WHIZARD 3
Phase Space and Adaptive Sampling Algorithms

Wolfgang Kilian

CPPS, University of Siegen

CmF/IAL Kickoff, Bonn, March 20, 2024

The WHIZARD MC generator

Scope: High-energy scattering processes (SM and BSM)

- Tree-level ME code generator O'Mega
- spectra/PDF, PYTHIA, SM + BSM models, ...
- Multi-channel version of VEGAS integrator: VAMP by T. Ohl
- main application: off-shell event generation for e^+e^- (and $\mu^+\mu^-$)
- LHC, ...: fully supported but limited by resources

1999 Whizard 1

2007 Whizard 2 (internal structure, scripting, ...)

2021 Whizard 3 (NLO QCD+EW with OpenLoops/Recola/GoSam)

- VAMP integrator parallelized (MPI) but algorithm unchanged

The Whizard Team (2024)

U Siegen
WK, Pia Bredt, Nils Kreher, Tobias Striegl

DESY
Jürgen Reuter, Krzysztof Mękała

U Würzburg
Thorsten Ohl

U Warsaw
Filip Żarnecki (exp.)

Phase-Space Mapping and Sampling I

Problem: resonances + soft/collinear peaks + uncontrolled cuts + O(10–20) dimensions

Solution: Self-optimizing Multi-channel integration
$\Rightarrow$ multichannel event generation

$$\sigma = \int \mathrm{d}\Phi \, |M(p)|^2 = \sum_c \int \mathrm{d}x_c \, \alpha_c(p(x_c)) \, \frac{\mathrm{d}\Phi}{\mathrm{d}x_c}(p(x_c)) \, |M(p(x_c))|^2$$

Choice of $x_c(p)$ and $\alpha_c(p)$: cancel $|M|^2$ peaks

- Whizard approach: denominator structures determine channels

MC performance parameters

Integration:

accuracy γ:

$$\text{error} = \frac{\gamma}{\sqrt{N}} \qquad \text{with } \gamma \stackrel{?}{=} O(1)$$

Increase γ value:
$\Rightarrow$ spend matrix element calls for optimizing (training) phs mappings

Exclusive quantities:

unweighting efficiency ϵ:

$$\epsilon = \frac{N_{\text{evt}}}{N_{\text{call}}} = \frac{w_{\text{avg}}}{w_{\text{max}}} \qquad \text{with } \epsilon \stackrel{?}{=} O(1)$$

Matrix element calls are costly!

Phase-Space Mapping and Sampling II

Improve phase-space sampling: Machine-Learning methods

- Further mapping of integration manifold, parameterized

$$\int \mathrm{d}x = \int \mathrm{d}y \, \frac{\mathrm{d}x}{\mathrm{d}y} \qquad \text{where } x(y) = x(y; \beta_1, \ldots \beta_N) \tag{1}$$

$\Rightarrow$ optimize β_i: adaptation / training / learning
$\Rightarrow$ iterate this: **deep** learning
cross-talk between integration channels!

- Tradeoffs:
  - How much precious calls can we spend for training?
  - How many calls do we eventually save in unweighting?
  - How much can we compute in parallel?

Machine-Learning Old and New

VAMP (VEGAS)

- Mapping piecewise linear (binned)
- All dimensions at once (single layer)
- Parameters optimized in parallel, based on local variance
- γ and ϵ good but limited

Newer algorithms (NIS etc.)

- Smooth mapping (splines / binned)
- Dimensions mapped iteratively (coupling layers)
- Parameters optimized iteratively, global variance
- Achievable γ and ϵ: better?

Project (SFB-TRR 257: A2b): Combine the best of two worlds?

SI (WK) + KA (G. Heinrich) + HD (A. Butter / T. Plehn)

MC Generation Issues: Summary

Data Input: simple but uncontrolled

Data Output: event samples in standard formats

Algorithm: search for optimal topological/invertible mappings

Parallel execution: mandatory on several levels, communication

Training samples: most expensive part of the calculation

Need for: speed!