Implementation of a multi-core interface for RECOLA and RECOLA 2 in WHIZARD

DESY Summer Students Program 2018

Vincent Bettaque

University of Hamburg
Department of Physics

September 7, 2018

Abstract

The WHIZARD general purpose event generator and the RECOLA one-loop amplitude provider are introduced as methods to compute scattering events at particle accelerators numerically. We describe the issues that arise from using RECOLA and RECOLA 2 as loop providers in WHIZARD and suggest a partial solution to these problems that allows the usage of multiple RECOLA cores simultaneously. The remaining issues and their possible origins are discussed and suggestions for further improvements are laid out.

Contents

1 WHIZARD & RECOLA

We refer here to the official documentation for both software packages, since they can explain their programs far better [1][2][3][4][5]. Even though most of the following should be self-explanatory by cross-checking the code changes [6], knowledge of the Sindarin file format used in WHIZARD is especially beneficial for understanding how calculations are done in general.

2 Compatibility Issues

We now address the line of thought that led to the solution laid out in the next chapter. This is done because the motivation for the changes may not be obvious at first. We then describe the problem in detail and then name other issues that directly or indirectly result from fixing the first one.

2.1 Status Quo

The first sign of an apparent issue was observed when using RECOLA 2 in WHIZARD through the wrapper already available for the first version. The functional test `recola_7`, which employs RECOLA to compute the cross section of $e^+e^- \to t\bar{t}$ in NLO, fails because it returns a different value than expected. After first examination, it turned out that the wrapper in WHIZARD is (re-)setting the renormalisation scale μ_{IR} (`mu_ir` in code) of RECOLA only before each integration step and not before the RECOLA processes are generated. This is only possible by calling the routine `set_dynamic_settings_rcl` with parameter 1 before generation. As of RECOLA 2, though, this routine is deprecated, meaning that μ_{IR} is never set in this case. This proved to be an issue, since the design of WHIZARD doesn't easily allow μ_{IR} to be set before generation of the RECOLA processes. But since the renormalisation scale can vary during integration, setting μ_{IR} at each step would also still be necessary to produce the correct result. This led to the belief that the current way of doing things is inherently flawed.

Belief turned into conviction when it was found that the variable μ_{IR} is actually part of the library COLLIER, which is used internally by RECOLA. The COLLIER documentation explicitly states, that the results in the end should be independent of the value for μ_{IR}. This is further backed by the alternative loop provider OpenLoops which also uses COLLIER and can do without modifying μ_{IR}.

The current guess is a problem regarding the subtraction terms in the matrix element. After closer inspection, it turned out that the test `recola_7` only uses RECOLA to compute the virtual part of $e^+e^- \to t\bar{t}$ in NLO. As the other amplitudes are tree-level, the WHIZARD-internal OMEGA integrator is used to calculate them. But since both Born and virtual amplitudes have regularised divergences that exactly cancel when added together, both should be computed by RECOLA to make sure that the divergences are removed properly. This leads to the problem that is discussed in this report, namely using multiple RECOLA cores at once.

2.2 Single-Core Implementation

Before the modifications described in this report, WHIZARD was only able to make use of RECOLA processes to compute one matrix element component during execution. This is mostly due to the way RECOLA manages and generates its processes. An example of this is shown in listing 2.1. RECOLA requires all processes to be defined upfront, only then can they be generated and computed. Any attempt to define processes after generation or to reset before generating processes results in an error on behalf of RECOLA.

```
1  program main_rcl
2
3    use recola
4
5    implicit none
6
7    integer, parameter :: dp = kind (23d0) ! double precision
8    real (dp)          :: p(0:3,1:6)
9    character(len=100) :: modelname
10
11   call set_output_file_rcl('*')
12
13   call set_print_level_squared_amplitude_rcl(1)
14
15 ! #1: Define process
16   call define_process_rcl(1,'u u~ -> g g tau+ tau-','NLO')
17
18 ! #2: Generate process
19   call generate_processes_rcl
20
21 ! Momenta of the phase-space point.
22   p(:,1) = [4000.0000000000d0,    0.0000000000d0,    0.0000000000d0,
       4000.0000000000d0]
23   p(:,2) = [4000.0000000000d0,    0.0000000000d0,    0.0000000000d0
       ,-4000.0000000000d0]
24   p(:,3) = [2387.4445571379d0,-2131.7219821216d0,  677.6712380335d0,
       -834.5145879427d0]
25   p(:,4) = [2084.0108209587d0, 1206.0274745508d0, 1266.0449626178d0
       ,-1133.8999008430d0]
26   p(:,5) = [1954.1326742459d0, -173.3442838631d0, -836.2617619034d0,
       1757.6269608155d0]
27   p(:,6) = [1574.4119476575d0, 1099.0387914340d0,-1107.4544387478d0,
       210.7875279701d0]
28
29 ! #3: Compute process
30   call compute_process_rcl(1,p,'NLO')
31
32 ! #4: Reset RECOLA and remove processes
33   call reset_recola_rcl
34
35 end program main_rcl
```

Listing 2.1: Example of a simple program using RECOLA

But since WHIZARD decides freely which integration cores it chooses to compute every component with, it is not known at compile time how many RECOLA cores and thus processes are needed overall. Only inside a single core the number of required RECOLA processes is set by the number of flavours (`n_flv`) that are being considered. This forced WHIZARD to reset RECOLA completely once a new matrix element was to be computed, effectively removing all processes of the previous RECOLA core. And since all cores are defined in the beginning, only the last defined RECOLA core survives.

The exact dynamics of this behaviour are implemented in `src/recola/recola.nw` and happen on two layers: in the wrapper, which is partly managed by a singleton object of type `rcl_controller_t`, and the core layer which, mostly consists of `recola_writer_t` and `prc_recola_t`. Both layers interact with each other through wrapper routines which start with `rclwrap`.

The only purpose of `rcl_controller_t` in this implementation is to keep track of the RECOLA process IDs handed out to the various instances of `prc_recola_t` and to make sure that things are done in the order intended by RECOLA. All other wrapper calls are almost directly propagated towards RECOLA and don't affect the singleton object. Calling `rclwrap_reset_recola` resets both the singleton and RECOLA.

2.3 Multiple Flavor Lists

The following issues all arose after implementing the solution to the previous problem since parts of the program relied on there being only one available RECOLA core.

The first problem that became apparent this way was the handling of flavor lists. They contain one event in all of the given flavor combinations. Usually only one explicit flavor is considered and thus the lists only contain a single process string. But multiple flavors are allowed to be considered in one WHIZARD process, as can be seen in listing 2.2:

```
1 process recola_6_p1 = E1, e1 => u:d, U:D
```

Listing 2.2: Example of a WHIZARD process with multiple quark flavors (up and down)

The flavor lists for each process are then generated by WHIZARD as files ending with `.flv.dat`. These files are then read by the RECOLA writer and used to initialise the cores. This means that one core can "contain" as many RECOLA processes as entries in the corresponding flavor list (denoted by `n_flv`). The issue arises from wanting to use two RECOLA cores to compute both the Born/virtual and the real amplitude of an event in NLO. Since the file name is always the same irregardless of component, the flavor lists of multiple core are overwritten by each other. This isn't a problem if the process string is the same (i.e. with Born and virtual). But computation of the real amplitude uses a different process string since it deals with additional radiation and hence extra particles in the event. It is thus necessary to create a distinction between the flavor lists of different cores.

2.4 Mismatching Indices

Another issue resulting from flavor lists are the indices used in the integration routines. For example in `prc_recola_compute_sqme` only the index `i_flv` ranging between 1 and `n_flv` is used to refer to the RECOLA processes. This is fine for a single core, since there is only one flavor list and the flavor index equals the process index because of that. But for multiple cores, there needs to be a way for these cores to keep track of the RECOLA processes they manage that depends on `i_flv`. This includes explicitly distinguishing between RECOLA cores and RECOLA processes, since they are sometimes used as if they are equivalent. An example for that is `prc_recola_compute_sqme`, where `i_flv` is used, and `prc_recola_compute_sqme_virt`, where the core variable `object%recola_id` is used for the same purpose.

3 Implementation Details

What follows is a summary of the steps taken to allow multiple RECOLA cores at once. The explicit code changes can be traced in the branch `allow-multiple-recola-processes` of the WHIZARD development git repository [6].

3.1 The Wrapper

The first step towards a solution was to extend `rcl_controller_t` in the RECOLA wrapper. Since WHIZARD initialises all cores at once before integration, no RECOLA processes are used at this point, which means that a reset of RECOLA will have no impact, as long as all process defined so far are redefined afterwards. This fact was used to create a new type `rcl_process_t`, which combines all the information required to define a process in RECOLA namely `id`, `process_string` and `order`. An array of that type called `processes` was then added to `rcl_controller_t` together with the routine `rcl_controller_add_process`, which adds a `rcl_process_t` to the array and increases the array size if necessary. The routine `rcl_controller_activate` is now used to initialise this array which means that `rclwrap_request_generate_processes` now has to be called before adding processes (a name change could be considered but wasn't done).

To keep track of the amount of processes in the array, `n_processes` was introduced. The already available `recola_id` isn't used for this purpose, since in theory not all available process IDs have to be present in the array. To further establish this difference, `rclwrap_get_n_processes` was renamed to `rclwrap_get_current_recola_id`, since it returns `recola_id` internally.

The final change in the wrapper was made to the way processes are defined. A direct routine to do this was removed by making `rclwrap_define_process` private, since it could cause issues with the new paradigm. Instead, `rclwrap_define_processes` was introduced which goes through the new `rcl_process_t` array and defines all the processes that are located there. If processes have been defined before (indicated through the new `defined` flag), RECOLA is reset entirely beforehand, making sure that only the processes in the array are defined afterwards. This allows to add and define new processes even after generation without losing previously defined processes. As implied earlier, this means that the changes made allow for an arbitrary amount of RECOLA processes and thus cores to be used in WHIZARD.

3.2 The Writer

The `recola_writer_t` needed the least amount of work, but the changes were important none the less. Its ID determines the ID of the whole calculation done by the corresponding core and also the the file name of the flavor list that is used. Because of that, this was the culprit for allowing the flavor lists to overwrite one another, since the writer ID was the same independent of matrix element component. This was solved by adding the already available component-specific prefixes in `recola_def_init` to distinguish between the different writer IDs. The rest worked automatically and only the functional tests had to be adjusted to recognise the new IDs.

3.3 The Core

The last big issue of the project was finding the correct RECOLA process IDs to use for integration. As described in Section 2.4, the old system relied on there being only one RECOLA core and had ambiguities regarding the correct process index to use for integration, anyway.

This issue was resolved by extending the RECOLA core `rcl_process_t`, such that all process IDs associated to it are stored in an internal array `recola_ids` with length `n_flv`. This means that `i_flv` can be used to access the correct process referring to that flavor since they are being added to the array with the same index through the routine `prc_recola_register_processes`. There, `i_flv` corresponds to the line of the `.flv.dat` file in which the process is found. Since now all process IDs can be tracked, the misleading variable `recola_id` was removed, since it always only referred to first RECOLA process despite there possibly being multiple.

Since almost all routines have access to `i_flv` through the parameter list, substituting `i_flv` and `object%recola_id` with `object%recola_ids(i_flv)` at the appropriate places took no effort. Only the changes required for the routine `prc_recola_replace_helicity_and_color_arrays` are not trivial and discussed in section 4.2. Once helicities are properly implemented, another index `i_hel` will become relevant. This means that the ID array would need to be expanded by one dimension, which can be done easily.

4 Unresolved Issues

The changes described in this report were the most important ones to allow both RECOLA and RECOLA to be fully made use of by WHIZARD. But there are still areas left in which inspection is needed to ensure that a regular usage is possible. The main points are described in the following.

4.1 Color & Spin Correlations

It is now possible to calculate the Born, real and virtual amplitudes using RECOLA in WHIZARD. But to find the subtraction terms for real and virtual, it is necessary to compute the color and sping correlations for the process. RECOLA already provides appropriate routines with `compute_colour_correlation_rcl`, `compute_spin_colour_correlation_rcl` and `compute_spin_correlation_rcl`. Because of that, it is probably only necessary to write a wrapper around it which is called at the same spot in the program as the wrappers for other libraries. This is probably done somewhere in the `whizard.nw` file.

4.2 Replace Helicity & Color Arrays

As said in section 3.3, the index `i_flv` isn't available in the routine `prc_recola_replace_helicity_and_color_arrays`. This is because the replacement has to be done for every flavor (and helicity) index at once, but is currently only done for `i_flv = 1`. Expanding this behavior for all flavors requires an additional array dimension, which also has to be then considered at the points where the arrays are needed.

4.3 Scale Dependence

Because of the previous issues, the results of NLO matrix elements are technically still scale dependent. This means that `mu_ir` is still actively reset with each integration step. Once the color and spin correlation are properly implemented though, one can remove this resetting and then test, if the results with real, virtual and sub part calculated with RECOLA still change for different `mu_ir` etc. This should be the case in theory, so further dependence may point at an unknown error in the code.

Bibliography

[1] W. Kilian, T. Ohl, J. Reuter, WHIZARD: Simulating Multi-Particle Processes at LHC and ILC , Eur.Phys.J.C71 (2011) 1742, arXiv: 0708.4233 [hep-ph]

[2] M. Moretti, T. Ohl, J. Reuter, O'Mega: An Optimizing matrix element generator, LC-TOOL-2001-040-rev, arXiv: hep-ph/0102195-rev

[3] WHIZARD Manual

[4] Stefano Actis, Ansgar Denner, Lars Hofer, Jean-Nicolas Lang, Andreas Scharf, and Sandro Uccirati. RECOLA: REcursive Computation of One-Loop Amplitudes. Comput. Phys. Commun., 214:140–173, 2017. arXiv:1605.01090, doi:10.1016/j.cpc.2017.01.004.

[5] Ansgar Denner, Jean-Nicolas Lang, and Sandro Uccirati. Recola2: REcursive Computation of One-Loop Amplitudes 2. Comput. Phys. Commun., 2017. arXiv:1711.07388, doi:10.1016/j.cpc.2017.11.013.

[6] Multi-Core Git Branch