Review of Python-based symbolic mathematics systems and libraries

Mateusz Paprocki[1,2] [1] University of Nevada, Reno, [2] SymPy Development Team

Abstract

We compare five Python-based symbolic mathematics systems and libraries: **Sage**, **Swiginac**, **SymPy**, **SymPyCore** and **Pynac** (in order of first appearance). We would like to make those programs more familiar to the reader and help him choose the right system for his problem domain.

Why Python?

Python is a modern, general purpose programming language that is easy to learn but has also very expressive syntax and semantics. It allows for interoperability with foreign libraries, thanks to its C API and supporting tools (e.g. Cython, f2py). Almost a complete scientific stack was created around Python, with NumPy and SciPy leading the way. One missing piece of this ecosystem was a system/library for symbolic mathematics. This gap was filled with programs that were either developed from scratch entirely in Python or use it as a command language (instead inventing yet another mathematical DSL).

Facts

	Sage	Swiginac	SymPy	SymPyCore	Pynac
Website	www.sagemath.org	swiginac.berlios.de	www.sympy.org	code.google.com/p/sympycore	pynac.sagemath.org
Initial release	24 February 2005	25 September 2005	11 March 2007	29 February 2008	8 August 2008
Stable release	4.7 (23 May 2011)	1.5.1 (15 April 2009)	0.7.1 (29 July 2011)	0.1 (29 February 2008)	0.2.2 (9 May 2011)
Download size	411 MB (Ubuntu 64-bit, lzma)	100 KB (gzip)	3.4 MB (gzip)	138 KB (gzip)	2.1 MB (bzip2)
Installed size	2.4 GB	very little	very little	very little	very little
License	GNU GPL	GNU GPL	New BSD	New BSD	GNU GPL
Author(s)	William Stein	O. Skavhaug & O. Certik	Ondrej Certik	Pearu Peterson	Burcin Erocal
Contributors	> 150	7	> 120	3	7
SCM software	Mercurial (HG)	SVN	GIT	SVN	Mercurial (HG)
Languages	Cython, Python	C++, Python	Python	Python, C	C++
Philosophy	multiple wrappers	wrappers (Ginac)	developed from scratch	derivative of SymPy	wrappers (Ginac)
Web interface	www.sagenb.org	none	live.sympy.org	none	www.sagenb.org

Descriptions

Sage is a free open-source mathematics software system licensed under the GPL. It combines the power of many existing open-source packages into a common Python-based interface.
Pros: full scientific stack, very functional, fast, large community
Cons: very large, not a library, complicated design

Swiginac is a Python interface to GiNaC, built with SWIG. The aim of Swiginac is to make all the functionality of GiNaC accessible from Python as an extension module.
Pros: small library, fast
Cons: not actively developed, written in C++, no community

SymPy is a Python library for symbolic mathematics. It aims to become a full-featured computer algebra system (CAS) while keeping the code as simple as possible in order to be comprehensible and easily extensible. SymPy is written entirely in Python and does not require any external libraries.
Pros: small library, pure Python, very functional, extensible, large community
Cons: slow, needs better documentation

SymPyCore is inspired by many attempts to implement CAS for Python and it is created to fix SymPy performance and robustness issues. SymPyCore does not yet have nearly as many features as SymPy. Our goal is to work on in direction of merging the efforts with the SymPy project in the future.
Pros: small library, fast, pure Python (with optional C extensions)
Cons: only basic functionality, weak documentation, no community

Pynac is a derivative of the C++ library GiNaC, which allows manipulation of symbolic expressions. It currently provides the backend for symbolic expressions in Sage. The main difference between Pynac and GiNaC is that Pynac relies on Sage to provide the operations on numerical types, while GiNaC depends on CLN for this purpose.
Pros: small library, fast
Cons: only basic functionality, not a standalone library (depends on Sage)

Sage vs. SymPy

Sage and SymPy may look very similar, but those are two very different systems with completely different internal design, non-overlapping features sets (e.g. Sage is very good at number theory and abstract algebra, but SymPy has sophisticated pretty printing and code generation frameworks) and quite different semantics:

```
sage: var('x')
x
sage: sin(10).n(digits=10)
-0.5440211109
sage: integrate(sin(x), x, 0, 2*pi)
0
sage: sin(x) == 1
sin(x) == 1
sage: solve(sin(x) == 1, x)
[x == 1/2*pi]
sage: K.<t> = GF(2)['t']
sage: factor(t**2 + 1)
(t + 1)^2
```

```
>>> var('x')
x
>>> sin(10).evalf(n=10)
-0.5440211109
>>> integrate(sin(x), (x, 0, 2*pi))
0
>>> sin(x) == 1
False
>>> solve(Eq(sin(x), 1), x)
[pi/2]
>>> t = Symbol('t')
>>> factor(t**2 + 1, modulus=2)
(t + 1)**2
```

Examples

```
sage: var('x')
x
sage: sin(2*x).diff(x)
2*cos(2*x)
```

```
>>> from swiginac import symbol, sin
>>> x = symbol('x')
>>> sin(2*x).diff(x)
2*cos(2*x)
```

```
>>> from sympy import var, sin
>>> var('x')
x
>>> sin(2*x).diff(x)
2*cos(2*x)
```

```
>>> from sympycore import var, sin
>>> var('x')
Calculus('x')
>>> sin(2*x).diff(x)
Calculus('2*Cos(2*x)')
```

SymPy vs. SymPyCore

SymPyCore is a fork of `sympy.core` module. The project was created to quickly prototype a new fast symbolic core for SymPy. SymPyCore, however, diverged from SymPy and it's unlikely it will be merged with it any time soon.

Pynac vs. Swiginac

Pynac and Swiginac both wrap GiNaC, a C++ library for symbolic mathematics. Pynac uses Python's C API for this and Swiginac uses SWIG library. Both don't wrap the entire library. Pynac requires Sage for coefficient arithmetics, so you can't use it as a standalone library, as opposed to Swiginac.

Conclusions

Sage is the most stable and feature complete Python-based mathematics system, so if you are looking for a ready to use solution that could replace Mathematica or Maple, then Sage may be the optimal choice. However, Sage is a heavy weight system with complex internals, so if a light weight, easily extensible and embeddable library is needed, then SymPy may be a better choice. The other three programs miss too many important features to be currently considered useful for the end user.