GiNaC
Symbolic computation with C++

Jens Völlinga

Institut für Physik
Universität Mainz
Outline

CAS in high energy physics
Fundamentals of GiNaC
Features of GiNaC
Applications
Perturbation theory

- Feynman rules
- Feynman diagrams
- Dirac algebra and traces
- Loop integration
- Summation of diagrams
- Phase space integration
- Checks
- ...
CAS in high energy physics

Perturbation theory

- Feynman rules
- Feynman diagrams
- Dirac algebra and traces
- Loop integration
- Summation of diagrams
- Phase space integration
- Checks
- ...
CAS in high energy physics

Perturbation theory

- Feynman rules
- Feynman diagrams
- Dirac algebra and traces
- Loop integration
- Summation of diagrams
- Phase space integration
- Checks
- ...

Combinatorics
- (Maple, C, Perl)

Algebra
- (Maple, FORM)

⇒ Data management
- (C, BASH, Perl)

Numerics
- (FORTRAN, C)

I/O
- (C, Tcl/Tk)
Taming COMPLEXITY ➔ Software Engineering
(coding, social, technology, ...)

CAS in physics
CAS in physics

Taming COMPLEXITY → Software Engineering
(coding, social, technology, ...)

What if you are a C++ programmer?

- Combinatorics
- Algebra
- Data management
- Numerics
- I/O
CAS in physics

Taming COMPLEXITY → Software Engineering
(coding, social, technology, . . . )

What if you are a C++ programmer?

- Combinatorics
- Algebra
- Data management
- Numerics
- I/O
CAS in physics

Taming COMPLEXITY → Software Engineering
(coding, social, technology,...)

What if you are a C++ programmer?

- Combinatorics
- Algebra
- Data management
- Numerics
- I/O
What if you are a C++ programmer?

- Combinatorics
- Algebra
- Data management
- Numerics
- I/O
CAS in physics

Taming COMPLEXITY → Software Engineering
(coder, social, technology,...)

What if you are a C++ programmer?

- Combinatorics
- Algebra → GiNaC
- Data management
- Numerics
- I/O
Write a C++ program:

```cpp
#include <iostream>
using namespace std;

#include <ginac/ginac.h>
using namespace GiNaC;

int main()
{
    symbol x("x");
    ex result = Li(2,x).diff(x);
    cout << result << endl;
    return 0;
}
```
Write a C++ program:

```cpp
#include <iostream>
using namespace std;
#include <ginac/ginac.h>
using namespace GiNaC;

int main()
{
    symbol x("x");
    ex result = Li(2,x).diff(x);
    cout << result << endl;
    return 0;
}
```
Write a C++ program:

```cpp
#include <iostream>
using namespace std;
#include <ginac/ginac.h>
using namespace GiNaC;

int main()
{
    symbol x("x");
    ex result = Li(2,x).diff(x);
    cout << result << endl;
    return 0;
}
```
Write a C++ program:

```cpp
#include <iostream>
using namespace std;
#include <ginac/ginac.h>
using namespace GiNaC;

int main()
{
    symbol x("x");
    ex result = Li(2, x).diff(x);
    cout << result << endl;
    return 0;
}
```
Compile the program:

$ g++ -o myprg myprg.cpp -lginac

Run it:

$ ./myprg
-log(1-x)*x^(-1)
Container for arbitrary algebraic expressions

\[ \rightarrow \texttt{ex} \]
Container for arbitrary algebraic expressions

→ \texttt{ex}

After declaration

\texttt{ex myexpr;}

or as a function parameter for example

\texttt{ex nloopfct(ex momentum)}

\{ ... \}
Container for arbitrary algebraic expressions

→ \texttt{ex}

After declaration

\texttt{ex \ myexpr;}

or as a function parameter for example

\texttt{ex \ nloopfct(ex \ momentum)}

\{ \ldots \}

use it like C++ built-in types:

momentum = sqrt(p) * pow(mu, 2) - d;

myexpr = sin(x).series(x, 10);

cout << myexpr << endl;

if (myexpr.has(x)) y = myexpr;
Container for arbitrary algebraic expressions
→ \texttt{ex}

After declaration
\begin{verbatim}
ex myexpr;
\end{verbatim}
or as a function parameter for example
\begin{verbatim}
ex nloopfct(ex momentum)
{
...
}
\end{verbatim}

use it like C++ built-in types:
\begin{verbatim}
momentum = sqrt(p) * pow(mu, 2) - d;
myexpr = sin(x).series(x, 10);
cout << myexpr << endl;
if (myexpr.has(x)) y = myexpr;
\end{verbatim}
Symbols → symbol

Declaration:

symbol x("x");
symbol eps("\epsilon")
Symbols → symbol

Declaration:
symbol x("x");
symbol eps("\epsilon")

What else?
- Numbers 1.34, 3/4, 2i, ...
- Mathematical functions
- Matrices
- Algebras
  ...
Features of GiNaC

- Arbitrary symbolic expressions
Features of GiNaC

- Arbitrary symbolic expressions
- Complex arithmetics with arbitrary precision
Features of GiNaC

- Arbitrary symbolic expressions
- Complex arithmetics with arbitrary precision
- Operations on rational functions, e.g. \( \frac{x^2 - y^2}{(x+y)^2} \rightarrow \frac{x-y}{x+y} \)
Features of GiNaC

- Arbitrary symbolic expressions
- Complex arithmetics with arbitrary precision
- Operations on rational functions,
  e.g. \[ \frac{x^2 - y^2}{(x+y)^2} \rightarrow \frac{x-y}{x+y} \]
- Symbolic derivation and series expansion
Features of GiNaC

- Arbitrary symbolic expressions
- Complex arithmetics with arbitrary precision
- Operations on rational functions,
  e.g. \( \frac{x^2 - y^2}{(x+y)^2} \rightarrow \frac{x-y}{x+y} \)
- Symbolic derivation and series expansion
- Matrices, vectors, linear equation systems
Features of GiNaC

- Arbitrary symbolic expressions
- Complex arithmetics with arbitrary precision
- Operations on rational functions, e.g. \( \frac{x^2 - y^2}{(x+y)^2} \rightarrow \frac{x-y}{x+y} \)
- Symbolic derivation and series expansion
- Matrices, vectors, linear equation systems
- In-/Output of expressions in various formats (text, binary, LaTeX, . . . )
Features of GiNaC

- Arbitrary symbolic expressions
- Complex arithmetics with arbitrary precision
- Operations on rational functions,
  e.g. \( \frac{x^2 - y^2}{(x+y)^2} \to \frac{x-y}{x+y} \)
- Symbolic derivation and series expansion
- Matrices, vectors, linear equation systems
- In-/Output of expressions in various formats (text, binary, LaTeX, . . .)
- New functions and classes can easily be added
Features of GiNaC

- Functions

\[ \Gamma, B, \psi, \binom{n}{k}, \ldots \]
\[ \sin, \cos, \tan, \sinh, \ldots \]
\[ \sqrt{\cdot}, \exp, \log, \zeta, S, H, Li, \ldots \]
Features of GiNaC

- Functions
  \[ \Gamma, B, \psi, \binom{n}{k}, \ldots \]
  \[ \sin, \cos, \tan, \sinh, \ldots \]
  \[ \sqrt{}, \exp, \log, \zeta, S, H, Li, \ldots \]

- Objects with indices:
  \[ p^\mu, A_{ij} \]

Special algebras: Clifford, SU(3)
Features of GiNaC

- Functions
  \[ \Gamma, B, \psi, \binom{n}{k}, \ldots \]
  \[ \sin, \cos, \tan, \sinh, \ldots \]
  \[ \sqrt{}, \exp, \log, \zeta, S, H, Li, \ldots \]

- Objects with indices: \( p^\mu, A_{ij} \)

- Special algebras: Clifford, SU(3)

- Automatic code generation (\( \rightarrow \) numeric integration)
Features of GiNaC

- **Functions**
  \[ \Gamma, B, \psi, \binom{n}{k}, \ldots \]
  \[ \sin, \cos, \tan, \sinh, \ldots \]
  \[ \sqrt{}, \exp, \log, \zeta, S, H, Li, \ldots \]

- **Objects with indices:** \( p^\mu, A_{ij} \)
- **Special algebras:** Clifford, SU(3)
- **Automatic code generation** (→ numeric integration)
- **Good documentation, open source (GPL)** → [www.ginac.de](http://www.ginac.de)
Applications with GiNaC

- Loop calculations: xloops
- nestedsums
- gTybalt
- feelfem, PURRS, MBDyn

Publications

On the Invariance of Residues of Feynman Graphs

The Massless Two-Loop Two-Point Function

The Electroweak Standard Model in the Axial Gauge

An Example of Clifford Algebras Calculations with GiNaC
V. Kisil, [arXiv:cs.MS/0410044]
GiNaC – a C++ library for symbolic computation

- Complete computational framework in C++
- Features of GiNaC
GiNaC – a C++ library for symbolic computation

- Complete computational framework in C++
- Features of GiNaC
- Developed since 1999
  Version 1.0 in 2001, current version 1.3.1
  Technically matured and well documented
- Active development going on
  (better factorization, more special functions, improved interface to MC integration)
www.ginac.de