*still\* doesn't check that you have flex/bison installed. Argh! Why?

\* It would be nice to be able to do this. Need to implement more general division algorithm?? iterated poly rings:

[Toggle line numbers](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#)

[1](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-dc660fc2fe355e0c538752bbb20b05510d862777_1) # Worksheet '\_scratch\_' (2006-07-28 at 12:14)  
 [2](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-dc660fc2fe355e0c538752bbb20b05510d862777_2) R.<x> = PolynomialRing(QQ)  
 [3](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-dc660fc2fe355e0c538752bbb20b05510d862777_3) S.<y> = PolynomialRing(R)  
 [4](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-dc660fc2fe355e0c538752bbb20b05510d862777_4) T.<z> = S.quotient(y^3-3)  
 [5](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-dc660fc2fe355e0c538752bbb20b05510d862777_5) z\*z\*z  
 [6](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-dc660fc2fe355e0c538752bbb20b05510d862777_6) Traceback (most recent call last):  
 [7](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-dc660fc2fe355e0c538752bbb20b05510d862777_7) TypeError: unable to find a common parent for 1 (parent: Fraction Field of Univariate Polynomial Ring in x over Rational Field) and 1 (parent: Univariate Polynomial Ring in y over Univariate Polynomial Ring in x over Rati...

\* feature request: paste into notebook should resize box.

\* possible change: sage\_eval -- should it only take strings as input?!

\* feature request: showing sourcecode of functions in pyrex files. (!!)

\* feature request: add imshow functionality to plot

\* in online simple sage calculator "? help" doesn't work.

\* e^(I\*pi) is perhaps lame.

\* Macaulay2 source package:

-- it still doesn't detect the SAGE readline,  
 so you have to have readline installed system-wide.  
 apt-get install libreadline-dev  
 -- it depends on etags  
 apt-get install exuberant-ctags

\* question: is this good or bad? maybe I don't understand the rigor

* of MPFR well.

[Toggle line numbers](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#)

[1](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-31311ec546436a0ef692a0e1ece2e5bd0bc65a0f_1) sage: R = RealField(10, rnd='RNDD')  
 [2](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-31311ec546436a0ef692a0e1ece2e5bd0bc65a0f_2) sage: p = R(pi)  
 [3](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-31311ec546436a0ef692a0e1ece2e5bd0bc65a0f_3) sage: p  
 [4](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-31311ec546436a0ef692a0e1ece2e5bd0bc65a0f_4) 3.1406  
 [5](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-31311ec546436a0ef692a0e1ece2e5bd0bc65a0f_5) sage: p.str(2)  
 [6](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-31311ec546436a0ef692a0e1ece2e5bd0bc65a0f_6) '11.00100100'  
 [7](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-31311ec546436a0ef692a0e1ece2e5bd0bc65a0f_7) sage: q = p.sin()  
 [8](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-31311ec546436a0ef692a0e1ece2e5bd0bc65a0f_8) sage: q.str(2)  
 [9](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-31311ec546436a0ef692a0e1ece2e5bd0bc65a0f_9) '0.00000000001111110110'

\* (reported by Justin Walker -- 2006-06-07)

> The install worked, but a bunch of the "per-user" stuff got dropped  
> into his home directory, presumably as expected, but owned by root,  
> so he can't modify the files.  
I'm a little puzzled about how that happened. Was he installing SAGE  
as user root but with his HOME environment variable set to  
his home directory? If you install SAGE as root, then  
there is no knowledge of any user's home directory in the  
environment. At worst, "per-user" stuff will be copied to  
the root user's /root directory.  
Probably the user wasn't logged in as root, but used sudo:  
was@sha:~$ sudo echo $HOME  
/home/was  
This is a tricky problem, because some small part of the install  
process requires starting SAGE, but starting SAGE requires having  
some startup files in $HOME. But this is definitely a bug, and  
I'll add it to the list.  
(from justin) Thanks! It's easy to recover from, modulo figuring  
out what to change, but it would be good to fix, if possible. One  
possible fix is to force $HOME to a known value (/dev/null? :-}) and  
not set up the HOME files in this case.  
> Does SAGE do this automatically when it starts up and discovers that  
> the environment it expects (the ~/.sage stuff) is either out of date  
> or non-existent?  
That seems like a very reasonable idea.  
> I suppose there are some that run as root by default, but that  
> strikes me as wrong-headed, and unsupported :-}  
SAGE could print a message on any startup as root warning the user  
that running SAGE as root is different. I like this idea.

\* It should be very easy to change properties of a graphics object, e.g.,

* change the rgbcolor without recomputing all the points!

\* 2006-05-21: Error when doing "sage -i " with abs path; Workaround: Just go to the correct directory and type "sage -i gap\_packages-4.4.7\_2.spkg" to test it out.

\* 2006-05-16:

* The error message for "sage -bf" sucks.

\* 2006-05-15:

* tracebacks include path of place where file was built?? this is hard coded? how to fix??? Maybe just do a substituion when doing the display of docs in sage/server/support.py

\* 2006-05-08: sajax: prompt doesn't change when change mode.

\* 2006-05-02: maxout.gnuplot randomly appears in my home directory

* after running "make test". Very annoying.

\* in some cases echelon forms with big entries -- totally dominated

* by rational reconstruction; this should be "easy" to fix by doing something much better (e.g., using GMP). Good example:

[Toggle line numbers](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#)

[1](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-8d71df9b8ae935ac7f4fc93c69102d8c4f649e74_1) sage: set\_verbose(2)  
 [2](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-8d71df9b8ae935ac7f4fc93c69102d8c4f649e74_2) sage: victor\_miller\_basis(200,50)

\* 2006-04: Should get better error message and caching, etc:

[Toggle line numbers](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#)

[1](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-0bced00a8c45536285993181f23d613791f0bddb_1) sage: ... what is F???  
 [2](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-0bced00a8c45536285993181f23d613791f0bddb_2) sage: time F.tropical\_basis(check = False)  
 [3](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-0bced00a8c45536285993181f23d613791f0bddb_3) /bin/sh: line 1: 3699 Aborted gfan\_tropicalbasis 2>/dev/null  
 [4](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-0bced00a8c45536285993181f23d613791f0bddb_4) CPU times: user 0.00 s, sys: 0.00 s, total: 0.00 s  
 [5](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-0bced00a8c45536285993181f23d613791f0bddb_5) Wall time: 3.03  
 [6](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-0bced00a8c45536285993181f23d613791f0bddb_6) \_30 = []  
 [7](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-0bced00a8c45536285993181f23d613791f0bddb_7) sage: time F.tropical\_basis()  
 [8](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-0bced00a8c45536285993181f23d613791f0bddb_8) CPU times: user 0.00 s, sys: 0.00 s, total: 0.00 s  
 [9](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-0bced00a8c45536285993181f23d613791f0bddb_9) Wall time: 0.00  
 [10](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-0bced00a8c45536285993181f23d613791f0bddb_10) \_31 = []

\* This is a good example to put in the docs.

[Toggle line numbers](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#)

[1](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_1) The toric ideal of the twisted cubic in projective 3 space:  
 [2](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_2) sage: R.<a,b,c,d> = PolynomialRing(QQ, 4)  
 [3](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_3) sage: I = ideal(b^2-a\*c, c^2-b\*d, a\*d-b\*c)  
 [4](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_4) sage: F = I.groebner\_fan(); F  
 [5](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_5) Groebner fan of the ideal:  
 [6](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_6) Ideal (c^2 - b\*d, -1\*b\*c + a\*d, b^2 - a\*c) of Polynomial Ring in a, b, c, d over Rational Field  
 [7](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_7) sage: F.reduced\_groebner\_bases ()  
 [8](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_8) [[-1\*c^2 + b\*d, -1\*b\*c + a\*d, -1\*b^2 + a\*c],  
 [9](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_9) [c^2 - b\*d, -1\*b\*c + a\*d, -1\*b^2 + a\*c],  
 [10](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_10) [c^2 - b\*d, b\*c - a\*d, -1\*b^2 + a\*c, -1\*b^3 + a^2\*d],  
 [11](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_11) [c^2 - b\*d, b\*c - a\*d, b^3 - a^2\*d, -1\*b^2 + a\*c],  
 [12](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_12) [c^2 - b\*d, b\*c - a\*d, b^2 - a\*c],  
 [13](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_13) [-1\*c^2 + b\*d, b^2 - a\*c, -1\*b\*c + a\*d],  
 [14](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_14) [-1\*c^2 + b\*d, b\*c - a\*d, b^2 - a\*c, -1\*c^3 + a\*d^2],  
 [15](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_15) [c^3 - a\*d^2, -1\*c^2 + b\*d, b\*c - a\*d, b^2 - a\*c]]  
 [16](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_16) sage: F.fvector()  
 [17](https://wiki.sagemath.org/Ideas_for_Features_to_Implement#CA-be33dee5b96cc24c90c84528eeb74100efab8dd5_17) (1, 8, 8)

\* ([MartinAlbrecht](https://wiki.sagemath.org/MartinAlbrecht)) Reference Manual -> Wiki -> Refernce Manual conversion to allow contribution to the docs via the Wiki interface.

\* ([MartinAlbrecht](https://wiki.sagemath.org/MartinAlbrecht)) Adding a "Open a notebook with these examples" button could be added to every Wiki page automatically if it has code blocks marked as #!sage (which would inherit from #!python). So viewers can play with/test examples directly.

\* ([DavidHarvey](https://wiki.sagemath.org/DavidHarvey)) There are some good ideas about matrix multiplication in these two articles: <http://crpit.com/confpapers/CRPITV26ElGindy.pdf#search=%22strassen%20without%20padding%22> (actually I know one of the guys who wrote this), and <http://www.cs.duke.edu/~alvy/papers/sc98/index.htm>. In particular they discuss strategies for (a) dealing with odd number of rows/columns, and (b) cache efficient ordering of the multiplication.

\* ([DavidHarvey](https://wiki.sagemath.org/DavidHarvey)) Arrrggghhh. Matplotlib is painfully slow. There are some pointers at <http://matplotlib.sourceforge.net/faq.html#SLOW> to follow up about this. They talk about 30x performance improvements based on configuration alone.

\* ([KiranKedlaya](https://wiki.sagemath.org/KiranKedlaya)) Computing the m-th power of a power series over a Q-algebra should return the expected result when m is rational. Right now it coerces m to an integer first.

\* ([RonanPaixao](https://wiki.sagemath.org/RonanPaixao)) Sage should have support for Dirac's delta function and Heaviside's step functions.

The step function is one defined as step(x)=Ú0 1 if if x<0 x>0

In Matlab the function is called heaviside()

Dirac's delta function (sometimes called impulse function) is defined as delta(x)={∞0ififx=0x≠0delta(x)={∞ifx=00ifx≠0

Some interesting properties of those functions are: ∫+∞−∞δ(x)dx=1∫−∞+∞δ(x)dx=1

∫δ(x)dx=step(x)∫δ(x)dx=step(x)

∫+∞−∞f(x)∗δ(x−k)dx=f(k)∫−∞+∞f(x)∗δ(x−k)dx=f(k)

∫+∞−∞f(x)∗step(x−a)dx=∫+∞af(x)dx∫−∞+∞f(x)∗step(x−a)dx=∫a+∞f(x)dx

A function can be transformed from continuous time to discrete time (sampling) with Dirac's comb, which is the sum of equally-spaced impulse functions.

A similar function is Kronecker's delta function: delta(x)={10ififx=0x≠0delta(x)={1ifx=00ifx≠0

References: <http://en.wikipedia.org/wiki/Dirac_delta_function> <http://en.wikipedia.org/wiki/Dirac_comb> <http://en.wikipedia.org/wiki/Sampling_(signal_processing>) <http://en.wikipedia.org/wiki/Kronecker_delta>

# **Malloc Replacements**

This page describes possible malloc replacements to be used for memory management in SAGE which is not directly controlled by Python such as manual malloc/realloc calls, the GMP library etc..

## **Introduction**

* Benchmarks performed on my ([malb](https://wiki.sagemath.org/MartinAlbrecht)) 1200 Mhz Pentium 3 notebook running vanilla SAGE 1.4

## **malloc**

* "By default GMP uses malloc, realloc and free for memory allocation, and if they fail GMP prints a message to the standard error output and terminates the program." (<http://www.swox.com/gmp/manual/Custom-Allocation.html>)
* GPL

[Toggle line numbers](https://wiki.sagemath.org/MallocReplacements#)

[1](https://wiki.sagemath.org/MallocReplacements#CA-945f36505c474fe5ed0ce23f694e7d369d25b0a6_1) sage: x = 3; y = 5  
 [2](https://wiki.sagemath.org/MallocReplacements#CA-945f36505c474fe5ed0ce23f694e7d369d25b0a6_2)   
 [3](https://wiki.sagemath.org/MallocReplacements#CA-945f36505c474fe5ed0ce23f694e7d369d25b0a6_3) sage: timeit \_ = x + y  
 [4](https://wiki.sagemath.org/MallocReplacements#CA-945f36505c474fe5ed0ce23f694e7d369d25b0a6_4) 100000 loops, best of 3: 7.37 mircoseconds per loop

## **PyMem\_Malloc**

The Python memory manager does keep track of some previous allocated, unused, but not yet freed blocks so it should be faster than malloc itself. It is currently used in GMP and Pyrex code in SAGE.

## **OMalloc**

* GPLed (not announced yet)
* faster than malloc/PyMem\_Malloc etc.

[Toggle line numbers](https://wiki.sagemath.org/MallocReplacements#)

[1](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_1) sage: x = 3; y = 5 # malloc (GMP default)  
 [2](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_2) sage: timeit \_ = x + y  
 [3](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_3) 100000 loops, best of 3: 8.14 microseconds per loop  
 [4](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_4) sage: timeit \_ = x + y  
 [5](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_5) 100000 loops, best of 3: 8.05 microseconds per loop  
 [6](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_6)   
 [7](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_7) sage: sage.rings.integer.pmem\_malloc()  
 [8](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_8)   
 [9](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_9) sage: timeit \_ = x + y # PyMem\_Malloc  
 [10](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_10) 100000 loops, best of 3: 8.27 microseconds per loop  
 [11](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_11) sage: x = 3; y = 5  
 [12](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_12) sage: timeit \_ = x + y  
 [13](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_13) 100000 loops, best of 3: 8.21 microseconds per loop  
 [14](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_14)   
 [15](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_15) sage: omalloc\_malloc()   
 [16](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_16)   
 [17](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_17) sage: timeit \_ = x + y # omalloc  
 [18](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_18) 100000 loops, best of 3: 7.55 microseconds per loop  
 [19](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_19) sage: timeit \_ = x + y  
 [20](https://wiki.sagemath.org/MallocReplacements#CA-90fdc582cce7716660a3c73272944d66caef6297_20) 100000 loops, best of 3: 7.55 microseconds per loop

Larger integers:

[Toggle line numbers](https://wiki.sagemath.org/MallocReplacements#)

[1](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_1) sage: x =   
 [2](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_2) 45234723458934634056310653251305043576435013985134515643531450714359863148659136951369851836545643565  
 [3](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_3)   
 [4](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_4) sage: y =   
 [5](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_5) 23423964523462384234234234872345623645623452345283548234652365482348562346523642348263486234954534554  
 [6](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_6)   
 [7](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_7) sage: timeit \_ = x + y #malloc  
 [8](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_8) 100000 loops, best of 3: 8.3 microseconds per loop  
 [9](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_9)   
 [10](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_10) sage: sage.rings.integer.pmem\_malloc()  
 [11](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_11)   
 [12](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_12) sage: timeit \_ = x + y #PyMem\_Malloc  
 [13](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_13) 100000 loops, best of 3: 8.58 microseconds per loop  
 [14](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_14)   
 [15](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_15) sage: omalloc\_malloc()   
 [16](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_16)   
 [17](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_17) sage: timeit \_ = x + y # omalloc  
 [18](https://wiki.sagemath.org/MallocReplacements#CA-810b6e5d74305474d71f8631b0ca9b037e4e73ef_18) 100000 loops, best of 3: 7.58 microseconds per loop

## **DLMalloc**

* LGPL
* used by OMalloc
* unsure what it does

## **default libCF Memory Manager**

* GPL
* seems to be faster for Integer addition than Python's manager

[Toggle line numbers](https://wiki.sagemath.org/MallocReplacements#)

[1](https://wiki.sagemath.org/MallocReplacements#CA-8edc33073c91e5c602526a95ad5eeadded8e97f4_1) sage: from sage.memory import \* #only in my (malb) working copy  
 [2](https://wiki.sagemath.org/MallocReplacements#CA-8edc33073c91e5c602526a95ad5eeadded8e97f4_2)   
 [3](https://wiki.sagemath.org/MallocReplacements#CA-8edc33073c91e5c602526a95ad5eeadded8e97f4_3) # PyMem\_Malloc  
 [4](https://wiki.sagemath.org/MallocReplacements#CA-8edc33073c91e5c602526a95ad5eeadded8e97f4_4) sage: x = 5; y = 3  
 [5](https://wiki.sagemath.org/MallocReplacements#CA-8edc33073c91e5c602526a95ad5eeadded8e97f4_5) sage: %timeit \_ = x+y  
 [6](https://wiki.sagemath.org/MallocReplacements#CA-8edc33073c91e5c602526a95ad5eeadded8e97f4_6) 100000 loops, best of 3: 8.15 microseconds per loop  
 [7](https://wiki.sagemath.org/MallocReplacements#CA-8edc33073c91e5c602526a95ad5eeadded8e97f4_7)   
 [8](https://wiki.sagemath.org/MallocReplacements#CA-8edc33073c91e5c602526a95ad5eeadded8e97f4_8) # default libCF memory manager  
 [9](https://wiki.sagemath.org/MallocReplacements#CA-8edc33073c91e5c602526a95ad5eeadded8e97f4_9) sage: memman\_malloc()  
 [10](https://wiki.sagemath.org/MallocReplacements#CA-8edc33073c91e5c602526a95ad5eeadded8e97f4_10) sage: %timeit \_ = x+y  
 [11](https://wiki.sagemath.org/MallocReplacements#CA-8edc33073c91e5c602526a95ad5eeadded8e97f4_11) 100000 loops, best of 3: 7.36 microseconds per loop

## **libCF new Memory Manager**

* GPL
* crashed on me ([malb](https://wiki.sagemath.org/MartinAlbrecht)) when trying to use it from withing SAGE

## **GivMM**

* GPL
* "Short description of the Givaro memory mechanism: each request of sz bytes returns the smallest bloc that contains at least sz bytes. Each bloc begins by a private data used to store size information or a pointer. All free blocs are linked in list. The allocation algorithm search in the corresponding list a free bloc, before call the system "malloc" routine. 512 different sizes of bloc are predefined. The maximum bloc size is 8054880 bytes. For higher bloc, the manager directly call malloc/free of the underlaying system."
* Seems to be slower than [PyMem](https://wiki.sagemath.org/PyMem)\_Malloc for Integer addition

[Toggle line numbers](https://wiki.sagemath.org/MallocReplacements#)

[1](https://wiki.sagemath.org/MallocReplacements#CA-8c0b700ab7d83a0473e0696df9b9aaf796da7c09_1) sage: from sage.memory import \*  
 [2](https://wiki.sagemath.org/MallocReplacements#CA-8c0b700ab7d83a0473e0696df9b9aaf796da7c09_2) sage: x = 5; y = 3  
 [3](https://wiki.sagemath.org/MallocReplacements#CA-8c0b700ab7d83a0473e0696df9b9aaf796da7c09_3)   
 [4](https://wiki.sagemath.org/MallocReplacements#CA-8c0b700ab7d83a0473e0696df9b9aaf796da7c09_4) # PyMem\_Malloc  
 [5](https://wiki.sagemath.org/MallocReplacements#CA-8c0b700ab7d83a0473e0696df9b9aaf796da7c09_5) sage: %timeit \_ = x+y  
 [6](https://wiki.sagemath.org/MallocReplacements#CA-8c0b700ab7d83a0473e0696df9b9aaf796da7c09_6) 100000 loops, best of 3: 8.15 microseconds per loop  
 [7](https://wiki.sagemath.org/MallocReplacements#CA-8c0b700ab7d83a0473e0696df9b9aaf796da7c09_7)   
 [8](https://wiki.sagemath.org/MallocReplacements#CA-8c0b700ab7d83a0473e0696df9b9aaf796da7c09_8) # GivaroMM  
 [9](https://wiki.sagemath.org/MallocReplacements#CA-8c0b700ab7d83a0473e0696df9b9aaf796da7c09_9) sage: givaro\_malloc()  
 [10](https://wiki.sagemath.org/MallocReplacements#CA-8c0b700ab7d83a0473e0696df9b9aaf796da7c09_10) sage: %timeit \_ = x+y  
 [11](https://wiki.sagemath.org/MallocReplacements#CA-8c0b700ab7d83a0473e0696df9b9aaf796da7c09_11) 100000 loops, best of 3: 8.36 microseconds per loop  
**MuPAD Pexpect Interface**

sage: mupad.package('"MuPAD-Combinat"')  
sage: combinat = mupad.combinat  
sage: examples = mupad.examples  
sage: S = examples.SymmetricFunctions()  
sage: S.s[2,1]^2  
  
s[3, 3] + s[4, 2] + s[2, 2, 1, 1] + s[2, 2, 2] + 2 s[3, 2, 1] + s[4, 1, 1] +  
  
 s[3, 1, 1, 1]  
sage: S.omega( S.s[3] )  
 s[1, 1, 1]  
sage: s = S.s  
sage: p = S.p  
sage: s[2,1] + p[2,1]  
 - e[2, 1] - e[3] + e[1, 1, 1]  
sage: s(\_)  
 s[2, 1] + s[3] - s[1, 1, 1]  
  
sage: combinat.tableaux.list(3)  
  
 -- +---+ --  
 | | 3 | |  
 | +---+ +---+ +---+ |  
 | | 3 | | 2 | | 2 | |  
 | +---+---+---+ +---+---+ +---+---+ +---+ |  
 | | 1 | 2 | 3 |, | 1 | 2 |, | 1 | 3 |, | 1 | |  
 -- +---+---+---+ +---+---+ +---+---+ +---+ --  
sage: t = \_[1]  
sage: t  
  
 +---+---+---+  
 | 1 | 2 | 3 |  
 +---+---+---+  
sage: combinat.tableaux.conjugate(t)  
  
 +---+  
 | 3 |  
 +---+  
 | 2 |  
 +---+  
 | 1 |  
 +---+  
sage: combinat.ribbonsTableaux.list([2,2],[1,1],2)  
  
 -- +---+---+ +---+---+ --  
 | | | 2 | | 2 | |  
 | + + +, +---+---+ |  
 | | 1 | | | 1 | |  
 -- +---+---+ +---+---+ --  
sage: combinat.tableaux.kAtom([3,2,1],3)  
  
-- +---+ +---+ --  
| | 3 | | 3 | |  
| +---+---+ +---+ +---+---+ +---+ |  
| | 2 | 2 | | 2 | | 2 | 2 | | 2 | |  
| +---+---+---+ +---+---+---+---+ +---+---+---+---+ +---+---+---+---+---+ |  
| | 1 | 1 | 1 |, | 1 | 1 | 1 | 2 |, | 1 | 1 | 1 | 3 |, | 1 | 1 | 1 | 2 | 3 | |  
-- +---+---+---+ +---+---+---+---+ +---+---+---+---+ +---+---+---+---+---+ --  
  
sage: M = S.Macdonald()  
sage: M.P[1]^2  
  
 (q t - 1) (t - 1) / 1  
- ------------------ P[2] - (t - 1) | ----------------- +  
 2 | 3 2  
 t - q t + q t - 1 \ - t + t + t - 1  
  
 q - t \ 2  
 ----------------------------- | (t - 1) P[1, 1]  
 3 2 |  
 (q t - 1) (- t + t + t - 1) /  
sage: mupad.mapcoeffs(\_, 'normal')  
  
 q - t + q t - 1  
 P[2] + --------------- P[1, 1]  
 q t - 1

# NotebookWiki

This page is to be intended as a Request For Comments. If there is something you would like to see in the proposed system, your input will be appreciated. Also, [[TomBoothby|the author]] acknowledges that NotebookWiki is a poor name for what may be a revolutionary fusion of two already great ideas. Other name ideas: WDE (Wiki Development Environment), Nowiki...

'''Motivation'''

The motivation behind the NotebookWiki is to bring together the strengths of both the SAGE Notebook (try it at http://sage.math.washington.edu:8100), and the concept of a wiki. A wiki harnesses the talents of many, enabling large-scale group authoring. However, most wikis are centered around textual content. The SAGE Notebook is an attempt to consolidate good features of other development environments to provide an intuitive interface for a SAGE user. However, its multi-user support is currently flawed in a number of aspects. By bringing together these ideas, we can create a strong multi-user environment that may one day be used by many developers to manage large-scale projects like SAGE itself.

'''Wiki Pages, Worksheets, and Filesystems'''

In the NotebookWiki, the concepts of "page" in a wiki and the "worksheet" in a notebook would be consolidated. To create a development environment, each page would represent a file or directory in a filesystem. Documentation and discussion about changes to each file of a project would be in a single location. At any given time, a user should be able to query the wiki for a tarball of the project, as well as incremental patches.

'''The Cell Paradigm'''

Typically, each section of a wiki has a bolded title, and a body. These are all input as a single body of text. The SAGE notebook, however, has a collection of cells which are input and managed as discrete units. Currently, the only operations allowed on the notebook cells are: Create, Delete, Evaluate, Introspect. This is good, but for a wiki editor, these would not be sufficient -- one must also be able to Join, Split, and Move cells. Embedding of cells may also be considered in the future.

'''Conflict Resolution'''

With every edit, the NotebookWiki should check for conflicts when a cell is edited. Through asynchronous http requests, the browser should be able to notify users when they are making conflicting changes. Conflict resolution could be handled through a chat system, where all users attempting to make changes are able to discuss the differences and come to agreement together. This is a significant improvement over the typical version control system, because rather than a single maintainer who may not be familiar with other changes to the files, the contributing developers will be more prepared to resolve the conflicts together.

'''Roadmap'''

This is very broad, and subject to complete revision.

\* Implement XMLRPC in Notebook.

\* Remove all HTML formatting from cell.py, notebook.py, and worksheet.py

\* The role of the server would be to communicate XMLRPC's between the client (pure javascript) and the notebook

\* All formatting should be done in javascript, communication through AJAX / XMLRPC

\* Remove main framework until the point that the javascript can be included as a single library

\* Incorporate library into wiki

# Open Source CAS

This page will survey open source computer algebra systems (OSCAS's). Some of this may appear as columns in [CCA](http://www.acm.org/sigsam/bulletin/).

A *computer algebra system* (CAS) is a mathematical software package capable of symbolic manipulation. The commercial CAS industry is big business. Few people know more about the CAS industry than Darren McIntyre, VP of Worldwide Sales at Maplesoft. He estimates the worldwide yearly expenditures on computer algebra (buying licenses, employee salaries, and so on) is at least $ 600 million. Clients include not just students and universities, but diverse industries who often find that a CAS is a convenient programming environment to model industrial problems.

For a list maintained by sigsam, see [here](http://www.sigsam.org/Resources/Software.html).

For the list by Oberwolfach, see [here](https://orms.mfo.de/).

Another list by the AMS [here](http://www.mathontheweb.org/mathweb/mi-software.html).

## **The terrain**

|  |  |  |
| --- | --- | --- |
| Axiom | [BSD](http://www.opensource.org/licenses/bsd-license.php) | [http://axiom-developer.org](http://axiom-developer.org/) |
| CADABRA | GPL | <http://cadabra.phi-sci.com/> |
| [DoCon](https://wiki.sagemath.org/DoCon) | open source | <http://homepages.inf.ed.ac.uk/wadler/realworld/docon2.html> |
| FriCAS | [BSD](http://www.opensource.org/licenses/bsd-license.php) | [http://fricas.sourceforge.net](http://fricas.sourceforge.net/) |
| GAP | GPL | [http://www.gap-system.org](http://www.gap-system.org/) |
| GIAC | GPL | <http://www-fourier.ujf-grenoble.fr/~parisse/giac.html> |
| GINAC | GPL | [http://www.ginac.de](http://www.ginac.de/) |
| GTYBALT | GPL | <http://wwwthep.physik.uni-mainz.de/~stefanw/gtybalt/> |
| JScience | BSD | <http://www.jscience.org/> |
| LiDIA | "open source" | <http://www.cdc.informatik.tu-darmstadt.de/TI/LiDIA/> (broken link) |
| Macaulay2 | GPL | <http://www.math.uiuc.edu/Macaulay2/> |
| Magnus | GPL | <http://sourceforge.net/projects/magnus/> |
| MAS | "open source" | <http://krum.rz.uni-mannheim.de/mas/> |
| Mathemagix | GPL? | <http://www.mathemagix.org/> |
| Mathomatic | LGPL | <http://launchpad.net/mathomatic> |
| Maxima | GPL | [http://maxima.sourceforge.net](http://maxima.sourceforge.net/) |
| NTL | GPL | <http://www.shoup.net/ntl/> |
| Pari | GPL | [http://pari.math.u-bordeaux.fr](http://pari.math.u-bordeaux.fr/) |
| [SageMath](https://wiki.sagemath.org/SageMath) | GPL | <http://www.sagemath.org/> |
| Scilab | "open source" | [http://www.scilab.org](http://www.scilab.org/) |
| Singular | GPL | [http://www.singular.uni-kl.de](http://www.singular.uni-kl.de/) |
| Symmetrica | public domain | <http://www.mathe2.uni-bayreuth.de/axel/symneu_engl.html> |
| SymPy | BSD | <http://sympy.org/> |
| Yacas | GPL | [http://yacas.sourceforge.net](http://yacas.sourceforge.net/) |

Omitted is *CAFE* (Computer Algebra and Functional Equations), a group writing a collection of CAS libraries (see [CAFE](http://www-sop.inria.fr/cafe/main-e.html)). They appear to be written in Aldor and Maple by (the late) Manuel Bronstein. I (=David Joyner) cannot determine the license (if any) they are released under. I am also unsure if the "open source" licenses of LiDIA, MAS, and Scilab are compatible with the above-mentioned open source definition. Several of these are under very active development and some of these are essentially dead. Two other sources of information are the Computer algebra handbook [GKW](https://wiki.sagemath.org/%5B#references)] and the internet sites [CA](https://wiki.sagemath.org/%5B#references)].

In any case, we see from this table that there are a lot of open source computer algebra systems out there. Some of these are special purpose (such as Symmetrica) and others are general purpose (such as Axiom). We shall start by surveying Maxima, a general purpose CAS.

* [Maxima\_as\_an\_OSCAS](https://wiki.sagemath.org/Maxima_as_an_OSCAS)
* [Axiom\_as\_an\_OSCAS](https://wiki.sagemath.org/Axiom_as_an_OSCAS)