WHIZARD: Algorithms and Computing Issues

Wolfgang Kilian

University of Siegen, Germany

CHEP, Beijing, August 2014

Today's Subjects

1. Phase Space: Whizard's Standard Algorithm
2. Beam Structure
3. Some QCD Issues
4. Remarks on Computing and Package Structure

1. Phase Space

Diagrams and Phase Space

Process:

$$p_1, p_2 \to q_1, \ldots q_n \qquad \text{short:} \quad p \to q$$

(Tree) amplitude, expanded: sum of complex-valued functions:

$$A(p,q) = \sum_i A_i(p,q)$$

Each $A_i(p,q) =$ Feynman graph.

WHIZARD: amplitude not expanded, terms A_i not explicitly available

Phase-space integral:

$$\int dq \sum_{s,c} |A(p,q)|^2$$

where

$$dq = \delta^4(p-q) \prod_i \frac{d^3 q_i}{2E(q_i)}$$

Each term $A_i(p,q)$ consists of

- Numerator: well-behaved function of momentum invariants
  (squared+spin-summed: polynomial)
- Denominator: contains zeros near physical region:
  1. Massless particle (soft):
  $$A_i(p,q) \sim \frac{1}{E(q)}$$
  2. Massless particles (collinear splitting):
  $$A_i(p,q) \sim \frac{1}{q_i \cdot q_j}$$
  3. Massive resonance:
  $$A_i(p,q) \sim \frac{1}{\sum q_j^2 - m^2 + im\Gamma}$$
- Furthermore: cuts
  $\Rightarrow$ integrand is zero in finite part of phase space
- Integration: phase-space parameterization $q = q(x)$
  $\Rightarrow$ integrand is zero in finite part of x space
  $\Rightarrow$ integrand is singular at edges of physical region

Summary:

Integrand varies over many orders of magnitude,
contains (integrable) singularities, discontinuities, zero regions

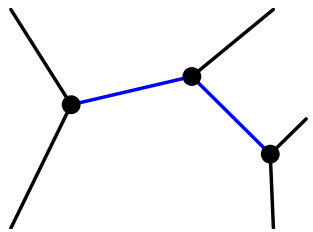
Interference terms $A_i^*(p,q)\,A_j(p,q)$ contain peaks from two diagrams simultaneously.

$\Rightarrow$ Direct Monte-Carlo rejection algorithm: extremely low efficiency

Need phase-space mappings that improve this efficiency

Phase-Space Parameterization

Simplest case: s-channel diagram



Phase space can be factorized into integration over $1 \to 2$ decay angles

$$dq_i\, dq_j \sim d\cos\theta_{ij}\, d\phi_{ij}$$

and integration over invariant masses

$$dm_{ij}^2 = d(q_i + q_j)^2$$

Invariant masses: also determine denominator zeros (resonance, soft)

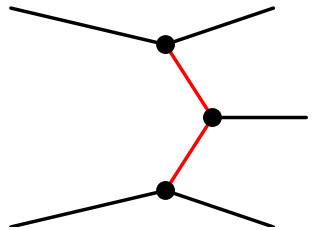
$\Rightarrow$ find mapping $m_{ij}^2 = m_{ij}^2(x)$ where Jacobian dm_{ij}^2/dx cancels denominator

- Soft: $m_{ij}^2 \sim e^x$ [with cut]
- Resonance: $m_{ij}^2 \sim \tan\left(c(x-1/2)\right)$

similar: collinear singularity vs. $\cos\theta_{ij}$ integration

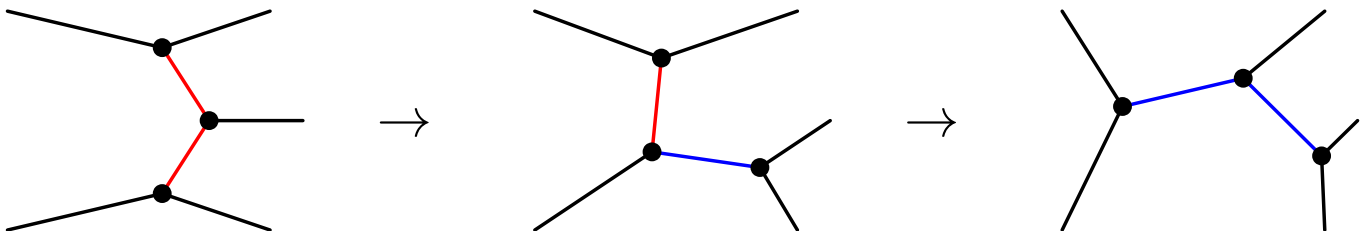
$\Rightarrow$ Integrand as function of x varies only moderately

More difficult: t-channel / multiperipheral diagram



singularity depends on $\cos\theta_{ij}$ (including initial momentum p_i)

WHIZARD approach: transform into s-channel diagram(s):



Apply mappings for $\cos\theta_{ij}$ as before.

WHIZARD: 'Wood' Phase Space

Algorithm:

1. Determine dominant pole structure: construct pseudo-graphs with largest number of singularities (including some subdominant terms)

   This is independent of the amplitude calculation, separate part of the program

   Phase-Space pseudo-graphs can be displayed. Don't confuse this with the Feynman graphs that are (implicitly) contained in the amplitude!
2. Transform t-channel pseudo-graphs into s-channel pseudo-graphs (**trees**).
3. Each tree has a set of mappings attached to it.
4. Distinct trees related by symmetries constitute a **grove**. The complete set of distinct groves is called the **forest**.

Phase-space evaluation: choose x values and evaluate one of the trees (phase-space channels) = compute $q(x)$ and the Jacobian.

Then invert the calculation to obtain the x and Jacobians for all other trees.

This would work for amplitudes that consist of a single Feynman graph.

So far, it does not work for amplitudes that contain multiple graphs, with different singularity structure.

Different amplitudes interfere.

For each phase space point q, we have the complete set of parameterizations $x^{(i)}$ and all corresponding Jacobians $|dq/dx^{(i)}|$.

Multi-Channel Phase Space

Multi-Channel phase-space integration was introduced by Kleiss and Pittau, applied to the EXCALIBUR Monte Carlo.

The Basic Idea:

- Choose a set of real-valued functions

$$g_i(q) \qquad \text{with} \quad g_i(q) \geq 0 \qquad \text{and} \quad \int dq\, g_i(q) = 1$$

  Each function represents a probability distribution.

  We can choose one function for each phase-space channel.

- Choose a set of real weight factors

$$w_i \qquad \text{with} \quad w_i \geq 0 \qquad \text{and} \quad \sum_i w_i = 1$$

  Each weight corresponds to a phase-space channel.

- The sum of all weighted probabilities

$$g(q) = \sum_i w_i g_i(q)$$

is again a real function with unit integral. It must be positive in the whole physical region:

$$g(q) > 0 \qquad \text{for all allowed } q \text{ configurations}$$

The phase-space integral is then broken down into channels

$$\int dq\,|A(q)|^2 = \int dq\,\frac{\sum_i w_i g_i(q)}{\sum_j w_j g_j(q)}\,|A(q)|^2$$
$$= \sum_i w_i \int dq\,\frac{g_i(q)}{g(q)}\,|A(q)|^2$$

All integrals contain the complete amplitude $A(q)$.

This helps the calculation if each g_i has the peak structure of its corresponding channel, i.e., $A_i(q)$ (squared).

- $\Rightarrow$ Where $A_i(q)$ is large, the factor is of order unity
- $\Rightarrow$ Where $A_i(q)$ is small but some other $A_j(q)$ is large, the enhancement occurs in $g(q)$ but not $g_i(q)$. This cancels in this particular term.

Therefore, **MadEvent** chooses $g_i(q) = |A_i(q)|^2/|A(q)|^2$.

In **WHIZARD**, we don't have individual $|A_i(q)|^2$.

(Interferences!)

Multichannel in WHIZARD

WHIZARD knows, for each channel, the parameterizations $q(x^{(i)})$, the inverse functions $x^{(i)}(q)$, and the corresponding Jacobians $|dq/dx^{(i)}|$.

For each channel, implement the corresponding parameterization.

$$\sum_i w_i \int dq\, \frac{g_i(q)}{g(q)}\, |A(q)|^2$$

$$= \sum_i w_i \int dx^{(i)}\, \frac{g_i(q(x^{(i)}))\, |\frac{dq}{dx^{(i)}}|}{g(q(x^{(i)}))}\, |A(q(x^{(i)}))|^2$$

Now choose

$$g_i(x^{(i)}) \sim \frac{1}{|dq/dx^{(i)}|}$$

Mappings

- $\Rightarrow$ Jacobians reflect pole structure
- $\Rightarrow$ Jacobians enhance denominator for $j \neq i$ and cancel extra peaks in $|A|^2$

Algorithm: (Evaluation of Wood Phase Space)

1. Choose channel $\#i$ with probability w_i
2. Choose $x^{(i)}$ randomly with uniform probability
3. Evaluate q, Jacobians and inverse mappings to get $x^{(j)}$, $j \neq i$
4. Evaluate complete amplitude for this point q
5. Sum all contributions from all channels

Adaptive Phase Space: Weights

Adaptive Improvement: Start with uniform weights $w_i = 1/N$

Compute integral and variance, broken down into channels.

When done, enhance w_i where variance is large, suppress w_i where variance is small.

Limit: number of points per channel must not be too small.

Repeat with new set of weights.

[So far: standard procedure]

However: Numerators and interferences still degrade the performance.

Example: $50\,\%$ efficiency per dimension.

$$20 \text{ dimensions} \quad \Rightarrow \quad \text{efficiency} < \frac{1}{2^{20}} \approx 10^{-6}$$

Improve this?

VEGAS Algorithm

Assume that the integrand can be approximately factorized

$$f(x) = f_1(x_1)\, f_2(x_2) \cdots f_n(x_n)$$

There exist mappings, one for each dimension

$$x_i = x_i(y_i) \qquad \text{with} \quad \frac{dx_i}{dy_i} \sim \frac{1}{f_i(x_i)}$$

But the mappings are not known analytically.

Choose a binning of the interval $y_i = (0, 1)$ and construct $x_i(y_i)$ as a piecewise continuous sequence of straight lines.

$\Rightarrow$ Bin width determines slope $=$ Jacobian for the current bin

$\Rightarrow$ For a large number of bins, dx_i/dy_i follows the optimal shape

Adaptive Phase Space: Grid

Adaptive Improvement: Start with uniformly spaced bins for each integration dimension, i.e., uniform grid.

Compute integral and variance, broken down into bins, summed over all dimensions except one.

When done, narrow bin where variance is large, widen bin where variance is small.

Limit: number of points per bin must not be too small.

Repeat with new grids.

[Also standard procedure]

VAMP: Vegas AMPlified

The VAMP module for integration uses the VEGAS algorithm in order to improve the mapping, for each channel separately.

Probability functions:

$$g_i(y^{(i)}) = \frac{1}{\left|\frac{dq}{dx^{(i)}}\right|} \times \prod_k \frac{1}{\left|\frac{dx_k^{(i)}}{dy_k^{(i)}}\right|}$$

First factor: analytically known and numerically evaluated.

Second factor: given by bin widths of VEGAS grids

The VAMP module contains the procedures needed for sampling (integration), adaptation, and rejection (simulation).

The two methods are applied independently.

- The Wood Phase Space may be exchanged by another (multichannel) phase space parameterization.
- The VAMP integration/simulation method may be exchanged by another (multichannel) integration method.

We implement this by making phase-space parameterization and multi-channel integration abstract data types (a.k.a. virtual classes).

The calling program doesn't know about details. It just knows that there is a phase space object with associated methods, and there is an integration object with associated methods.

The algorithms can be exchanged with any other algorithm that implements the abstract data type and its type-bound procedures.

The algorithms can be chosen separately for each process, within the same WHIZARD run.

By choosing the combination of both methods, we get

$\Rightarrow$ dominant contributions with their singularities are regularized

$\Rightarrow$ the residual variation is tamed by VAMP grid adaption. This can also lessen the impact of subdominant peaks.

$\Rightarrow$ Adapting both grids and weights simultaneously, we can achieve both an accurate Monte-Carlo integration

$$\text{statistical error} = \frac{\Delta\sigma}{\sigma} \approx \frac{0.1\ldots10}{\sqrt{N}} = \frac{\text{accuracy}}{\sqrt{N}}$$

and a reasonable reweighting efficiency

$$\epsilon = \frac{\text{average integrand}}{\text{maximum integrand}} \approx 0.1\ldots10\,\%$$

The algorithm forces us to select only dominant channels (pole structures):

- The number of free parameters is large. Example: 1000 channels, 15 dimensions, 20 bins

$$1000 \times 15 \times 20 = 300,000 \quad \text{free parameters}$$

  All parameters are adapted simultaneously.
- An adaptation run should not evaluate less points than there are free parameters.
- Other programs: no grids, but keep all graphs as independent channels.

  $\Rightarrow$ We select channels in advance in order to limit the impact of statistical fluctuation.

Where this works rather well:

- Scattering processes with a pronounced resonance structure: electroweak processes, top-pair production, SUSY etc.
- Signal and background preferably simulated together, no need for separate runs

Possible caveats:

- Subdominant processes may play a role: strong peaks, but low contribution to the result.
  $\Rightarrow$ The channel selection should try to keep these contributions.
- There is no guarantee that the algorithm converges (it does suprisingly well)
- The program can only estimate the actual maximum of the integrand.
- In some cases, one needs a large number of calls per iteration. In some cases, one needs a large number of iterations.
  $\Rightarrow$ Always check the intermediate results for convergence. Watch out for apparent instabilities.

What You Observe When Running WHIZARD

Default setup:

1. The program constructs amplitude expression for each process:
   - calls OMega as external program (or GoSam, etc.)
   - compiles and dynamically links the amplitude code
2. For each integration run, it constructs the phase space. By default, using the Wood phase space module:
   - Call the cascades module which writes a configuration file for the phase space.
   - Read the configuration file and set up the phase space data for integration

   [In a second run, WHIZARD just re-reads this phase-space file.]

3. Adaptation pass: sample phase space with a given number of calls, repeating for a given number of iterations.

   For each iteration, WHIZARD prints the current integral and error and efficiency estimates.

   Between iterations, grids and weights are adapted. Quoted accuracy and efficiency should improve.

   At the end of the adaptation pass, the accumulated average is quoted with error (but should not be taken too seriously).

4. Integration pass: sample again, no more adaption.

   The final average and error (with χ^2 and efficiency) can be quoted.

5. Simulation: importance sampling and rejection using previous grids and weights

   $\Rightarrow$ produce event files, optionally analyze results.

Performance?

- The Wood/VAMP combination of algorithms performs surprisingly well, much better than originally expected.
- Also works for QCD amplitudes (no dominant resonances)
- Phase space and VAMP use up a significant fraction of CPU time and memory.

  For the interesting cases with complicated matrix elements, amplitude evaluation nevertheless takes largest fraction of CPU time

  This may change with a new algorithm for amplitude evaluation (OMega Virtual Machine)
- Plan: implement alternative algorithms and check possible improvements

Note: there are no systematic performance comparisons with other programs, yet. (Depends on too many parameters.)

2. Beams

Structure Function Chain

At high-energy colliders, beams have a structure.

The master formula is a convolution

$$\sigma(s) = \int dx_1\, dx_2\, f(x_1, x_2) \int dy_1\, dy_2\, g(y_1, y_2) \cdots \hat{\sigma}(x_1 x_2 y_1 y_2 \cdots s)$$

Some structure functions are of single-beam type

$$f(x_1, x_2) = f_1(x_1)\, \delta(1 - x_2) \quad \text{(or vice versa)}$$

- PDF (parton distribution functions) [hadron collider]
- ISR (initial-state photon radiation) [all colliders]
- EPA (effective photon approximation) [photon collisions]

Others are of double-beam type

- Beamstrahlung [electron/positron collider]
- Laser-backscattering spectrum [photon collider]

WHIZARD adopts a generic approach to the structure-function chain.

- Abstract data type and methods for structure-function object.
- Generic handling of color and spin.
- Structure-function objects can be implemented as event generators
- Optional: generation of transverse momentum and explicit radiated particles (beam remnants, photons, etc.)
- Structure-function chain connects s.f. objects
- Consecutive evaluation
- Extra parameters x_i are combined with the parameters of phase space module before entering the integration module.

  $\Rightarrow$ multi-channel integration can be applied to beam structure

Usage

Input in SINDARIN command language

Hadron collider example

```
sqrts = 14 TeV
$pdf_builtin_name = "CTEQ6L"
...
beams = p, p => pdf_builtin
```

Lepton collider example

```
sqrts = 500 GeV
...
beams = "e+", "e-" => circe2 => isr
```

PDF (hadron colliders)

For PDF evaluation:

Standard package LHAPDF

must be linked to WHIZARD at compile time. Then, all LHAPDF structure functions that the user has downloaded, are available.

Alternative, for the impatient:

Built-in interface and data for a few commonly used structure functions (CTEQ6 etc.)

can be used anytime, no download necessary.

Beamstrahlung

Beamstrahlung is an important issue for precision physics at Linear electron-positron Colliders. Even at a future high-luminosity Circular Collider, beamstrahlung cannot be neglected.

Classical interaction of the Coulomb fields of the two colliding beams, results in a statistical distribution of energy loss for the colliding particles.

Beamstrahlung is polarization-dependent.

Simulation of beamstrahlung: dedicated programs GuineaPig, CAIN.

Beamstrahlung Options in WHIZARD

1. Circe1:
   - Output of GuineaPig runs parameterized by smooth functions
   - Usable either as structure functions or as generator
   - In WHIZARD: analogous to PDF

   Caveat: Only fixed number of hard-coded parameter sets (ILC)

2. Circe2:
   - Output of GuineaPig/CAIN runs parameterized by histograms
   - Usable as generator
   - In WHIZARD: exchangable with Circe1

   Caveat: Requires histogram-data file for given parameter set

3. Beam-events file:
   - Output of GuineaPig/CAIN runs used directly
   - Usable as pseudo-generator
   - In WHIZARD: also analogous

   Caveat: Beam-event file has finite number of events

Initial-State Radiation

Photon radiation from incoming charged particles
enhanced by powers of $\log \frac{s}{m^2}$

$$f(x) \approx \epsilon(1-x)^{-1+\epsilon} \qquad \text{with} \qquad \epsilon = \frac{\alpha}{\pi} Q^2 \log \frac{s}{m^2}$$

$\Rightarrow$ important for electrons/positrons, less so for protons

Number of radiated photons: undefined (Poisson distribution)

Energy of radiated photons: smooth distribution, singular at $x = 1$

$\Rightarrow$ described by effective structure function

WHIZARD implementation: result of all-order soft resummation and
third-order explicit (parameterized) calculation by Skrzypek/Jadach, 1991.

Limitations of the effective structure function approach:

Individual radiated photons cannot be reconstructed. WHIZARD produces a single photon (optionally with a logarithmic p_T distribution) that carries the missing energy.

Improved exclusive description planned for future precision lepton-collider studies.

Polarization

Lepton-Collider beams will be polarized. Polarization direction and polarization fraction is arbitrary. Polarization of some final-state particles is also observable.

WHIZARD uses a density-matrix formalism (flavor, spin, color) for all internal calculations.

- ⇒ Input of arbitrary polarization matrices is possible for incoming beams
- ⇒ All quantum correlations can be retained for spin (and color).
- ⇒ ... or can be traced over whenever a part of the calculation is not available in density-matrix form

Internal Data Structures:

`quantum_numbers_t`: sparse matrix with complex entries for correlated spin/color/flavor information, stored as tree structure

`interaction_t`: + momenta and parent-daughter relationships

`evaluator_t`: + pointers and method for fast repeated evaluation of interaction products and squares

3. QCD

QCD in WHIZARD

WHIZARD applications have been focusing on

- Lepton Collider Simulations
- Electroweak interactions

These depend on QCD less than typical hadron-collider/jet physics applications. Therefore, WHIZARD's QCD implementation is less advanced than in other universal MCs.

Nevertheless, QCD in WHIZARD is implemented

- Completely to leading order (tree-level / leading-log shower)
- Partly to next-to-leading order, currently under active development

Color-Flow Formalism

$\Rightarrow$ Fabian Bach's talk last week

Color Flow: describe color in terms of connections between colored particles (instead of basis vectors). Full QCD requires color-singlet subtraction.

For amplitude calculation, OMega (interfaced from WHIZARD) uses a color-flow basis

- $\Rightarrow$ color algebra is trivial
- $\Rightarrow$ no basis transformation required for interfacing with shower

This is not essential for WHIZARD!

The color-flow basis is required by standard shower algorithms, therefore part of the LHA/LHEF event-format standard.

WHIZARD can interface any matrix-element program. To make the color information available, it has to transform it into the color-flow basis.

Parton Shower

WHIZARD provides a leading-log parton shower, in various incarnations:

- Write partonic events to file, in LHEF format. Run external parton shower.
- Internally call PYTHIA shower.
  - PYTHIA 6 is included in the WHIZARD package, for this purpose
  - PYTHIA parameters can be set from SINDARIN script
- Internally call analytic parton shower (C.Bauer et al.)
  - Unique algorithm, implemented only in WHIZARD (initial and final state)
  - Validated with LEP data

Hadronization can be done externally, or internally via the included PYTHIA package. WHIZARD does not provide its own fragmentation/hadronization model.

Matching

At leading order, matching finite-order matrix elements with parton shower can be done by the MLM scheme

$\Rightarrow$ implemented within WHIZARD

NLO

requires:

- Automatic construction of radiation and loop process components
- NLO (loop) matrix elements
  - $\Rightarrow$ external program replaces OMega, currently GoSAM
- Subtraction scheme for real and virtual corrections
  - $\Rightarrow$ currently FKS subtraction
- NLO matching to parton shower: next step

These parts are not yet publicly available. Preview: cf. tutorial

Plan for QCD

Precision physics at e^+e^- colliders will require more precise QCD description. (Partly also LHC.)

- New, fast matrix-element evaluation with OMega Virtual Machine (B. Choukoufe)
- Public version with NLO matrix elements (C. Weiss)
- Jet observables
- CKKW Matching Algorithm
- Improved shower and NLO matching

For setting priorities, shower tuning, and more detailed validation:
cooperation with experimental groups would be welcome

4. Computing Issues

History of WHIZARD

WHIZARD 1

WHIZARD (first versions) was written for electroweak processes at e^+e^- colliders: phase-space module for automatically generated matrix elements. Three options:

1. Amplitudes generated by CompHEP (for comparison)
2. Amplitudes generated by MadGraph (note: MadEvent didn't exist yet)
3. Amplitudes generated by OMega

$\Rightarrow$ used for event samples and studies (TESLA, ILC, CLIC)

$\Rightarrow$ LHC studies also possible

Why did we discontinue support for WHIZARD 1?

⇒ WHIZARD 1 was essentially developed by one person. Code was not well maintainable, growed much beyond original scope.

⇒ Too many data structures hard-coded and difficult to extend (despite object-oriented and modular structure)

⇒ Mixture of programming languages, interdependence of Make and PERL scripts

⇒ Usage patterns: cuts and selection, analysis, parameter scans didn't work well with fixed-format input files

Last stable version: WHIZARD 1.97

WHIZARD 2

Main Changes

- Steering language SINDARIN unifies all input (parameters, cuts, workflow)
  - $\Rightarrow$ and serves as complete scripting language for a wide range of more complex use cases
- Introduction of `interaction_t` and related data structures unified the internal handling of quantum states.
- Fully spin-correlated on-shell decays in the simulation pass, if requested
- Parton Shower and Matching

Extensible Software Design

WHIZARD 2.2: Major part of the code rewritten in 2012–13, because

Fortran 2003 supports abstract data types and thus standard software design patterns that allow us to cleanly

- separate interface (abstract) from implementation (concrete)
- introduce mock implementations for automatic test suite
- exchange algorithms on a case-by-case basis, controlled by the user

[Would have been straightforward also in C++. But changing the main coding language would have raised lots of other issues.]

Current production version: WHIZARD 2.2.2

Examples for abstract interfaces in WHIZARD:

- Matrix-element / amplitude definition, generation, and evaluation
- Random-number generator
- Phase-space parameterization
- Multi-channel integration
- Structure-function definition and evaluation
- Event-sample transformation (shower, decay, hadronization etc.)
- Event file formats

The requirements for adding (e.g.) a new event format are thus well defined, checked by the compiler, and local in code.

Parallel Computing

OpenMP

If configured with OpenMP enabled, on a multi-core computer, WHIZARD will

- evaluate the sum over helicities in parallel
- evaluate the inverse transformations of Wood phase space in parallel

making use of all available cores.

MPI

The VAMP integrator supports MPI (multi-CPI machine or HPC cluster)

- sample different parts of the VAMP grid in parallel

??? currently not yet exploited by WHIZARD

Why Fortran?

WHIZARD uses OCaml (OMega amplitude generator), Fortran 2003, and Make. Works with gfortran 4.8.

Some Reasons for this choice

- WHIZARD initially had to interface Fortran programs
  (Fortran interoperable with C since Fortran 2003)
- Multi-dimensional arrays as part of the language.
- Safe dynamic memory allocation / deallocation
- Module encapsulation vs. C++ classes/namespaces
- INTENT declarations for parameter passing vs. pointers in C++
- KIND declaration for trivial switching to higher precision
- Abstract data types (Fortran 2003) vs. virtual classes in C++

Some Drawbacks of Fortran

- No generic programming (cf. STL in C++)
- Compile cascades (solved in Fortran 2008)

Software Development Issues

Since WHIZARD 2, we have gradually adapted a more professional attitude towards software management:

- Versioning and software repository on Hepforge: subversion
- Bug tracker (on HepForge).
- Installation decoupled from user run directories.
- Automatic configuration by autotools framework, enforcing standard structure.
- Completely automatic test suite, broken down into unit and functional tests.
- Test suite automatically run for each commit by continuous integration server (Jenkins)

5. Conclusions

Conclusions

WHIZARD for users

WHIZARD is suited for all kinds of (perturbative) processes at LHC, ILC, and future colliders.

Unique combination of algorithms and models for matrix elements, phase space, event handling

Currently under way: more accurate (and NLO) QCD and SM, more detailed beam description (radiation) for e^+e^-, various efficiency improvements

WHIZARD for developers

The collaboration is open for discussion, external contributions and genuine participation. Modular software design and management should be helpful.

References

Second WHIZARD Forum

Castle of Würzburg, March 16–18, 2015

whizard.hepforge.org

whizard@desy.de