Sage Development Days

Some baseless opinions about Maxima

Robert Dodier

What is Maxima?

Maxima is a symbolic computation system

. . . weakly-typed

. . . with a lot of built-in knowledge

. . . written in Lisp

Maxima has grown by accretion, not design

Maxima is a house of many chambers

Symbolic, weakly-typed, lot of built-in knowledge

Laissez-faire attitude — "figure it out later"

Built-in assumptions more or less equivalent to calculus, real analysis, complex analysis

Not too hard to invent new assumptions; more or less impossible to retract existing ones

A laissez-faire attitude

For the most part, Maxima is happy to leave alone an expression which it doesn't recognize

What is $1 + (\text{if } x > 0 \text{ then } y \text{ else } z)$?

Maxima doesn't know, and doesn't complain about it
$\Rightarrow$ Door is open to inventing a definition

"Attempting to prevent stupidity also prevents cleverness."

Cleverness = extrapolating from stated rules and known examples to novel situations

(Maxima does have a certain amount of cleverness prevention, e.g. $0^0 \Rightarrow$ error. This is a misfeature IMNSHO.)

Maxima development history

1968–1982 Project MAC (Machine-Aided Cognition) at MIT. Much of present-day code dates to that era

1982–2001 Fateman coerced MIT into turning over a snapshot of Macsyma to US Dept of Energy. Commercial version was developed in the 1980's and 1990's, while the DOE snapshot ("DOE Macsyma") was distributed to universities. One copy was maintained by Bill Schelter, who asked for and received permission from DOE to release his copy under terms of GPL (1998)

2001–present Death of Bill Schelter (2001), project moved to Sourceforge, new maintainers for Maxima (Jim Amundson) and GCL (Camm Maguire), many new participants, lots of new code, increase of downloads

Recent developments 2001–present

Improved plotting functions

Improved documentation

Several user interface projects

Imported math libraries (LAPACK, QUADPACK)

Mathematical functions

Many add-on packages of varying complexity (ODE's, tensors, etc etc)

Many bug fixes

Ports to various Common Lisp implementations

Gaps

"Assume" system — weak, bugs

Limits, definite integrals — bugs

Vectors — multiple inconsistent add-on packages

Solution of equations — polynomials OK but not much else

Arrays — multiple types, not consistent w/ lists and matrices

Boolean expressions — weak

I'd like to see any/all of these gaps filled in

Large-scale development wish list items

More powerful user language

Make Maxima easier to use as a computational engine

Worksheet-like user interface

Link or absorb specialized math libraries

Regularize simplification system

Maxima user language

Difficult to mix general programming with math — user language is too weak for general programming

Present user language is a simplified Algol-like language

Dynamic scope, not lexical

No built-in namespace system (there is a prototype)

Could get an improved language by (1) modifying existing language, or (2) replacing with e.g. Python

CLPython is Python written in CL; port to SBCL in progress

Present user language inherits useful behavior from Lisp; seems likely getting same behavior in Python would require modifying some basic properties (e.g. evaluation policy)

Maxima as a computational engine

e.g. behind a GUI or web interface or whatever

Ideally, one input $\Rightarrow$ one output

Major obstacle: Maxima assumes someone is at the console so (1) OK to ask questions and (2) print out-of-band stuff

I have made some progress about turning off the questions

About form of input and output, could be plain text, Lisp, or MathML expressions

If there were a foreign function interface, could be direct function call

Worksheet-like user interface

Combine text, computations, graphics in one document (e.g. Mathcad)

Several user interfaces which more or less work (TeXmacs, XMaxima, etc)

Probably best approach at present is to work on an existing project

Worksheet interface would benefit from improvements to Maxima as a computational engine

Link or absorb specialized math libraries

Make use of large body of existing work

Maxima has already imported LAPACK, QUADPACK, LBFGS, probably others

Fortran can be translated to CL via f2cl

No single foreign function interface for all Lisps
$\Rightarrow$ can't yet simply link to C or Fortran libraries

(Linking to C/Fortran is a mixed blessing — potential for segfaults)

Regularize simplification system

Simplification is "where the math is" in Maxima

Put built-in and user-constructed rules on same footing

Make it easy to extend or retract rules

Base system would come with built-in rules suitable for working in real or complex domains (as at present)

Easily specialized or otherwise modified by introducing new rules and object and/or retracting old ones

Maxima as the C language of symbolic computing

Large-scale goals and general direction

I want Maxima to be a comfortable environment for practical and impractical mathematics

Large-scale goals just mentioned support the "comfortable environment"

New chunks of mathematics within this environment are always welcome

Maxima project disorganization

I like it this way so I'm not trying to change it, and I'll resist such efforts by others

No legal entity, only individuals

No dictator (benevolent or otherwise), things get done when somebody feels like it

No project vision statement or explicit goals

No explicit task assignments or roles

Several project administrators; administrators do not set policy

Maxima reminds me of a workshop with workers pursuing different tasks

Maxima and Sage

License review

Working around “asksign” nuttiness

Reduce function call overhead: (1) link Maxima into same image as Sage via FFI or CLPython ??

or (2) invent a binary or near-binary representation of S-expressions?