IFJPAN-IV-2009-10

January 2010

Universal Interface of TAUOLA

Technical and Physics Documentation

N. Davidson[a,b], G. Nanava[e], T. Przedzinski[c], E. Richter-Was[d], Z. Was[a]

[a] *Institute of Nuclear Physics, Polish Academy of Sciences,*
ul. Radzikowskiego 152, 31-342 Cracow, Poland
[b] *University of Melbourne, Department of Physics*
Australia.
[c] *The Faculty of Physics, Astronomy and Applied Computer Science,*
Jagellonian University, Reymonta 4, 30-059 Cracow, Poland.
[d] *Institute of Physics, The Faculty of Physics, Astronomy and Applied Computer Science,*
Jagellonian University, Reymonta 4, 30-059 Cracow, Poland.
[e] *Physikalisches Institut Bonn University, Nussallee 12, D-53115 Bonn, Germany*

Abstract

Because of their narrow width, τ decays can be well separated from their production process. Only spin degrees of freedom connect these two parts of the physics process of interest for high energy collision experiments. In the following, we present a Monte Carlo algorithm which is based on that property. The interface supplements events generated by other programs, with τ decays. Effects of spin, including transverse degrees of freedom, genuine weak corrections or of new physics may be taken into account at the time when a τ decay is generated and written into an event record. The physics content of the C++ interface is already now richer than its `FORTRAN` predecessor. April 15, 2011

† This work is partially supported by EU Marie Curie Research Training Network grant under the contract No. MRTN-CT-2006-0355505 and by Polish Government grant N202 06434 (2008-2011).

Table of Contents

PROGRAM SUMMARY

Manuscript title: Universal interface of TAUOLA Technical and Physics Documentation

Authors: N. Davidson, G. Nanava, T. Przedziński, E. Richter-Wąs, Z. Wąs

Program title: TAUOLA++, versions 1.0.2, 1.0.3, 1.0.4

Licensing provisions: None

Programming languages: C++, FORTRAN77

Operating system(s) for which the program has been designed: Linux, MacOS

RAM required to execute with typical data: <10MB

Has the code been vectorised or parallelized?: No

Number of processors used: 1

Supplementary material: None

Keywords: tau decays; Monte Carlo simulation; Event Record interface; event generation; spin effects;

CPC Library Classification: 100: Physics of Elementary Particles and Fields

External routines/libraries used: HepMC v2.0 or later.

CPC Program Library subprograms used: Demonstration programs use PYTHIA, MC-TESTER

Nature of problem:
The code of Monte Carlo generators often has to be tuned to the needs of large HEP Collaborations and experiments. In particular τ lepton decays need to be added (or modified) to the previously generated (or measured) events encapsulated in an event record.

Solution method:
The new algorithm, the universal interface of TAUOLA which works with the HepMC event record of C++ applications is documented. It uses the τ decay generator as described in [2] with the updates explained in [1]. For the new interface spin treatment was improved. For example it features complete spin effects in processes mediated by Z/γ^* interactions. The effects of electroweak corrections can be taken into account in this case as well. In general, the program superseeds its FORTRAN predecessor [1]. The event record analysis as well as initialization is also modernized.

Restrictions:
The input event record must meet the requirements described in Section 2.3.1 of the documentation.

Unusual features:
Two sets of installation scripts; an additional tool for calculating tables for electroweak corrections.

Running time:
Depends on the size and complexity of the events. Small events (<50 particles), require 1 to 7 minutes for processing 1M events on PC/Linux with a 2.4GHz processor.

References:

[1] P. Golonka, B. Kersevan, T. Pierzchała, E. Richter-Wąs, Z. Wąs, M. Worek, Comput. Phys. Commun. **174** (2006) 818.

[2] S. Jadach, Z. Was, R. Decker and J. H. Kühn, Comput. Phys. Commun. **76** (1993) 361.

1 Introduction

Before we can present `TAUOLA Interface`, now in C++, we have to briefly explain the applications and constraints which helped us to define the main tasks that an interface for a τ decay generator has to resolve.

In the present day experiments at High Energy Physics accelerators, the interpretation of results has become increasingly involved. Not only is the detector response complex, but some theoretical effects need to be removed. Otherwise results are difficult to interpret for the non-specialist. For that purpose the concept of work with realistic and idealized observables was introduced as well as, finally, pseudo-observables which can be easily understood by theorists, such as W and Z masses or couplings.

Good examples of this approach were measurements of the two-fermion final states at LEP. Because of the increasing precision of the experimental measurements, the definitions of the quantities to be measured, while simple at first, later evolved into several options [1], each based on the properties of individual detectors and each requiring individual discussion of the systematic error.

One could assume, that if all theoretical effects are embodied into one theoretical black-box and, experiments while using it tune parameters (representing pseudo-observables) to the data, the interpretation of the observed effects could be separated into theoretical and experimental components. Unfortunately this strategy is limited, as it leaves little room for cases where theory and experimental effects are convoluted: the size and even nature of the theoretical corrections depend on the experimental conditions. Such discussion on observables involving τ decays can be found in [2].

For the LHC experiments, τ decays are not of primary interest in themselves, but rather will be used to measure properties of τ production processes. Let us explain this using examples. The physics effects necessary for the prediction of hard processes at the LHC experiments can be separated into several parts, among them: parton showers, the underlying event and structure functions, final state QED bremsstrahlung, QED bremsstrahlung interference between initial and final states and finally the hard process including electroweak corrections. Such separation is not only for the convenience of organizing theoretical work but provides an efficient and flexible structure to the framework used for experimental data analysis (see eg. [3]). Some such building blocks are of genuine theoretical interests, some others are not so much. The hard process usually depends on the parameters intended for the measurement, eg. the W or Higgs mass, or new coupling constants. Other building blocks may be less interesting, nonetheless they may affect the results of measurements. This is certainly true in the case of the underlying event, missing transverse energy or p_T distributions generated from parton showers [2]. It may also be the case for QED final state bremsstrahlung or initial-final state interference (where potential difficulties may be expected [4] and predictions may need to be fixed with the help of experimental data).

The black-box approach, where all simulation segments are put together by theorists, may look advantageous to the experimental user. However in such cases one has less flexibility to distinguish experimental effects from the theoretical ones, thus limiting control on the systematic errors. These particular problems may be left unnoticed. Typically the difficulties will not affect all observables. Unfortunately, complications tend to show up only when more detailed discussion on the systematic errors of experimental analysis is performed.

In the present document we discuss the implementation of τ decays into a simulation chain as a separate module, now in C++, which can be configured by the end user. For the

purpose of generating τ decays themselves, the TAUOLA library, as described in [5, 6, 7] is used. This part of the code is expected to be a black-box for the High Energy experimental user. At present, from the technical side, the black-box consists of the same FORTRAN code as described in [8]. We will call it TAUOLA FORTRAN. Such organization makes it easy for low energy phenomenologists or experimentalists to work on this part of the code, such as the activities described in [9], leading to the new parametrization of hadronic τ decay currents becoming available for High Energy experimental users in a rather straightforward way.

The role of the interface is to prepare information on the τ (four-momentum, spin state) in a format which is understood by TAUOLA FORTRAN, and as a post processing step to return (insert) τ decay products to the primary event record. Finally, the role of such interfacing code is to calculate dedicated weights from the production process information as well as from the decay, and unweight accordingly to standard MC procedures. Spin effects, electroweak corrections and also effects of anomalous couplings can be introduced in this way.

1.1 C++ Interface and its FORTRAN predecessor.

A rather modest version of TAUOLA Universal Interface based on the FORTRAN HEPEVT event record is described in [8], we will call it TAUOLA Fortran Interface. A new version of TAUOLA Universal Interface based on HepMC [10], the most popular event record of C++, will be documented here. It also includes new functionalities. We will call it TAUOLA C++ Universal Interface, or, if no ambiguity could arise simply TAUOLA C++ Interface or just TAUOLA Interface if it is clear that the C++ not the FORTRAN version is in mind[1]. The PHOTOS generator for QED bremsstrahlung in decays, which was previously distributed together with TAUOLA FORTRAN, is not discussed in the present paper[2]. It is now embodied in a separate module [11, 12] of the Monte Carlo simulation chain, as the PHOTOS generator has found significant applications outside the domain of τ decays. That is why, we distribute the C++ version of PHOTOS separately [13, 14].

Important physics improvements, with respect to the TAUOLA FORTRAN Interface implementation, described in [8], comes with the code presented here. In particular, transverse spin correlations have been implemented for processes mediated by Z/γ^* and genuine electroweak corrections are now available for such processes. With the new interface, it is rather straightforward to implement effects beyond standard model physics. Only read-in data-tables should be replaced, no modification to the code itself is needed. Further minor extensions include an algorithm to decay a single τ with a user defined polarization, or the availability of methods to access generated τ leptons helicity states.

This documentation describes version 1.0.4 of the interface[3].

[1] The main class of TAUOLA C++ Universal Interface is called Tauola.

[2] PHOTOS was distributed together with TAUOLA since Ref. [8]. That is why it is present in our TAUOLA FORTRAN but will be not used here.

[3] Version 1.0.3, with respect to version 1.0.2 (documented in our paper first preprint v.1) introduced changes affecting the use of Plots and Debugging methods, see Appendix C.11. Separation of the main programs in the examples and testing directories was introduced. Configuration of this part of the distribution is now separated from configuration of the TAUOLA libraries as well. Changes were introduced for the sake of better modularity and clarity for the first use and installation. In Version 1.0.4 alternative initialization scrips were introduced as an option. See section B.2 for details. Finally some bug fixing and code cleaning was performed as well.

2 Requirements for TAUOLA Interface

2.1 General Requirements

For a τ decay to be generated it is enough to know its spin state and define the frame in which the decay should be performed. In case there are more than one τ lepton in the final state, the quantum spin state of both (or more) τ leptons must be provided in the form of a density matrix. The exact algorithm for the generation of spin correlations has existed since the papers [5, 6, 7]. However, for the algorithm to function, the density matrix must be known exactly as well.

In practice, the actual form of the spin density matrix (in our present paper it will be called R_{ij}, exactly as in the original TAUOLA FORTRAN and its documentation) is often available with some approximations only. With the increasing precision of experiments, one may need to remove certain approximations introduced into R_{ij}. Already now, our program features, as an option, complete spin effects in decays of τ pairs originating from the annihilation of quarks. Effects of genuine weak corrections are included, and an extension for the implementation of new physics signatures is straightforward.

One should not forget that the density matrix itself is not the only place approximations occur. Effects of higher order QCD corrections need to be taken into account to define the kinematical configuration of initial partons used in spin density calculations (otherwise the density matrix for each individual process would have to be provided). At present, this is available in the leading (collinear) approximation only.

Before we will discuss details of specific implementation, let us recall first, the minimal list of steps the interface has to perform, independent of the programming language and data structure used.

1. The τ lepton or lepton pair(s) have to be localized in the event record. For processes mediated by W bosons (or charged Higgs), ν_τ has to be localized as well.

2. If possible, the hard process leading to τ production has to be determined. This is necessary to control transverse spin correlations. Preferably minimal information from the host generators should be used. This is to reduce dependence on the host program[4].

3. Flavours and orientation of fields entering the production vertex for intermediate states, such as Z/γ^*, have to be reconstructed too. This orientation (with respect to $\tau^\pm$ rest-frames) is necessary for calculation of the $\tau^\pm$ spin density matrix, if the spin of intermediate state is different from zero.

4. The relative orientation of τ^+ and τ^- rest-frames should be established and respected by Lorentz transformations.

5. Transformation of τ decay products from the τ rest-frame to lab frame has to be performed and the event record has to be completed with τ decay products.

[4]We target also applications when event records will be filled by measured data. For example, measured $\mu^+\mu^-$ events can be modified and final state muons replaced appropriately with $\tau^+\tau^-$ pairs. Then the generation of τ lepton decays is necessary.

2.2 C++ Specific Requirements

The C++ version of TAUOLA Interface implements all functionalities of its predecessor, TAUOLA Interface coded in FORTRAN [8]. It can be attached to any Monte-Carlo program where τ's are generated, provided its output is available through a HepMC [10] event record.

This condition is not very restrictive, is seems that HepMC will remain a generally accepted standard for the near future. However, already now several different options for how HepMC is used are widespread. The possibility to flexibly adaptate our event record interface to different options has been considered in the design, drawing experience from MC-TESTER [15, 16]. We have also envisaged the possibility that HepMC may one day be replaced by another standard of event record, and we have provided an easy way to extend the interface to a possible new event record standard.

2.3 Object Oriented Event Records – The Case of HepMC

In adapting TAUOLA Interface to the C++ event record format the difference between the HEPEVT event record used in the FORTRAN version of TAUOLA Interface and HepMC event record which is used for the C++ based interface has to be taken into account. In the first case the whole event was represented by a common block containing a list of particles with their properties and with integer variables denoting pointers to their origins and descendants. The HepMC event structure is built from vertices, each of them having pointers to their origins and descendants. Links between vertices represent particles or fields. In both, FORTRAN and C++ cases, the event is structured as a tree[5], the necessary algorithms are analogous, but nonetheless different.

In HepMC version 2.04, an event is represented by a GenEvent object, which contains all information regarding itself, including event id, units used for dimensional quantities in the event and the list of produced particles. The particles themselves are grouped into GenVertex objects allowing access to mother and daughter particles of a single decay. Vertices provide an easy way to point to the whole branch in a decay tree that needs to be accessed, modified or deleted if needed. The information of a particle itself is stored in a GenParticle object containing the particle id, status and momentum as well as information needed to locate its position in the decay tree. This approach allows traversing the event record structure in several different ways.

The HepMC event record format is evolving with time, making it necessary to adapt the code to the new versions. HepMC version 2.05 is used as a reference. In the case of version 2.03 restrictions on methods for units conversion have to be taken into account, for details see Appendix B.6. One should keep in mind that future changes to HepMC may restrict backward compatibility.

Evolution of the HepMC format itself is not a crucial problem. On the contrary, the conventions for how physics information is filled into HepMC represent the main source of technical and also physics challenge for our interface. This is quite similar to the previous HEPEVT - FORTRAN case. Let us discuss this point in more detail now.

[5] At least in principle, because in practice its properties may be rather of a graph without universally defined properties. This makes our task challenging.

2.3.1 Event Record Structure Scenarios

While many Monte-Carlo generators (eg. `PYTHIA 8.1` [17], `HERWIG++` [18]) store events in `HepMC` format, the representations of these events are not subject to strict standards and can therefore vary between Monte-Carlo generators or even physics processes. Some examples of these variations include the conventions of status codes, the way documentary information on the event is added, the direction of pointers at a vertex and the conservation (or lack of conservation) of energy-momentum at a vertex. Below is a list of properties for basic scenarios we have observed in Monte-Carlo generators used for testing the code.

This list will serve as a declaration for convention of `HepMC` filling, which the interface should be able to interpret correctly.

- **4-momentum conservation** is assumed for all vertices in the event record.
- **Status codes:** only information on whether a given particle is incoming, outgoing or intermediate will be used.
- **Pointers at a vertex** are assumed to be bi-directional. That is, it is possible to traverse the record structure from mother to daughter and from daughter to mother along the same path.

Extensions/Exceptions to this specifications are handled in some cases.

- Vertices like $\tau^{\pm} \to \tau^{\pm}$ and $\tau^{\mp} \to \tau^{\mp}\gamma$ where 4-momentum conservation is not preserved, but this non-conservation is balanced, for example, between the two branches outgoing from a Z.
- Lines representing intermediate bosons may be missing. In fact this may be unavoidable, if several diagrams contribute simultaneously. In that case, our algorithm makes a choice based on an approximation that only the dominant single diagram is considered and an intermediate boson state is defined accordingly on the fly. Other possible treatments: statistical choice of the dominant process, or calculations based on higher order matrix elements for the hard process, are not available at present.
- As in the `FORTRAN` cases, we expect that new types of conventions for filling the event record will appear, because of physics motivated requirements. Unfortunately, the resulting options do not always guarantee an algebraically closed structure. Host program specific patches may need to be written for `TAUOLA Interface`. Debugging it could be time consuming, and will need to be repeated for every new case.

Detailed conventions for the actual filling of physics information into `HepMC` format is defined by authors of each Monte Carlo program. In future, an important special case of event records filling with information extracted from experimentally observed event (eg. $Z \to \mu^+\mu^-$ modified later to $Z \to \tau^+\tau^-$) should be allowed. Obviously, a new type (or types) of `HepMC` filling will then appear.

3 Design

The structure of our code is documented using Doxygen standards [19] and is presently available from the project web page [20]. The source code for this web page is also available in our package distribution. Doxygen documentation can be thus compiled on a users platform, and hence provide documentation which matches the actual version of the distribution.

Let us present here briefly the directory structure and list the main classes with a short description of their functionality.

3.1 Interface Structure and Responsibilities

The choice of splitting the source code into three main modules, see Fig. 3.1 (blue part), allows separation of the `FORTRAN` related code from the abstract C++ interface and the concrete implementation of the interface created for the appropriate event record.

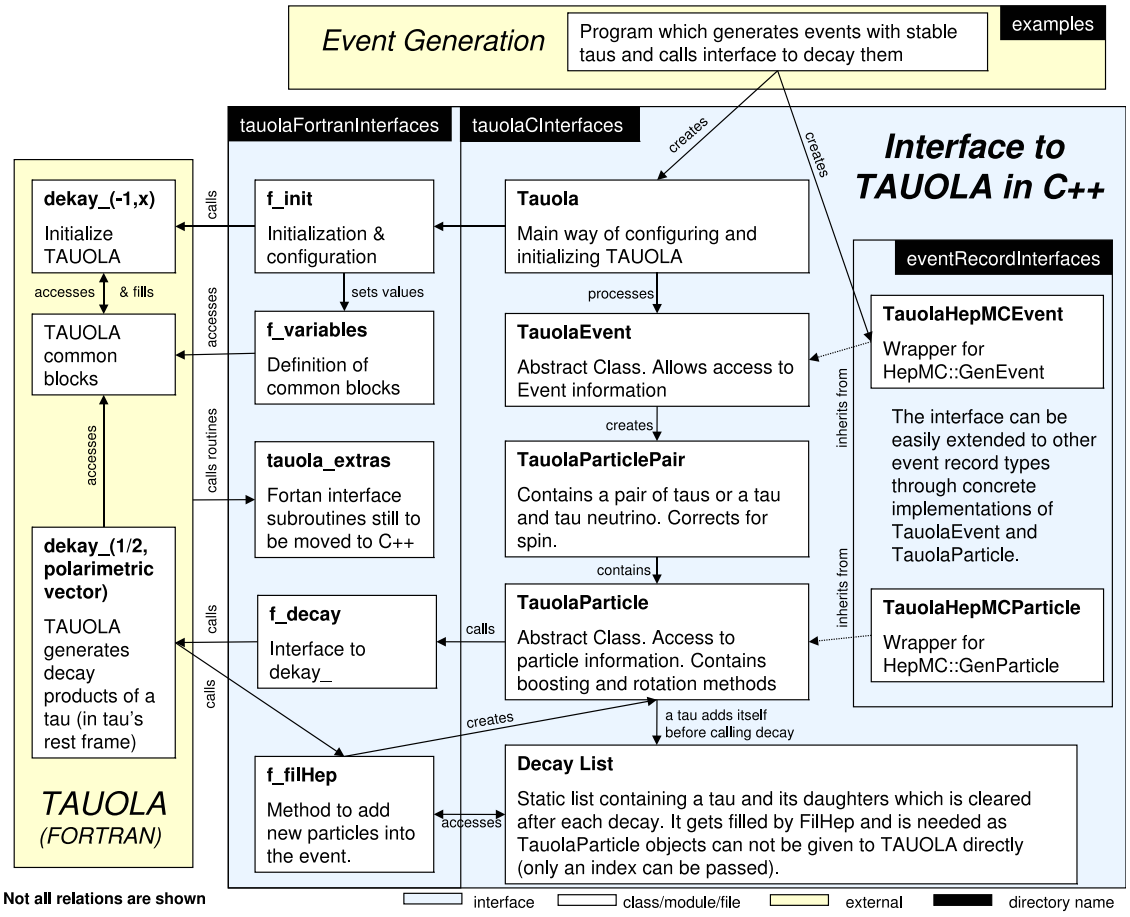


Figure 1: `TAUOLA C++ Interface` class relation diagram

- **Tauola Fortran Interface**
  This part of the code provides an interface to the `FORTRAN` library of `TAUOLA`. In particular, it provides routines necessary for library initialization and a wrapper for the routine which invokes the decay of a single τ. Parts of the interface code are still left in `FORTRAN`, but can be rather easily rewritten to C++. The most important method,

filhep_, is already implemented in C++. Its FORTRAN predecessor writes single particles to the HEPEVT common block. At present the method filhep_ inserts the particle into the HepMC event record but remains to be called from the FORTRAN library. For further details see Appendix A.

- **Tauola C++ Interface**
  The abstract part of the interface to the event record. The class TauolaEvent contains information regarding the whole event structure, while TauolaParticle stores all information regarding a single particle. All particles used by the interface are located in the event in the form of a list of TauolaParticle objects. The last class located here, TauolaParticlePair, is the core of all polarization and decay algorithms. They are independent from the event record used by the interface as they operate on these two abstract classes presented above.

- **Event Record Interface**
  The event record implementation classes. All classes stored here represent the implementation of specific event record interfaces and are responsible for reading, traversing and writing to the event record structure. Only TauolaEvent and TauolaParticle classes must be implemented. The HepMC event record interface is implemented through TauolaHepMCEvent and TauolaHepMCParticle.

3.2 Directory Structure

- **src/eventRecordInterfaces/** - source code for classes which interface with HepMC.
  Classes:
  - **TauolaHepMCEvent** - interface to HepMC::GenEvent objects.
  - **TauolaHepMCParticle** - interface to HepMC::GenParticle objects.

- **src/tauolaCInterfaces/** - source code for general TAUOLA Interface classes, such as those responsible for spin correlations and boosting.
  Classes:
  - **DecayList** - storage class for keeping track of TauolaParticles and their indices.
  - **Tauola** - controls the configuration and initialization of TAUOLA FORTRAN.
  - **TauolaEvent** - abstract base class for event information.
  - **TauolaParticle** - abstract base class for particles in the event. This class also handles particle boosting.
  - **TauolaParticlePair** - contains two objects of type TauolaParticle that are related by the same mother. Spin correlations and other minor algorithms are handled here.

- **src/tauolaFortranInterfaces/** - interface to TAUOLA FORTRAN routines and common blocks.
  Files:
  - **f_Decay** - contains a wrapper for the TAUOLA FORTRAN routine for decaying τ's (DEKAY).
  - **f_FilHep** - provides a method which TAUOLA FORTRAN calls to fill a τ decay product into the event record.
  - **f_Init** - contains a wrapper for the TAUOLA FORTRAN routines for tauola initialization.

  - **f_Variables** - contains definitions of TAUOLA FORTRAN routines and common blocks used by other methods in tauolaFortranInterfaces.
  - **tauola_extras.f** - contains extra FORTRAN routines (taken from the TAUOLA Interface in FORTRAN) which should ultimately be migrated to C++.

- **src/utilities/** - source code for utilities that help in debugging and plotting distributions.
  Classes:
  - **Log** - general purpose logging class that allows filtering out output messages of TAUOLA C++ Interface and keeps statistics regarding a TAUOLA run.
  - **Plot** - a simple class that gathers data for some useful debug plots.

- **examples/** - examples of different TAUOLA C++ Interface uses.
  - **taumain_stand_alone_example** - stand alone example with a simple $e^+e^- \to \tau^+\tau^-$ event in HepMC format and then τ's decayed by TAUOLA.
  - **single_tau_gun_example** - example of TAUOLA linked with PYTHIA 8.1 and used to decay a single τ selected from the event record.
  - **taummk_pythia_example** - example of TAUOLA linked with PYTHIA 8.1. It prints the share of energy for charged hadronic tau decay products from PYTHIA and TAUOLA.
  - **taumain_pythia_example** - example of TAUOLA linked with PYTHIA 8.1, and the decay chain analysed with MC-TESTER.

- **examples/testing/** - Directory containing advanced tests. For details on running and compiling the examples, see Appendix B.4.

- **SANC/** - code for the computation of electroweak corrections.

- **include/** - directory for the header files.

- **lib/** - directory for the compiled libraries.

- **documentation/** - contains doxygen documentation and this latex file.

- **tauola-fortran/** - standard TAUOLA FORTRAN distribution exactly as described in Ref. [8][6]. It is kept intact and is prepared for future updates, see Ref. [9] for details of that project.

3.3 Algorithm Outline

An overview of the algorithm for the TAUOLA Universal Interface is given below, for more detail the reader should refer to the project's Doxygen documentation [20]. Documentation of the TAUOLA FORTRAN Interface [8] describes some aspects of the spin correlation algorithm which are also relevant to this interface.

The first step is creation of a TauolaHepMCEvent object from a HepMC::GenEvent event record. At this step the units for dimensional quantities (four-momenta, masses, etc.) are checked, and if needed the HepMC event record is reset to use GEV and MM ensuring proper

[6] Our interface is prepared for the cleo initialization of FORTRAN TAUOLA. For other cases the configuration routines of the interface need to be adapted.

execution of the τ decay library. After a `TauolaHepMCEvent` is created the `decayTaus()` method should be executed by the user's code[7], invoking the following process:

1. The `HepMC` event record is traversed and a list of all stable τ's in the event is created.
2. From each τ location found, the tree is traversed backwards so that information about the production process can be extracted and used for the calculation of the spin density matrix.
3. The siblings of the τ are identified through common parents, i.e. by requiring that they are produced at the same `HepMC` vertex. In cases such as $\tau \to \gamma\tau$, the parent(s) are defined as the particle(s) which produced the first τ; τ and ν_τ siblings are paired to the τ.
4. The density matrix is set-up using information about the τ-pair and their parent type (for Z/γ processes, grandparent information is also required). This is described in detail in Sec. 4. The density matrix assumes a center-of-mass frame for the τ-pair, with the τ's and their grandparents orientated as shown in Fig. 2(a).
5. The pair is then decayed by executing the `DEKAY` routine for each τ in the pair. The `DEKAY` routine is located in the `tauola-fortran` directory, for details see Appendix A.2.
6. A spin weight is calculated using the polarimetric vectors returned from `TAUOLA FORTRAN` and the density matrix previously set-up (described in Sec. 4).
7. If the decays are rejected, the pair is decayed anew and the process is repeated until the decays are accepted. In this way unweighting of spin effects is performed.
8. Once accepted, the decay products are added into the event record with the procedure as follows:
   (a) As the density matrix is only valid in the special reference frame of Fig. 2(a), the τ-pair are boosted and rotated into this hard process frame.
   (b) The DEKAY routine of `TAUOLA FORTRAN` is executed with state = 11 or 12 (write). This initiates `TAUOLA FORTRAN` to return the daughter information via the `filhep_` routine (see Section A.2).
   (c) The τ's status code is changed from "1" (stable particle) to "2" (intermediate particle).
   (d) A new `HepMC::GenParticle` object is created for each daughter and the appropriate tree structure is created and added into the event.
   (e) Each daughter is boosted using the τ's 4-momentum (as `TAUOLA` constructs a decay for a τ at rest) to the hard process frame.
   (f) The τ's and their decay products are boosted back into the laboratory frame.
9. As the final step, the position of vertices containing the τ's and their decay products is set according to the τ's momentum and lifetime.

The underlying `HepMC::GenEvent` is hence modified with the insertion of τ decay products. All that remains is the conversion of the event back into its initial units, which is done via the `eventEndgame()` routine of the `TauolaHepMCEvent` class. Also in the `eventEndgame()` routine, vertex positions are adjusted according to the τ lifetime.

[7]Prior to this step the user may want to execute `Tauola::decayOne(...)` for τ leptons, for cases where `TAUOLA Universal Interface` is expected not to work properly. For details see Appendix C.9.

4 Calculation of Spin Correlations

If more than one τ lepton is present among final state particles, then not only is the individual spin state for each τ necessary for proper generation, but the complete correlation matrix of all τ leptons must be taken into account as well. In the case of τ-pair production, the standard algorithm explained in [5, 7] can be used without much modification. For the single τ produced in a τ, ν_τ pair, it is convenient to use the same algorithm as well, even though it is not necessary from the physics point of view.

Let us describe now the algorithm given in Refs. [5, 7]. We will use the definitions and notations from these papers as well. Spin correlations and spin polarization effects can be simulated by accepting or rejecting a pair of generated τ decays based on a weighting factor `wt`.

$$wt = \frac{1}{4} \sum_{i,j=0}^{4} h_i^1 h_j^2 R_{ij} \tag{1}$$

where h^1 and h^2 are the polarimetric vectors for the τ^+ and τ^- respectively and R_{ij} is the density matrix associated with the τ production vertex. The matrix R_{ij} depends on the mechanism and particular kinematical configuration of the τ pair production. h_i^1 and h_j^2 depend on the respective decays of τ^+ and τ^-. The approach can be used for $\tau - \nu_\tau$ production as well. In this case a ν_τ decay is not performed and its polarimetric vector is set to $h = (2, 0, 0, 0)$.

A pair of τ decays should be accepted if the weight is greater than a randomly generated number between 0 and 1. If this condition fails, the τ pair decays should either be rejected and regenerated, or rotated[8], and the weight recalculated. The production process does not need to be reprocessed.

The density matrices, R_{ij}, for the most standard processes of τ-pair production, are documented below. The following frame convention[9] was adopted:

- The τ-pair's center of mass system is used
- The τ^+ (if present) lies along the positive z axis
- The τ^- (if present) lies along the negative z axis
- The incoming beams (if present) lie in the z-y plane.
- If applicable, the incoming antiquark (or antilepton) y momentum component is positive.

h is defined such that h_0=1 and $h_{1,2,3}$ form the polarimetric vertex returned from `TAUOLA FORTRAN` (see DEKAY in Appendix A.2). h is defined in the rest frame of the τ it was calculated for. One should stress that formally speaking, R_{ij} does not represent a Lorentz invariant object. Its first index is defined in the rest frame of the τ^+, whereas the second index is in the frame of the τ^-.

[8] Rotation instead of rejection increases efficiency by a factor of four. This however only affects the generation of τ lepton decays and represents a small fraction of the total time of constructing the event.

[9] Fig. 2(a) illustrates our choice too. There however the reaction frame is rotated by an angle θ around the x axis.

In the following subsections we will list the form of R_{ij} for the most commonly used processes of τ-pair (or τ, ν) production.

4.1 Form of R_{ij} for Standard Processes

4.1.1 $Z/\gamma \to \tau^+\tau^-$

$$R = \begin{pmatrix} 1 & 0 & 0 & 2P_z(cos\theta) - 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 2P_z(cos\theta) - 1 & 0 & 0 & 1 \end{pmatrix}$$

where P_z is calculated from the square of the matrix elements of the Born-level $2 \to 2$ process $f\bar{f} \to \tau^+\tau^-$.

$$P_z(s,\theta) = \frac{\frac{d\sigma(s,\theta,+,+)}{d\Omega}}{\frac{d\sigma(s,\theta,+,+)}{d\Omega} + \frac{d\sigma(s,\theta,-,-)}{d\Omega}}$$

θ is the angle between the incoming antiparticle beam and outgoing τ^+. If the incoming beam cannot be reconstructed from the event record the average of P_z should be used (which is equal to $P_z(cos\theta = 0)$). The "+" denotes that the spin states of τ^+ and τ^- are parallel to the τ^+ momentum. For "−" it is placed in the opposite direction.

The spin correlation matrix explained above is approximate and is valid only for longitudinal spin effects. The object, $R_{i,j}$, is nonetheless prepared to host complete spin effects. For that purpose the information available from the module based on `SANC` can be used, see Appendix D. A further advantage to using this module is that genuine weak effects can be calculated and included as a weight[10], not only on polarization, but on the cross section as well. This effect was found to be numerically important [21, 22] for final states of virtuality largely surpassing the Z mass and should be taken into account prior to the implementation of new physics effects.

In the formulas above, the hard process kinematical variables s and θ have to be known for each event. These variables, together with the flavour of the incoming beam are used for calculating electroweak corrections or the function P_z.[11]

To apply the method we need to identify the four momenta of the τ^+ and τ^- pair first. In the rest frame of the pair the two effective partons leading to the hard process are not necessarily back to back. Two scattering angles θ_1 and θ_2 can be thus reconstructed. The angle θ_1 is between the τ^+ and the first incoming state, θ_2 is between τ^- and the second one[12]. Both angles are calculated in the rest frame of the τ pair. The average angle $\theta^\bullet$, according to the description given in [25], is taken: $\cos\theta^\bullet = \frac{\sin\theta_1\cos\theta_2 + \sin\theta_2\cos\theta_1}{\sin\theta_1 + \sin\theta_2}$.

[10] In a similar way one can implement effects of new physics, such as Z' into the program. With the help of our interface effects of weak corrections on the cross section, and not only on polarization can be installed with additional weights.

[11] The principle behind our solution is quite similar to the one used in `PHOTOS` Monte Carlo where it is was shown [23] to be valid up to NLO (QED) precision level. It relies on the factorization properties of fully differential distributions into the appropriately chosen Born level ones and emission factors. To achieve such precision in the case of spin correlations in proton-proton collisions, rather challenging work on QCD matrix elements would be necessary. At present, our predictions will not be better than LL *on spin effects*. It is known [24], that the solution can not be constructed beyond NLO.

[12] We choose the first incoming state to be antiparticle.

If events originate from a generator such as PYTHIA, the flavour of incoming partons is explicitly given or it can be calculated using information encoded in the event record. In the generic case, this information is not available, and one will have to rely on measured structure functions and statistical choice. This is the method, for example, to be applied for embedding techniques in which muons are replaced by Monte Carlo generated taus in $Z \to \mu^+\mu^-$ (or $W \to \mu\nu_\mu$) decays present in experimental data. Known physics τ lepton background can be thus estimated in this way. The resulting uncertainty will be free from the initial state QCD contribution. Also angular polarization dependence due to different d and u quark couplings to the Z is not that large. The mismatch between choosing a quark and antiquark may have a larger effect.

The density matrix presented above features only longitudinal spin correlations. Once the matrix R is replaced with the one featuring complete spin effects, see Appendix D, the transverse spin effects are taken into account as well.

4.1.2 $H^0 \to \tau^+\tau^-$ and $A^0 \to \tau^+\tau^-$ and mixed $A^0/H^0 \to \tau^+\tau^-$

The complete density matrix for a scalar neutral Higgs boson H^0 is rather simple,

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

it is also true for a pseudoscalar neutral Higgs boson A^0

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

A mixed $A^0/H^0 \to \tau^+\tau^-$ represents only a slightly more complicated case:

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{(\beta cos\phi)^2 - sin\phi^2}{(\beta cos\phi)^2 + sin\phi^2} & -\frac{2\beta cos\phi sin\phi}{(\beta cos\phi)^2 + sin\phi^2} & 0 \\ 0 & \frac{2\beta cos\phi sin\phi}{(\beta cos\phi)^2 + sin\phi^2} & \frac{(\beta cos\phi)^2 - sin\phi^2}{(\beta cos\phi)^2 + sin\phi^2} & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

where $\beta = \sqrt{1 - (\frac{2M_\tau}{M_{A^0/H^0}})^2}$ and ϕ is the scalar-pseudoscalar mixing angle.

4.1.3 $W^\pm \to \tau^\pm\nu$

For W the matrix R_{ij} takes the following form:

$$R = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

4.1.4 $H^{\pm} \to \tau^{\pm}\nu$

For charged Higgs decay the matrix R_{ij} differs from the W case by signs only:

$$R = \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}$$

4.2 Cases of Partly defined Hard Processes

4.2.1 $\tau^+\tau^-$ Pair with Multiple Parents and Sisters

If a $\tau^+\tau^-$ pair is found with multiple parents rather than a single parent, the parent type is assumed to be a Z/γ with the 4-momentum reconstructed from the 4-momentum of the τ pair. The density matrix 4.1.1 is used. For more details on the construction of effective incoming beams see Section 4.4.

4.2.2 $\tau\nu_\tau$ Pair with Multiple Parents and Sisters

If a $\tau^+\nu$ pair is found with multiple parents rather than a single parent, the parent type is assumed to be a $W^{\pm}$ with the 4-momentum reconstructed from the 4-momentum of the τ ν_τ pair. The density matrix 4.1.3 is used.

4.2.3 Single τ and Multiple Unpaired τ's

By default a single τ, with no τ or ν_τ pairing, is treated as unpolarized and spin effects are ignored. However, in such cases the `TAUOLA decayOne` method, see Appendix C.9, can be used for the inclusion of spin effects[13]. This method can be applied by the user to impose a spin state on a single τ. The method can even be used for multiple final state τs and exact spin correlations. In these cases a user defined quantization frame for each τ may be necessary. For each τ a distinct routine for boosting from its rest-frame to the lab-frame may be necessary. The appropriate method is explained in the Appendix mentioned above.

Another user defined option (which may become a default part of `TAUOLA Interface` in the future) is when pairing can not be done on the basis of inspecting τ mother(s) but can be performed according to the closeness of the reconstructed invariant masses of the pairs to the masses of W's or Z's (if the appropriate Standard Model processes are under considerations). In general, as such configurations will often appear for processes of new physics, again a user defined and hard process dependent solutions based on the `TAUOLA decayOne` method, might be the only option.

[13] In principle this method can use the polarization vector stored in the `HepMC` event record in a straightforward manner. The proper orientation of the quantization frame must be nonetheless controlled by the user. In particular the polarization vector must be given in the τ rest frame.

4.3 Quantum Entanglement and Helicity States

One of the convenient feature of Monte Carlo simulation is the availability of variables used in the generation of hard processes. However, it is often not possible to define in an exact manner what is the energy transfer, eg. in the Z propagator, if several diagrams contribute simultaneously. Nonetheless use of such information is tempting to validate the algorithms used in experimental analysis to define observables.

Another example of such useful variables are the helicities of τ^+ and τ^-. Even though it is possible to attribute such variables only in the ultrarelativistic limit and its use can be restricted to a downgraded physics approximation only (quantum entanglement [26] ignored), helicity was offering invaluable help at the time of LEP, for the measurement of τ polarization [27].

Our program provides the helicity states of τ^+ and τ^- even in cases when exact spin effects are taken into account in the generation, as is the case of processes mediated by an intermediate Z/γ^* state or in a Higgs boson decay. We just attribute helicity states for the τ lepton after decays are already generated and accepted. The approximation used in calculation of these helicities is explained in Section 4.5 and technical aspects of our solution are given in Appendix C.7. This information can be used for solutions similar to the ones in [27], but this time for LHC purposes.

4.4 Handling of Events with Bremsstrahlung or Parton Shower Activity

Obviously, there are cases, when spin correlations calculated from Born level processes can not be applied directly. Good examples are: (a) $Z \to \tau^+\tau^-\gamma$, (b) $f1 + f2 \to Z + X$, $Z \to \tau^+\tau^-$ where the intermediate state Z is explicitly stored, (c) $f1 + f2 \to \tau^+\tau^- X$ or $f1 + f2 \to \tau^+\nu_\tau X$ where the Z state is not available. Here X represents a parton shower and/or final state bremsstrahlung. The first step, necessary eg. for calculation of the spin correlation matrix, is to reconstruct the effective Born level variables s and θ and the incoming state flavours; the arguments of the function P_Z. This is equivalent to the construction of effective incoming and outgoing τ fields. Let us discuss now each case:

a) An additional photon is added to the τ with which it forms the smaller virtuality. For construction of the transformation between the laboratory frame and rest frame of the τ's, the effective state, $Z - \gamma$, frame is used instead of the intermediate Z frame. This choice is motivated by inspection of the properties of the spin amplitudes. In such a frame the effective incoming states, $f1$ and $f2$, will not necessarily be back to back. The average of the two directions (θ^*, $\theta^\bullet$, see Ref. [25]) can be used. The virtuality of the Z is nonetheless used in the effective Born calculation.

b) Additional fields, X, representing parton showers should be subtracted from $f1$ or $f2$, preferably from the one with which it forms the smaller virtuality. This is motivated by inspection of the spin amplitudes. Once the effective incoming states are constructed, the definition of boosting routines is straightforward.

c) This case is a combination of the two above. Additional fields should be subtracted from $f1$, $f2$ or considered as originating from the intermediate Z (which is reconstructed on the fly) together with τ^+ and τ^-. The minimalization of virtuality should be used as a guide whenever a combinatorial choice has to be made. However electromagnetic charge

or colour charge should not be neglected. Obviously photons should not be combined with neutrinos nor gluons with leptons.

The approach presented above is explained in Ref. [8] in more detail. The principle is based on the simplest factorization properties of SM/QCD matrix elements. In particular the assumption is made that photon(s) can be treated as (nearly) collinear with one of the final state τ leptons. Then the kinematics can be built on the $\tau^+\tau^-$ pair rest frame and the effective incoming states. The z axis is taken along the effective antiparticle incoming state direction and the y axis is of the half plane including the τ^-. The second effective beam is then placed in the zy plane as well, even though it is not back to back with the first one. For calculation of R_{ij}, the variable s should be calculated from the effective incoming states. The scattering angle can be calculated in the Z frame using the formula for θ^* or $\theta^\bullet$, see Ref. [25]. In the collinear limit for photon emissions, Lorentz transformation between the τ^+, τ^- and Z rest frames is reduced to a simple boost along the $\tau^+ - \tau^-$ flight directions in the Z rest-frame.

Further improvements with respect to that description require explicit use of higher order matrix elements. The approximation described is already quite good and works up to $\alpha_{QED}/\pi \simeq$ 0.1 - 0.2 % precision level, for observables where it is not requested explicitly that high p_T photons are present.

The situation with initial state parton shower emissions is similar, however in this case the omitted terms for the R calculation may be of the order of α_{QCD}/π. Thus significantly larger, but still at the level of ten percent or so.

If the event record under study originates from experimental data, the flavours of the incoming partons can not be known from the event record itself. Instead, information in PDFs can be used to attribute such flavours on a statistical basis and then used in the calculation of the matrix R_{ij}.

4.5 Exact Spin Effects and Helicity States

Earlier in this section we have listed some examples of spin polarization, spin correlation and density matrices. Those matrices can be easily changed/replaced by the externally calculated ones for the sake of studying new physics phenomena, for example to study the consequence of certain types of spin correlations on signal/background separation. In this way, the effects of new physics or ad hoc modification of spin effects, component by component, can be easily included.

For calculating complete spin correlations, including the effects of genuine weak corrections in the case of the single boson mechanism of τ-pair production, another solution is also prepared, see Section 5. It works in the case of an intermediate γ^*/Z state produced in the annihilation of a pair of quarks. This can serve as an example for other processes which can be implemented in a similar manner.

Let us stress that in the case of including complete spin effects it is not possible to attribute helicity states to the produced τ leptons. This can be done only in an approximate way. For that purpose, a modified version of formula (1) can be used. The weight for each helicity configuration is:

$$weight(\pm\,,\pm\,) = \frac{1}{4}\sum_{i,j,i'j'=0}^{4} h^1_{i'}h^2_{j'}R_{ij}P^1_{ii'}P^2_{jj'} \qquad (2)$$

The matrices $P^{1,2}$ (spin projection operators) read

$$P^{1,2} = \begin{pmatrix} 1 & 0 & 0 & \pm 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \pm 1 & 0 & 0 & 1 \end{pmatrix}$$

As a consequence for `Hel1`=± 1 and `Hel2`=± 1

$$\begin{pmatrix} weight(+,+) = & (h_0^1 + h_3^1) & (h_0^2 + h_3^2) & (R_{00} + R_{03} + R_{30} + R_{33}) \\ weight(+,-) = & (h_0^1 + h_3^1) & (h_0^2 - h_3^2) & (R_{00} - R_{03} + R_{30} - R_{33}) \\ weight(-,+) = & (h_0^1 - h_3^1) & (h_0^2 + h_3^2) & (R_{00} + R_{03} - R_{30} - R_{33}) \\ weight(-,-) = & (h_0^1 - h_3^1) & (h_0^2 - h_3^2) & (R_{00} - R_{03} - R_{30} + R_{33}) \end{pmatrix}$$

The actual helicities can then be attributed by unweighting the above helicity weight.

It is no omission that we are not explicit on sign conventions for the generated helicity variables `Hel1` and `Hel2`. This depend on the particular choice of boosting routine. Our choice adopted here is, that for the τ pair produced at the Z peak on average both `Hel1` and `Hel2` will be negative and `Hel1 = Hel2`.

5 Electroweak Corrections and Refined Spin Effects

5.1 External Calculation of the Spin Density Matrix R_{ij}

Our program is equipped with methods for calculating simple spin density matrices for most of the interesting hard processes. These methods are explained in the text of the paper, see Section 4. In some cases, notably in the case of $\tau^+\tau^-$ produced from the annihilation of a pair of quarks, the standard density matrices may not be sufficient for some applications. A more exact solution is also available. Instead of a native R_{ij} density matrix, an externally calculated one can be used.

The solution is based on the `SANC` library [28, 29] for calculation of electroweak corrections[14]. With its help the density matrix R_{ij} for the $q\bar{q} \to \tau^+\tau^-$ process can be calculated as a function of the incoming state flavour and Born level variables (Mandelstam s and scattering angle θ). An additional two weights are also provided, which include the matrix elements squared and averaged over the spin. For additional weights, genuine weak corrections are respectively switched on and off. This may be helpful for the evaluation of genuine weak corrections for states of large s, significantly above the Z peak, where they become sizable. See eg. Refs. [21, 22].

For better modularity of the interface and to speed up execution of the program, pretabulation is used. At first, a dedicated module has to be invoked, as will be explained later. In such dedicated runs, R_{ij} is calculated and stored in a lattice of (s and $\cos\theta$) points.

Later, in the actual execution of our interface, these pretabulated values of R_{ij} are interpolated to the actual phase space point. For this purpose, the standard bilinear interpolation

[14] It may serve as an example of how other calculations featuring heavy Z' boson, for example, may be used in our interface.

algorithm is used. Additionally, in order to avoid numerical errors, for $\cos\theta$ values near -1 and 1 we use the linear extrapolation algorithm.

Pretabulation is prepared for three domains of s: around the Z peak, close to the WW pair production threshold and over a broad energy range. The specific choice for pretabulation zones is 85 GeV$< \sqrt{s} <$ 110 GeV, 160 GeV$< \sqrt{s} <$ 220 GeV[15] and 6 GeV$< \sqrt{s} <$ 17 TeV. For s below 36 GeV^2 the analytic form taken from Ref. [30] is used. It features all spin and mass effects, but electroweak corrections, and even Z exchange, are not taken into account. This is reasonable for $s < 36$ GeV^2 (up to, say, 100 GeV^2).

The advantage of this solution is that results of the `SANC` library calculation can be modified by the user before it is loaded into our interface without altering the code of the interface itself.

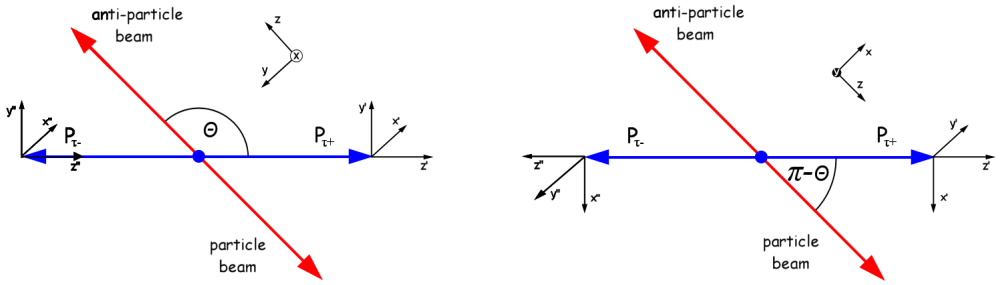
5.2 Conventions of Frames: KORALB, SANC and TAUOLA Interface

It is not essential for `TAUOLA Interface` and the component which calculates electroweak corrections to follow exactly the same conventions for spin quantization. In `TAUOLA Interface` we follow the frame orientation exactly as in paper [31]. The adopted frame orientation is shown in Fig. 2(a). In `SANC`, the orientation of axes is different, see Fig. 2(b). In the case of our interface, the beam momenta are laid along the z axis. The antiparticle beam is parallel, particle beam is antiparallel. The y component of the τ^- is always positive. The θ angle to be used for calculation of the density matrix is between the directions of the antiparticle beam and the τ^+. In the case of the `SANC` module, the τ momenta, p_{τ^+} and p_{τ^-}, lie in the xz plane. The xz plane is the reaction plane: a beam of particles (quarks or leptons) is parallel to the z axis. The x component of p_{τ^-} is always negative. The $y^{'}$ and $y^{''}$ axes of the τ^+ and τ^- spin frames correspondingly have opposite direction to each other. The $y^{''}$ axis is parallel to the y axis of the hard process frame. Appropriate rotations and other convention adjustments are performed by the program in preparation of the R_{ij} tables: `SANCtable.cxx`.

5.3 Numerical Significance of Electroweak Corrections

One may wonder whether the numerical results induced by electroweak corrections are of any practical purpose. They are expected to be of the order of 1% and indeed are not even that large for an intermediate state virtuality of up to 100 GeV above the Z boson mass. The situation changes significantly, however, at higher energies. As can be seen from Figures 3 and 4 the effect may be of the order of even 50% at virtualities of several TeV. This is quite in agreement with the results of Refs. [22, 21]. In Figures 5(a) and 5(b) we collect results for τ polarization calculated at $\cos\theta = -0.2$. Again, the effects are small up to the energy scale of about 500 GeV. At larger scales, corrections become sizable. The electroweak corrections should therefore be considered in studies aiming for new physics phenomena such as $Z' \to \tau^+\tau^-$ decays.

[15] In the case when the application is to Z', this pretabulation zone should be replaced, for example by $M_{Z'} - 3\Gamma_{Z'} < \sqrt{s} < M_{Z'} + 3\Gamma_{Z'}$.



(a) KORALB-like orientation

(b) SANC module orientation

Figure 2: The relative orientation of reference frames for the spin states of τ^+, τ^- and for hard processes as used in our interface (Fig. (a)) and in the SANC module (Fig. (b)) are shown. In Fig (a) the axes x (not shown explicitly), x' and x'' are parallel to each other and point behind the picture, axis z is parallel to the direction of the anti-particle beam. In Fig (b) the axis z is parallel to the direction of the particle beam, the axis y' points behind the picture. Axes y (not shown explicitly) and y'' point toward the reader and are antiparallel to y'.

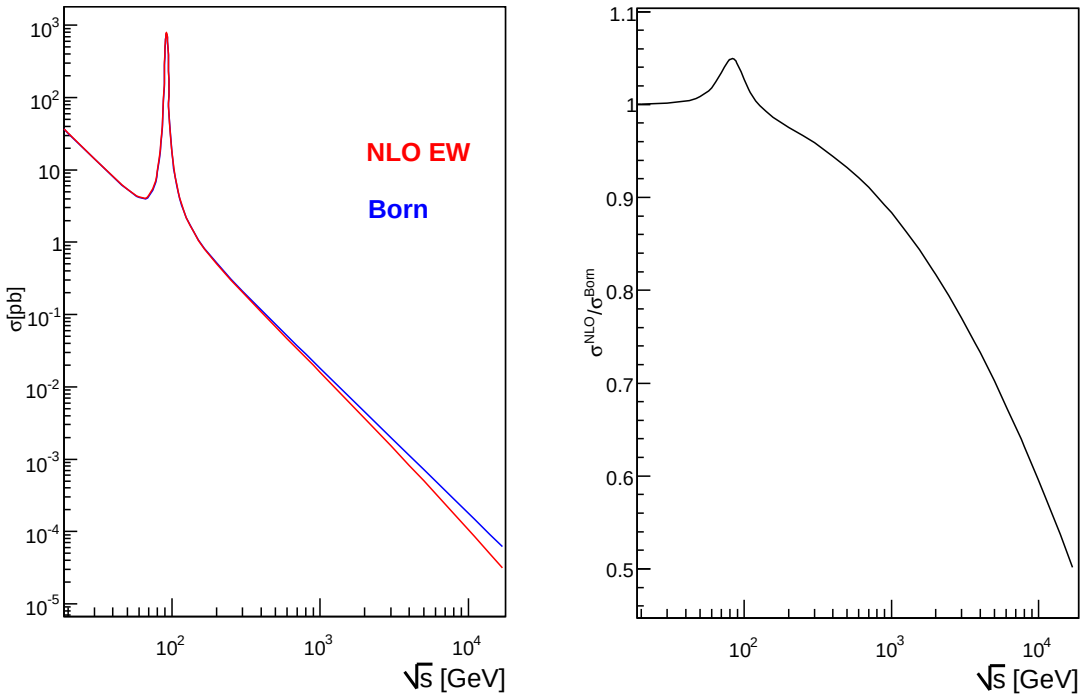


Figure 3: The integrated cross section of τ pair production from up quarks calculated with and without NLO EW corrections (red and blue lines) is shown in the left hand side plot. The ratio of the two distributions is given on the right hand plot. We use the alpha scheme for electroweak corrections. This is why light fermion loops contribute to the difference between the two lines. The differences between the alpha scheme Born predictions and expressions used in the host program must be understood before the correcting weight (see Appendix C.7) is used.

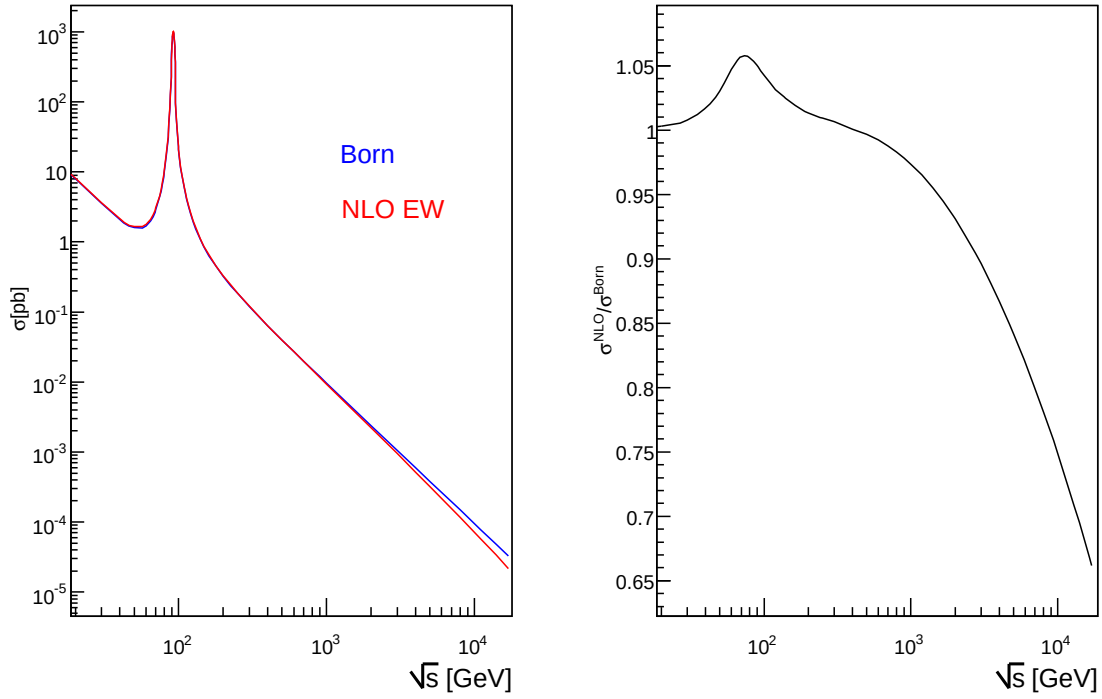


Figure 4: The integrated cross section of τ pair production from down quarks calculated with and without NLO EW corrections (red and blue lines) is shown in the left hand side plot. The ratio of the two distributions is given on the right hand plot. We use the alpha scheme for electroweak corrections. This is why light fermion loops contribute to the difference between the two lines. The differences between the alpha scheme Born predictions and expressions used in host program must be understood before the correcting weight (see Appendix C.7) is used.

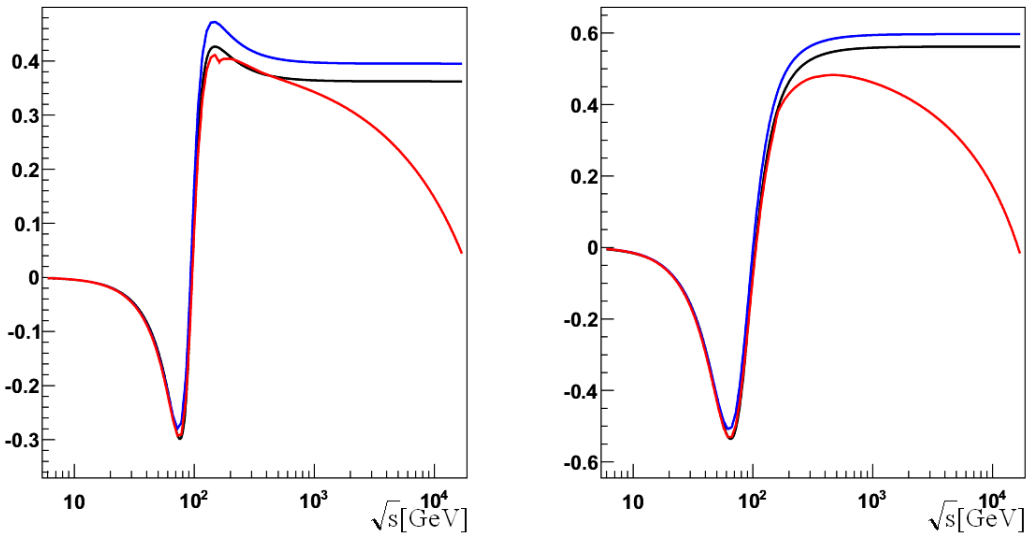
6 Tests of Spin Correlations and Numerical Results

There are two purposes of the results presented in this section. On one hand these results complement the technical tests described in Appendix B.3 with the ones oriented toward a particular hard processes. The technical tests should be repeated for every new program installation or configuration. On the other hand, results of the present section are of potential physics interest as well. They illustrate the dominant spin effects on idealized distributions for Z, W and H decays. Distributions are similar for the e^+e^- and LHC measurements.

Tests presented here were conducted using `MC-TESTER` [15, 16]. `MC-TESTER` allows semi-automated comparisons of invariant mass distributions of each sub-group of eg. τ or Z stable decay products. The results of these tests were also compared to the results obtained with the `FORTRAN` interface (which has been well validated by comparison with analytical and numerical calculations for τ pair production[16]).

In addition to this, we created custom `MC-TESTER` macros for plotting other spin sensitive quantities and compared these to published results. Numerical results are presented later in the section, see Figs. 8(a), 8(b) and 9(a), 9(b).

[16] This represents tests of the interface. In all cases τ decays are generated with the help of `TAUOLA FORTRAN`. For a review of physics oriented tests of τ decays themselves, and projects for future improvements based on low energy e^+e^- data, see Ref. [9].

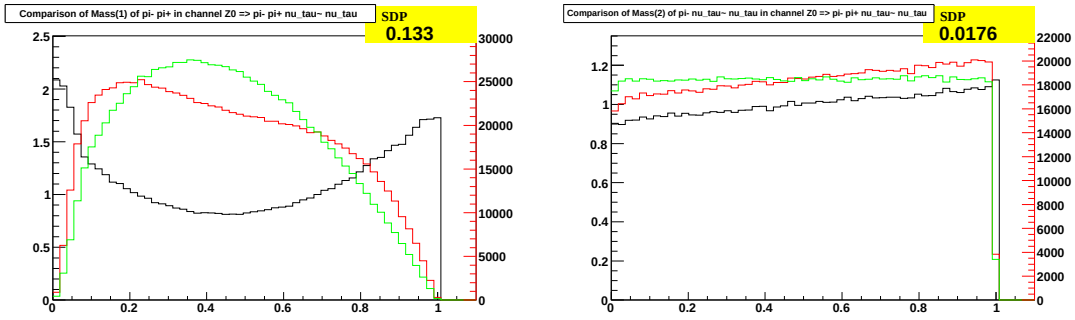


(a) Up quarks (b) Down quarks

Figure 5: Polarization for τ leptons produced from up quarks (Fig. (a)) and down quarks (Fig. (b)) at $\cos\theta = -0.2$. The red line is with electroweak corrections, the black is Standard Born as is default in the interface. The blue line is Born according to the alpha scheme. The main purpose of these results is a technical test of the software installation. Note however the inadequateness of the alpha scheme Born, which is significantly different from the other two results even at relatively low energies. The small bump on the red line on Fig. (a) is due to the WW threshold. It is insignificant for positive $\cos\theta$.

6.1 $Z/\gamma \to \tau^+\tau^-$

The longitudinal spin effects for a Z decay into τ's was tested by restricting the τ decay mode to $\tau^\pm \to \pi^\pm\nu_\tau$ and examining the invariant mass of the $\pi^+\pi^-$ pair, $M_{\pi^+\pi^-}$ (see Fig. 6(a)) and the π energy distribution in the rest frame of the Z (see Fig. 6(b)). The effect of Z polarization on these distributions was studied in [32] and we obtained consistent results with the new C++ implementation of `TAUOLA Interface`.

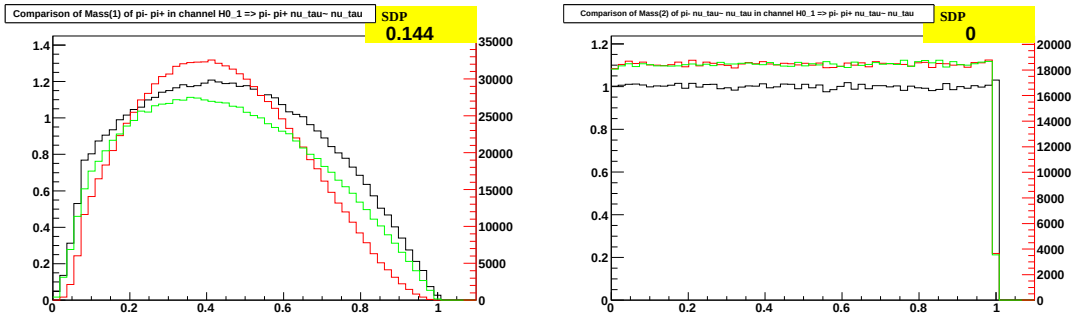


(a) $M_{\pi^+\pi^-}$ (b) $1 - 2\frac{E_{\pi^+}}{M_Z}$

Figure 6: Longitudinal spin observables for the Z boson ($e^+e^- \to Z$ at 500 GeV). Distributions are shown for spin effects switched on (red), spin effects switched off (green), and their ratio (black).

6.2 $H^0/A^0 \to \tau^+\tau^-$

As was done for the Z decay in Section 6.1, longitudinal spin effects for the Higgs decay into τ's was tested using $M_{\pi^+\pi^-}$ (Fig. 7(a)) and the π energy distribution in the rest frame of the H^0 (see Fig. 7(b)), which was flat as expected.



(a) $M_{\pi^+\pi^-}$ (b) $1 - 2\frac{E_{\pi^+}}{M_Z}$

Figure 7: Longitudinal spin observables for the H boson for $\tau^\pm \to \pi^\pm\nu_\tau$. Distributions are shown for spin effects switched on (red), spin effects switched off (green), and their ratio (black).

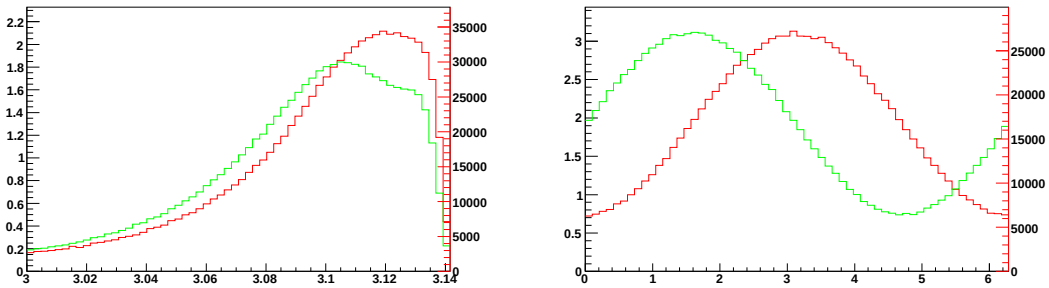
Let us now turn to transverse spin correlations. In Fig. 8(a) the benchmark histogram as produced by our `FORTRAN Interface` and given in Fig. 3 of Ref. [33] is reproduced[17]. It features acollinearity of the π^+, π^- pair in the Higgs boson rest frame, both τ's decay to $\pi^\pm\nu$. For the same decay set up, Fig. 8(b) features the acoplanarity of the planes built respectively from the directions of the decay products of τ^+ and τ^-. The spin effect is indeed large. However, it requires use of unobservable neutrino momenta. It is difficult or even impossible to achieve sufficient experimental precision in reconstruction of the reaction frame necessary for this observable. The first observable, presented in Fig. 8(a), also suffers from the same limitation.

The two other tests, Figures 9(a) and 9(b) present the distribution of the acoplanarity angle for the two planes built respectively from the momenta of $\pi^+\pi^0$ and $\pi^-\pi^0$; the decay products of ρ^+ and ρ^-. All are in the rest frame of the ρ-pair. It is directly based on measurable quantities. The $\rho^\pm$ originate respectively from $\tau^\pm \to \nu\rho^\pm$ decays. There is no need for Higgs rest frame reconstruction in this case. Events are divided into two categories. If the energy difference between charged and neutral pions coming from the two τ's are of the same sign, they contribute to Fig. 9(a), otherwise they contribute to Fig. 9(b). For details of the definition and for more numerical results obtained with the `TAUOLA FORTRAN Interface`, see [34].

6.3 $W^\pm \to \tau^\pm\nu_\tau$ and $H^\pm \to \tau^\pm\nu_\tau$

For the simplest decay mode, $\tau^\pm \to \pi^\pm\nu_\tau$, as was already discussed in Ref. [32], the pion energy spectrum should be softer in the case of $W^\pm$ decays and harder in the case of charged Higgs decay. This is indeed reproduced in Figs. 10(a) and 10(b) and the spectra are reversed for the two cases.

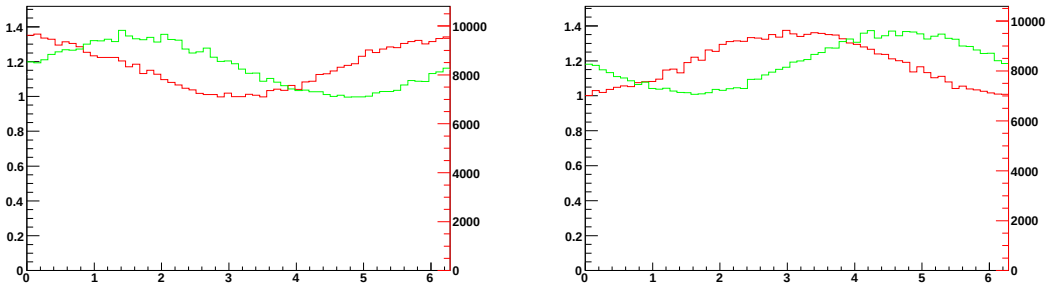
[17] In this plot the case of non zero scalar-pseudoscalar mixing was chosen. This is the origin of the difference with Ref. [33].



(a) $\pi^+\pi^-$ acollinearity distribution ($\approx \pi$)

(b) $\pi^+\pi^-$ acoplanarity distribution

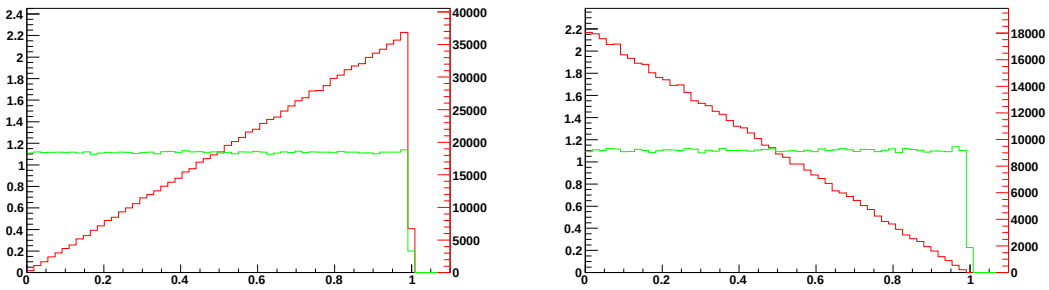
Figure 8: Transverse spin observables for the Higgs boson for $\tau^\pm \to \pi^\pm \nu_\tau$. Distributions are shown for scalar Higgs (red), scalar-pseudoscalar Higgs with mixing angle $\frac{\pi}{4}$. For the definition of angles see Section 6.2.



(a) acoplanarity distribution ($y_1 y_2 > 0$)

(b) acoplanarity distribution ($y_1 y_2 < 0$)

Figure 9: Transverse spin observables for the Higgs boson for $\tau^\pm \to \pi^\pm \pi^0 \nu_\tau$. Distributions are shown for scalar Higgs (red), scalar-pseudoscalar Higgs with mixing angle $\frac{\pi}{4}$ (green). For the definition of angles see Section 6.2.



(a) $W^+ \to \nu_\tau \tau^+,\ \tau^+ \to \pi^+ \bar{\nu}_\tau$

(b) $H^+ \to \nu_\tau \tau^+,\ \tau^+ \to \pi^+ \bar{\nu}_\tau$

Figure 10: Pion energy spectrum in the rest frame of W (left hand side) and H^+ (right-hand side). Spin effects included (red line) and neglected (green line) are plotted. The variable $1 - 2\frac{E_{\pi^+}}{M_{W^+}}$ or $1 - 2\frac{E_{\pi^+}}{M_{H^+}}$ is used respectively.

7 Outlook

Let us summarise briefly the next steps which are planned for the work on `TAUOLA Universal Interface`.

- Further extension of our work will focus on studies to recover spin states from quantities which can be measured experimentally. A discussion of theoretical systematic error for such reconstructed spin states will require discussion of QCD corrections.

- At present, our interface is designed to work with the `HepMC` event record. However, if a new event record structure became popular it would be rather easy to adapt to, thanks to the design we tested in `MC-TESTER` [15, 16].

  `TAUOLA Universal Interface` as well as `TAUOLA` itself are expected to remain framework-like code where the user is supposed to modify some of the parts according to her/his particular purposes.

- The segment of code for analyzing the hard process and generating spin states is now becoming a significant component of the project and already exceeds by far the category of peripheric methods related to `TAUOLA Interface`. In the future it should be moved to a separate class.

- We expect that in the next few years better parametrization of hadronic form-factors based on τ data from the Belle and BaBar collaborations and refined models of decays will become available. Then the directory `/tauola-fortran` will be replaced by a new version incorporating those achievements.

Acknowledgments

Useful discussions with P. Golonka during the early stage of project development and discussions with members of the ATLAS and CMS collaborations, and the LCG group are acknowledged. We are especially indebted to the pilot users of the interface, in particular to: Julia Yarba, Sami Lehti, Eric Torrence, Sho Iwamoto, Marcin Wolter and Anna Kaczmarska.

Partial support from Polish-French collaboration no. 06-124 within IN2P3 through LAPP Annecy during the final completion of this work is also acknowledged.

References

[1] Two Fermion Working Group Collaboration, M. Kobel *et al.*, `hep-ph/0007180`.

[2] E. Richter-Was, T. Szymocha, and Z. Was, *Phys. Lett.* **B589** (2004) 125–134, `hep-ph/0402159`.

[3] The ATLAS Collaboration, G. Aad *et al.*, `0901.0512`.

[4] DELPHI Collaboration, J. Abdallah *et al.*, *Eur. Phys. J.* **C47** (2006) 273–294, `hep-ex/0604038`.

[5] S. Jadach, J. H. Kühn, and Z. Wąs, *Comput. Phys. Commun.* **64** (1990) 275.

[6] M. Jeżabek, Z. Wąs, S. Jadach, and J. H. Kühn, *Comput. Phys. Commun.* **70** (1992) 69.

[7] S. Jadach, Z. Wąs, R. Decker, and J. H. Kühn, *Comput. Phys. Commun.* **76** (1993) 361.

[8] P. Golonka *et al.*, *Comput. Phys. Commun.* **174** (2006) 818–835, `hep-ph/0312240`.

[9] S. Actis *et al.*, *Eur. Phys. J.* **C66** (2010) 585–686, 0912.0749.

[10] M. Dobbs and J. B. Hansen, *Comput. Phys. Commun.* **134** (2001) 41–46, https://savannah.cern.ch/projects/hepmc/.

[11] E. Barberio, B. van Eijk, and Z. Wąs, *Comput. Phys. Commun.* **66** (1991) 115.

[12] E. Barberio and Z. Wąs, *Comput. Phys. Commun.* **79** (1994) 291–308.

[13] N. Davidson, T. Przedzinski, and Z. Was, `PHOTOS C++ Interface` source code and documentation aviable from http://wasm.web.cern.ch/wasm/C++.html or http://www.ph.unimelb.edu.au/~ndavidson/photos/doxygen/index.html.

[14] N. Davidson, T. Przedzinski, and Z. Was, 1011.0937.

[15] P. Golonka, T. Pierzchala, and Z. Was, *Comput. Phys. Commun.* **157** (2004) 39–62, hep-ph/0210252.

[16] N. Davidson, P. Golonka, T. Przedzinski, and Z. Was, *Comput. Phys. Commun.* **182** (2011) 779–789, 0812.3215.

[17] T. Sjostrand, S. Mrenna, and P. Skands, *Comput. Phys. Commun.* **178** (2008) 852–867, 0710.3820.

[18] M. Bahr *et al.*, *Eur. Phys. J.* **C58** (2008) 639–707, 0803.0883.

[19] D. van Heesch, www.doxygen.org.

[20] N. Davidson, G. Nanava, T. Przedzinski, E. Richter-Was, and Z. Was, `TAUOLA` and `TAUOLA C++ Interface` source code and documentation aviable from http://wasm.web.cern.ch/wasm/C++.html or http://www.ph.unimelb.edu.au/~ndavidson/tauola/doxygen/index.html.

[21] N. E. Adam, V. Halyo, S. A. Yost, and W. Zhu, *JHEP* **09** (2008) 133, 0808.0758.

[22] N. E. Adam, V. Halyo, and S. A. Yost, *JHEP* **05** (2008) 062, 0802.3251.

[23] P. Golonka and Z. Was, *Eur. Phys. J.* **C50** (2007) 53–62, hep-ph/0604232.

[24] R. Kleiss, *Nucl. Phys.* **B347** (1990) 67–85.

[25] Z. Was and S. Jadach, *Phys. Rev.* **D41** (1990) 1425.

[26] A. Einstein, B. Podolsky, and N. Rosen, *Phys. Rev.* **47** (1935) 777–780.

[27] ALEPH Collaboration, A. Heister *et al.*, *Eur. Phys. J.* **C20** (2001) 401–430, hep-ex/0104038.

[28] A. Andonov *et al.*, *Comput. Phys. Commun.* **181** (2010) 305–312, 0812.4207.

[29] A. Andonov *et al.*, *Comput. Phys. Commun.* **174** (2006) 481–517, hep-ph/0411186.

[30] S. Jadach and Z. Was, *Acta Phys. Polon.* **B15** (1984) 1151.

[31] S. Jadach and Z. Wąs, *Acta Phys. Polon.* **B15** (1984) 1151, Erratum: **B16** (1985) 483.

[32] T. Pierzchała, E. Richter-Wąs, Z. Wąs, and M. Worek, *Acta Phys. Polon.* **B32** (2001) 1277–1296, hep-ph/0101311.

[33] Z. Was and M. Worek, *Acta Phys. Polon.* **B33** (2002) 1875–1884, hep-ph/0202007.

[34] K. Desch, A. Imhof, Z. Was, and M. Worek, *Phys. Lett.* **B579** (2004) 157–164, hep-ph/0307331.

[35] http://root.cern.ch/root/Availability.html .

[36] LCG project http://lcg.web.cern.ch/LCG/.

[37] M. Kirsanov, A. Ribon, and O. Zenin, *PoS* **ACAT08** (2008) 114, See also: http://lcgapp.cern.ch/project/simu/generator/.

[38] GNU Autotools (autoconf, automake and libtool) http://www.gnu.org/software/.

[39] A. E. Bondar *et al.*, *Comput. Phys. Commun.* **146** (2002) 139–153, hep-ph/0201149.

[40] Z. Was and P. Golonka, *Nucl. Phys. Proc. Suppl.* **144** (2005) 88–94, hep-ph/0411377.

A Appendix: Interface to TAUOLA FORTRAN

From the point of view of a C++ interface, the τ decay library should be seen as a black-box. The code of the library is stored in the directory `tauola-fortran` and is identical to the one stored in the directory `TAUOLA`, documented in Ref. [8]. Minor adaptations which have been made affect platform-dependent files only. In the future, we expect this part of the code to be replaced with a new version based on the work proposed in [9]. This is one of the main reasons why we still keep the code in `FORTRAN`.

This section is addressed to developers of the interface, and special users interested in versions of the `TAUOLA` initialization other than the default one, `cleo`. For this purpose we describe the common blocks and routines which allow communication between `TAUOLA` and `TAUOLA C++ Interface`, even though they are explained already in Refs. [5, 6, 7, 8]. The repetition here is convenient to provide an easier explanation of the user configuration discussed in Appendix C.

A.1 Common Blocks

IDFC τ PDG id

IDFF *int* PDG id of the 'first' τ must be 15 or -15

TAUPOS Position of τ's in the event record common block

NPA *int* first τ position

NPB *int* second τ position

PARMAS Particles masses and widths

AMTAU *float* mass of τ

AMNUTA *float* mass of ν_τ

AMEL *float* mass of e

AMNUEL *float* mass of ν_e

AMMU *float* mass of μ

AMNUMU *float* mass of ν_μ

AMPIZ *float* mass of π^0

AMPI *float* mass of $\pi^\pm$

AMRO *float* mass of ρ. (As used in some but not all decay channels for parametrization of hadronic currents. The same is true for all other unstable intermediate state particles and resonances.)

GAMRO *float* width of ρ

AMA1 *float* mass of a_1

GAMA1 *float* width of a_1

AMK *float* mass of $K^\pm$

AMKZ *float* mass of K^0

AMKST *float* mass of K^*

GAMKST *float* width of K^*

JAKI Control variables for the decay channels

JAK1 *int* chosen decay channel for the first τ, if set to 0 a random choice will be made according to predefined branching ratios

JAK2 *int* chosen decay channel for the second τ, if set to 0 a random choice will be made according to predefined branching ratios

JAKP *int* used in some `FORTRAN` applications only

JAKM *int* used in some `FORTRAN` applications only

KTOM *int* used in some `FORTRAN` applications only

TAURAD Variables for QED radiative corrections in leptonic τ decay channels

ITDKRC *float* 1/0 radiative correction on/off

XK0DEC *float* minimal energy of photon to be generated (with respect to maximum possible value).

TAUBRA Variables for the composition of τ decay channels.

GAMPRT[30] *float (non-negative)* one dimensional matrix of τ decay branching ratios used if `JAK1=0` or `JAK2=0`. This can be changed by the user at any time during the generation. GAMPRT does not need to sum up to 1. The default of any entry can be changed by invoking the method `Tauola::setTauBr(int i, double br)`.

JLIST[30] *integer* one dimensional table, to be left unchanged. It is basically a `FORTRAN` emulation of a table of pointers.

NCHAN *integer* the number of τ decay channels which can be used, all higher than NCHAN values of GAMPRT are dummy.

TAUKLE Further variables for the composition of some of the τ decay sub-channels

BRA1 *float* relative branching ratio between $a_0 \to \pi^+\pi^+\pi^-$ and $a_0 \to \pi^0\pi^0\pi^+$

BRK0 *float* relative branching ratio of K_0 decay

BRK0B *float* relative branching ratio of $\bar{K}_0$ decay

BRKS *float* relative branching ratio of K^* decay

A.2 Routines

A.2.1 Important `TAUOLA FORTRAN` Routines

INIETC Initialize the content of the JAKI and TAURAD common blocks.
Return type: *void*
Parameters: *none*

INIMAS Initializes the masses stored in the common block PARMAS
Return type: *void*
Parameters: *none*

INIPHX Initializes parameters of QED, in the common block QEDPRM.
Return type: *void*
Parameters: *none*

INITDK Initializes kinematical information on τ decay channels: the branching fractions to be used for when JAK1=0 or JAK2=0, masses and flavours of scalars for each decay channel, names of the decay channels. This is well documented in the `FORTRAN` version of the program. This routine should be left unchanged. The defaults of the GAMPRT matrix residing in the common block TAUBRA, can be changed before any consecutive execution of the routine DEKAY.
Return type: *void*
Parameters: *none*

INIPHY Initializes parameters of QED, in the common block QEDPRM,
Return type: *void*
Parameters:

1. *float PI*=3.1415...
2. *float ALFINV* =$1/\alpha_{QED}$
3. *float ALFPI* =α_{QED}/π
4. *float XK0* = a dummy variable at present

DEXAY Generates a decay of a polarized τ. DEKAY is more powerful for spin effects and should be used in preference to this.
Return type: *void*
Parameters:

1. *int state* parameter defining the choice between τ^+ and τ^-. If set to 1, the decay of τ with ID=IDFF will be performed, otherwise it will be performed for a tau with ID=-IDFF. In both cases FILHEP will be invoked to store the appropriate decay products in the event record.
2. *double pol[4]* input τ polarization vector.

DEKAY Generates an unpolarized τ decay
Return type: *void*
Parameters:

1. *int state* parameter defining the choice between τ^+ and τ^-. If set to 1, the decay of a τ with ID=IDFF will be performed, for 2 it will be performed for a tau with ID=-IDFF. For 11 and 12 FILHEP will be invoked to store the appropriate decay products in the event record.
2. *double [4]* returns the vector H see Section 4

A.2.2 C++ Routines Called by `TAUOLA FORTRAN`

void filhep_(int n, int status, int pdg_id, int mother_first, int mother_last, int daughter_first, int daughter_last, float p4[4], float p_inv_mass, bool photos_flag); puts a particle into the event record. A long list of its parameters (variables named in `HEPEVT` style), is given below:

1. *n* index of the particle
2. *status* status code of the particle
3. *pdg_id* PDG id of the particle
4. *mother_first* index to the particle's first mother
5. *mother_last* index to the particle's last mother
6. *daughter_first* index to the particle's first daughter
7. *daughter_last* index to the particle's last daughter
8. *p4[4]* 4-momentum, the last component is energy
9. *p_inv_mass* mass of the particle
10. *photos_flag* should `PHOTOS` be called for this particle

void tralo4_(float * kto, float p[4], float q[4], float * ams); `FORTRAN` routine which is used to boost the four vector p[4] from the first/second τ's (`kto`=1/2) rest frame to the laboratory frame q[4]. `ams` denotes a four vector mass or virtuality.

float amas4_(float*); returns the mass of the argument four vector.

void bostr3_(float*, float*, float*); This routine performs boosting (with boost parameter given by the first argument) of a four vector (second argument) into a four vector (third argument).

B Appendix: User Guide

B.1 Installation

`Tauola C++ Interface` is distributed in a form of an archive containing source files and examples. Currently only the Linux and Mac OS[18] operating systems are supported: other systems may be supported in the future if sufficient interest is found.

The main interface library requires that `HepMC` [10] (version 2.04 or later) has been installed and its location has been provided during the configuration step. This is sufficient to compile the interface and to run the simple, standalone example.

However, in order to run the more advanced examples located in the `/examples` directory, it is required to install also:

- `ROOT` [35] version 5.18 or later
- `PYTHIA 8.1` [17] or later. `PYTHIA 8.1` must be compiled with `HepMC` 2.xx so that the `PYTHIA` library `hepmcinterface` exists.
- `MC-TESTER` [15, 16] version 1.24 or later. In `MC-TESTER` the same path to `HepMC` as in our main interface library has to be used.

[18] For this case LCG configuration scripts explained in Appendix B.2 have to be used.

In order to compile the TAUOLA C++ Interface:

- Execute ./configure with the additional command line options:

  --with-HepMC=<path> provides the path to the HepMC installation directory. One can also set the HEPMCLOCATION variable instead of using this directive. This path is required for the interface to compile.

  --prefix=<path> provides the installation path. The include and lib directories will be copied there if make install is executed later. If none has been provided, the default directory for installation is /usr/local.

- Execute make

- Optionally, execute make install to copy files to the directory provided during configuration.

After compiling the TAUOLA/tauola-fortran part, the TAUOLA C++ Interface will be compiled and the /lib and /include directories will contain the appropriate libraries and include files.

In order to compile the examples, enter the /examples directory and:

- Compilation of TAUOLA C++ Interface has to have already be completed.

- Execute ./configure to determine which examples can be compiled. Additional paths can be provided as command line options:

  --with-Pythia8=<path> provides the path to the Pythia8 installation directory. One can set the PYTHIALOCATION variable instead of using this directive. This path is required for all additional examples and tests.

  --with-MC-Tester=<path> provides the path to the MC-TESTER installation directory (the libHepMCEvent must be compiled as well, see [16] for more details). One can set the MCTESTERLOCATION variable instead of using this directive. This path is required for all additional examples and tests. This option implies that ROOT has already been installed (since it is required by MC-TESTER). The ROOT bin directory should be listed in the variable PATH and the ROOT libraries in LD_LIBRARY_PATH.

  Note that the installation of TAUOLA C++ Interface and installation of the examples can be independent. This is why, if the /examples directory is not in its default place, the following options must be provided during configuration:

  --with-Tauola=<path> provides the path to the TAUOLA libraries and include files.

  --with-HepMC=<path> provides the path to HepMC.

- execute make

If neither Pythia8 nor MC-TESTER are present, only the simple example will be provided. The /examples directory will contain the compiled example files.

B.2 LCG configuration scripts; available from version 1.0.4

For our project still another configuration/automake system was prepared for use in the LCG/Genser project[19] [36, 37].

To activate the set of autotools[38]-based installation scripts, enter the `platform` directory and execute the `use-LCG-config.sh` script there. Then, the installation procedure and the names of the configuration script parameters will differ from the ones described in our paper. The instructions given in the './INSTALL' readme file created by the `use-LCG-config.sh` script should be followed. One can also execute `./configure --help`. It will list all options available for the configuration script.

Some short information on these scripts can be found in `README` of the main directory as well.

B.3 Elementary Tests

The most basic test which should be performed is verification that the interface is installed correctly, that all τ leptons are indeed decayed by the program and that energy momentum conservation is preserved. `TAUOLA` has its own database of parameters and as a consequence the τ lepton mass may differ between the program performing a τ's production and `TAUOLA` performing its decay. This leads to the sum of τ decay product momenta not exactly matching the τ's momentum. Although this effect may seem negligible, it may break the numerical stability of programs like `PHOTOS` if they are applied later.

Once correct execution of the basic program steps have been confirmed, ie. τ leptons are decayed, energy momentum is conserved and there are no double decay occurrences in the event tree, step one of the program installation tests is completed[20].

In principle, these tests have to be performed for any new hard process and after any new installation. This is to ensure that information is passed from the event record to the interface correctly and that physics information is filled into `HepMC` in the expected manner. Misinterpretation of the event record content may result in faulty generation by `TAUOLA`. For example spin correlations may be missing or badly represented, or some τ leptons may remain undecayed.

B.4 Executing Examples

Once elementary tests are completed one can turn to the more advanced ones. The purpose is not only to validate the installation but to demonstrate how the interface can be used and how spin affects some distributions.

The examples can be run by executing the appropriate `.exe` file in the `/examples` directory. In order to run some more specific tests for spin effects and decays of the following intermediate states: Z, W, H, $H^{\pm}$, the main programs residing in subdirectories of the same name placed

[19] We have used the expertise and advice of Dmitri Konstantinov and Oleg Zenin in the organization of configuration scripts for our whole distribution tar-ball as well.

[20] We have performed such tests for all choices of the `HepMC` event record obtained from `PYTHIA 8.1` processes and listed later in the paper. Further options for initializations (parton shower hadronization or QED bremsstrahlung on/off etc.) were studied. This installation step was a necessary one of program development as well.

in the /examples/testing directory should be executed. For tests of all τ decay modes, the directory /examples/testing/tau is prepared. In all cases the following actions have to be performed:

- Compile TAUOLA C++ Interface as well as the examples. Note that paths both to Pythia8 and MC-TESTER must be provided.
- Check that the appropriate system variables are set: normally set by the script /configure.paths.sh (the configuration step mentions this script).
- Enter the /examples/testing directory and execute make. Modify test.inc if needed.
- Enter the chosen directory and execute make.

The appropriate .root files as well as .pdf files generated by MC-TESTER will be created inside the chosen directory. One can execute 'make clobber' to clean the directory. One can also execute 'make run' inside the /examples/testing directory to run all available tests one after another. New source code changes can easily be validated in this way. Tests are run using examples/testing/tauola_test.exe and booklets will be produced with comparisons to the benchmark files.

A set of benchmark MC-TESTER root files are packed with the interface distribution in the subdirectories of examples/testing/. They can be used as examples to start new work or simply to construct comparison plots to validate new versions or new installations of TAUOLA Interface.

In Appendix C possible modifications to the examples settings are discussed. This may be interesting as an initial step for physics studies users. Numerical results of some of these tests are collected in Section 6 and can be thus reproduced by the user.

B.4.1 Monitoring τ Decay Channels

It is important to check, if the τ decays themselves, are generated correctly on a user's platform. For that purpose, our last demo (directory /examples/testing/tau) is prepared. If the test is activated, the user performs a standard MC-TESTER comparison of his program execution with the pre-generated one (of 10 million events). In this case all τ decay modes are activated and MC-TESTER simply analyzes the τ decays themselves.

B.5 Library Linking

In order to link the libraries to a user's project, both the static libraries and shared objects are constructed. To use TAUOLA FORTRAN and TAUOLA Interface in an external project, additional compilation directives are required. For the static libraries:

- add -I<TauolaLocation>/include at the compilation step,
- add <TauolaLocation>/lib/libTauolaCxxInterface.a and <TauolaLocation>/lib/libTauolaFortran.a at the linking step.

For the shared objects:

- add -I<TauolaLocation>/include at the compilation step,
- add -L<TauolaLocation>/lib along with -lTauolaCxxInterface -lTauolaFortran at the linking step.
- TAUOLA libraries must be provided for the executable; eg. with the help of LD_LIBRARY_PATH.

<TauolaLocation> denotes the path to the TAUOLA installation directory.

B.6 Known Issues

We list here difficulties we've encountered during the testing phase and during installation for particular configurations.

The first problem occurs if a user incorrectly configures the units of PYTHIA to be different from the units in HepMC. (For example: if PYTHIA produces output in MeV, while HepMC interprets input as being in GeV). In this case, the built-in routines of TAUOLA Interface will treat the input as being in GeV and will adapt its output to those units as well. If this kind of situation occurs (the user will be notified by many warnings of four-vector momentum not being conserved), one can force HepMC to use MeV units just before filling it with the PYTHIA event record data. The HepMC event will be automatically converted to GeV when TAUOLA Interface is called.

Another example of a known compatibility issue arose because of a difference between the assumed default HepMC version 2.05 and version 2.03 (currently used, for example, in Athena[21]). In this case, the script platform/to-HepMC-2.03.sh will be automatically invoked during the configuration step. However, in version 2.03 methods for unit conversion are absent, therefore GeV and mm will be expected for input and used for output. The method Tauola::setUnits(...), described in Appendix C.12, becomes dummy.

At present, modification to our TAUOLA C++ Interface has to be introduced eg. for use in the Athena system of the ATLAS collaboration software. This is to allow for backwards compatibility with older versions of HepMC and to prevent name clashes with the old TAUOLA FORTRAN Interface in environments where both interfaces are loaded concurrently. On the other hand, there is no problem with the library /lib/libTauolaFortran.a, the main part of the TAUOLA FORTRAN code itself. The version used by Athena can be loaded instead of ours. In Athena, the binp variant of the cleo initialization is used for TAUOLA; in this variant, for the 4π decay modes of τ's, parametrization based on Novosibirsk data is used [39].

All necessary changes for our TAUOLA C++ Interface can be introduced with use of the script platform/to-Athena.sh. It can be invoked by executing the make athena command in the main directory. Recompilation of the interface must then be performed.

C Appendix: User Configuration

In this section we give a description of how the user can configure TAUOLA FORTRAN and the TAUOLA Interface. All configuration is done via the static class Tauola. Below is the

[21] Software framework of the ATLAS collaboration.

complete list of user configurable parameters and basic information on their meaning.

C.1 Spin Correlation

By default, all spin correlations are turned on. However one may be interested to partially or completely switch off their effects for the sake of numerical experiments which validate whether a measurement will be sensitive to certain spin correlation components. This technique may be useful to evaluate the significance of spin correlations for signal/background separation as well.

Several partial treatments of spin correlations are possible. In general, the most complete intervention is to simply rewrite the matrix R_{ij} for the particular channel. The following methods are nonetheless provided:

- `Tauola::spin_correlation.setAll(bool flag)`
  Turns all spin correlation computations on or off depending on the flag, which can be either **true** or **false**. Note: this should be called after Tauola::initialize().
- `Tauola::spin_correlation.<tau parent>=flag`
  Turns particular spin correlation computation on or off for a given τ parent depending on the flag which can be either **true** or **false**.

  Implementation of this switch is provided for: **<tau parent>=GAMMA, Z0, HIGGS, HIGGS_H, HIGGS_A, HIGGS_PLUS, HIGGS_MINUS, W_PLUS, W_MINUS**. The keywords denotes the τ parent.

Example:

```
Tauola::spin_correlation.setAll(false);
Tauola::spin_correlation.HIGGS=true;
```
Turns all spin correlations off, except HIGGS.

Finally one can replace the density matrix following the description given in Appendix D.3. In this case one does not need to recompile the code.

C.2 Decay Mode Selection

By default, all τ decay modes will be generated according to predefined branching fractions. Methods to modify the default values are provided:

- `Tauola::setSameParticleDecayMode(int mode)`
  Set the decay mode of the τ with the same PGD code as set in Tauola::setDecayingParticle() (by default this sets the decay mode of τ^-).
- `Tauola::setOppositeParticleDecayMode(int mode)`
  Set the decay mode of the τ with the opposite PGD code as set in Tauola::setDecayingParticle() (by default this sets the decay mode of τ^+).

Example:
`Tauola::setSameParticleDecayMode(Tauola::PionMode);`
`Tauola::setOppositeParticleDecayMode(4);`
Forces only the modes $\tau^- \to \pi^-\nu_\tau$ *and* $\tau^+ \to \rho^+\nu_\tau(\rho^+ \to \pi^+\pi^0)$ *to be generated*

- `Tauola::setTauBr(int mode, double br)`
  Change the τ branching ratio for the channel *mode* from default to *br*. Note: this should be called after Tauola::initialize().
  Example:
  `Tauola::setTauBr(3, 2.5);`
  Sets the rate for the channel $\tau^\pm \to \pi^\pm\nu_\tau$ *to 2.5. Arbitrary, but consistent, units for all channels may be used. Normalization will be performed anyway.*

- The `int mode` enumerators which are arguments of `setOppositeParticleDecayMode`, `setSameParticleDecayMode`, `setTauBr` have the following meaning:
  - 0 - `Tauola::All` - All modes switched on
  - 1 - `Tauola::ElectronMode` - $\tau^\pm \to e^\pm\nu_\tau\nu_e$
  - 2 - `Tauola::MuonMode` - $\tau^\pm \to \mu^\pm\nu_\tau\nu_\mu$
  - 3 - `Tauola::PionMode` - $\tau^\pm \to \pi^\pm\nu$
  - 4 - `Tauola::RhoMode` - $\tau^\pm \to \rho^\pm\nu$
  - 5 - `Tauola::A1Mode` - $\tau^\pm \to A_1^\pm\nu$
  - 6 - `Tauola::KMode` - $\tau^\pm \to K^\pm\nu$
  - 7 - `Tauola::KStarMode` - $\tau^\pm \to K^{*\pm}\nu$
  - 8 - $\tau^\pm \to 2\pi^\pm\pi^\mp\pi^0\nu$
  - 9 - $\tau^\pm \to 3\pi^0\pi^\pm\nu$
  - 10 - $\tau^\pm \to 2\pi^\pm\pi^\mp 2\pi^0\nu$
  - 11 - $\tau^\pm \to 3\pi^\pm 2\pi^\mp\nu$
  - 12 - $\tau^\pm \to 3\pi^\pm 2\pi^\mp\pi^0\nu$
  - 13 - $\tau^\pm \to 2\pi^\pm\pi^\mp 3\pi^0\nu$
  - 14 - $\tau^\pm \to K^\pm K^\mp\pi^\pm\nu$
  - 15 - $\tau^\pm \to K^0\bar{K^0}\pi^\pm\nu$
  - 16 - $\tau^\pm \to K^\pm K^0\pi^0\nu$
  - 17 - $\tau^\pm \to 2\pi^0 K^\pm\nu$
  - 18 - $\tau^\pm \to \pi^\pm\pi^\mp K^\pm\nu$
  - 19 - $\tau^\pm \to \pi^\pm\pi^0\bar{K^0}\nu$
  - 20 - $\tau^\pm \to \eta\pi^\pm\pi^0\nu$
  - 21 - $\tau^\pm \to \pi^\pm\pi^0\gamma\nu$
  - 22 - $\tau^\pm \to K^\pm K^0\nu$

- `Tauola::setTaukle(double bra1, double brk0, double brk0b, double brks)`
  Change the τ sub-channels branching ratio between (i) $a_0 \to \pi^+\pi^+\pi^-$ and $a_0 \to \pi^0\pi^0\pi^+$ (ii) subchannels of K_0 (iii) subchannels of $\bar{K}_0$ and (iv) subchannels of K^*. Note: this should be called after Tauola::initialize().

  Example:
  `Tauola::setTaukle(0.5, 0.5, 0.5, 0.6667);`

Set the parameters to their default values

C.3 Decaying Particle

The following method is prepared to impose the sign for the 'first τ', that is to allow reversal of the signs of the `SameParticle` and `OppositeParticle` τ:

- `Tauola::setDecayingParticle(int pdg_id)`
  Set the PDG id of the particle which `TAUOLA` should decay as 'first τ'. Both particles with pdg_id and -1*pdg_id will be decayed. The default is 15, but one may want to use -15 for special applications.

  Example:
  `Tauola::setDecayingParticle(-15);`
  Set `SameParticle` τ *to be* τ^+

C.4 Radiative Corrections

The user may want to configure parameters used in the generation of QED corrections in the leptonic decay channels of τ's. For that purpose the following methods are provided:

- `Tauola::setRadiation(bool switch)`
  Radiative corrections for leptonic τ decays may be switched on or off by setting the switch to **true** or **false** respectively. By default this is **true**[22].

  Example:
  `Tauola::setRadiation(false);`
  Switch radiative corrections off in τ *decays*

- `Tauola::setRadiationCutOff(double cut_off)`
  Set the cut-off for radiative corrections of τ decays. The default of 0.01 means that only a photon of energy (in its rest frame) above 0.01 of half of the decaying particle mass will be explicitly generated.

[22] Only in the case of leptonic τ decays can radiative corrections be generated in `TAUOLA` [6]. The algorithm relies on the first order complete matrix element and no exponentiation is available. If the multiple photon option is requested or if radiative corrections for other decay channels are needed `PHOTOS` Monte Carlo can be used instead [13]. In [40] it was shown, that the numerical effects due to the parts not included in `PHOTOS` of the first order matrix element is numerically more significant than multiple photon effects. This conclusion is based on our standard numerical tests and will not necessarily be the case for other applications.

C.5 Decay of Final State Scalars

In some cases a user may want TAUOLA to decay short living scalar particles produced in $\tau^{\pm}$ decays, rather than invoking a host generator for the post processing step. For that purpose a special algorithm is prepared, even though high precision is then not assured. This might not be a problem if the algorithm is used for τ decays only where events with such decays are rather rare:

- `Tauola::setEtaK0sPi(int a, int b, int c)`
  The three parameters a, b and c switch on or off the decay of η, K_s^0 and π^0 respectively. A value of 1 is on and 0 is off.

  Example:
  `Tauola::setEtaK0sPi(1,0,1);`
  In the event branch starting from a τ, η and π^0 decay, but K_s^0 remains undecayed.

C.6 Scalar-Pseudoscalar Higgs

Users may wish to study spin correlations in processes involving scalar, pseudoscalar or mixed scalar-pseduoscalar decays into τ's. All options are supported by this interface. The spin density matrix will be calculated correctly for scalar Higgs (assumed PDG id of 25) and for pseudoscalar Higgs (assumed PDG id of 36) without any additional user configuration. For other cases, such as a mixed scalar-pseduoscalar Higgs or the decay of non-Higgs scalar particles, the following methods are provided:

- `Tauola::setHiggsScalarPseudoscalarMixingPDG(int pdg_code)`
  The PDG Monte-Carlo code of the Higgs which should be treated by the interface as a scalar-pseudosclar mix. The default value is PDG id 35. Please note that if *pdg_code* is set to the value of an existing spin case (eg. 25, the regular scalar Higgs) the scalar-pseudoscalar case will be assumed.
- `Tauola::setHiggsScalarPseudoscalarMixingAngle(double angle)`
  The scalar-pseudoscalar mixing angle. ie. ϕ in the coupling: $\bar{\tau}(cos(\phi) + isin(\phi)\gamma_5)\tau$. By default $\phi = \frac{\pi}{4}$.
- **Examples:**
  `Tauola::setHiggsScalarPseudoscalarMixingPDG(24);`
  `Tauola::setHiggsScalarPseudoscalarMixingAngle(3.1415/3.0);`
  Spin correlations will be calculated for the Higgs boson as though it is a scalar-pseduoscalar with mixing angle of $\frac{\pi}{3}$

C.7 Helicity States and Electroweak Correcting Weight

Independent of the generation process, the information on helicities of τ^+ and τ^- can be returned[23] with the help of accessors:

[23] One has to be careful because the actual sign may depend on the process and boosting routine.

- `int Tauola::getHelPlus()`
- `int Tauola::getHelMinus()`

Note that these helicities are not used in the interface and carry approximate information only.

The electroweak weight can be returned with the help of accessors:

- `double Tauola::getEWwt()` - for cross section with electroweak corrections included
- `double Tauola::getEWwt0()` - for cross section at born level

These methods provide information once processing of a given event is completed.

C.8 Redefine on flight τ decay channels

In section C.2 it is explained how the branching ratios for tau decay channels are initialized. Since `TAUOLA` version 1.0.2, the appropriate constants can be modified at any moment of the program execution. For that purpose the user can define the functions

```
void redPlus(TauolaParticle *tau)
void redMinus(TauolaParticle *tau)
```

The `TauolaParticle` pointer can be used, for example, to make a choice of decay channels dependent on the flavour of the τ's mother. The functions will be executed respectively by `TAUOLA Interface` prior to each consecutive τ^+ or τ^- decay, if in the initialization

```
Tauola::setRedefineTauPlus(redPlus);
Tauola::setRedefineTauPlus(redMinus);
```

commands are present. An example with explanatory details is given in `taumain_pythia_example.c`.

C.9 Use of the `TAUOLA decayOne` method

In Section 3.3 an algorithm to decay all τ leptons present in the event record is explained. For that purpose the `decayTaus()` method is provided. To decay a single τ lepton in a way independent of the event record content another, simple method is provided. Obvious examples when it can be useful, are processes where the hard matrix element originates from models of new physics, and different flavours of such models are to be tested. In such cases, universal methods of finding spin states of the τ to be decayed may not exist. Depending of the precision required one may need to: decay a τ without taking into account its spin state, impose its individual spin state as input information or provide a method which can be used for full density matrix generation. In the last case control over Lorentz transformations between the τ rest-frame and laboratory frame have to be available for the user.

Fortunately for all these applications a rather simple method is sufficient. It can be used to generate a decay of an individual τ, without information on its parents.

- `Tauola::decayOne(`
  `TauolaParticle *tau, bool undecay, double polx, double poly, double polz)`

  The main routine for decaying a tau. Only the first parameter is mandatory. The first parameter is a pointer to the τ that needs to be decayed.
  The `undecay` flag determines the reaction that should be taken if the τ already has daughters. By default the flag is set to **false**, which means that an already decayed τ will be left unchanged. Setting this flag to **true** allows the interface to first undecay the τ and replace it with a new decay.

  The last three parameters are the components of the τ polarization 3-vector[24]. In the case of `TAUOLA decayOne`, the decayed τ is treated as a standalone particle, without considering its mothers, daughters or siblings. In case the user wants to input the polarization vector (as a default, the τ is treated as not polarized), the last three parameters have to be used.

- `Tauola::setBoostRoutine(`
  `void (*boost)(TauolaParticle *tau, TauolaParticle *target) )`
  Once executed, `Tauola:decayOne` will use the user function instead of the default one, to boost τ decay products from their rest frame to the lab frame. This feature may be essential, in future, for the use of `Tauola::decayOne` as part of a user algorithm for the generation of exact spin correlations in multi τ final states.

The `single_tau_gun_example.c` is provided in the directory `/examples`. If the polarization `polx=0, poly=0, polz=1` is chosen, then the helicity state is taken: left handed τ^- or right handed τ^+. If, again as given in the example, `Tauola::setBoostRoutine` is used with the proposed method, then `polx=0, poly=0, polz=1` will not mean helicity state, but rather the τ spin polarization vector oriented along the z axis of the lab frame (in fact along its space component in the τ rest-frame). Obviously a spin effect chosen this way, will depend on the direction of the τ momentum.

C.10 Logging and Debugging

This section describes the basic functionality of the logging and debugging tool. For details on its content we address the reader to comments in the `/src/utilities/Log.h` header file.

Let us present however some general scheme of the tool's functionality. `TAUOLA Interface` allows filtering out some amount of data displayed during the program run and provides a basic tool for memory leak tracking. The following functions can be used from within the user program after including the `Log.h` file:

- `Log::Summary()` - Displays a summary of all messages.
- `Log::SummaryAtExit()` - Displays the summary at the end of a program run.
- `Log::LogInfo(bool flag)`
  `Log::LogWarning(bool flag)`
  `Log::LogError(bool flag)`
  `Log::LogDebug(int s, int e)`

[24]Note that the τ^- polarization 3-vector (0,0,1) will mean left-handed τ^- and right-handed τ^+ (if the default boosting routine is used).

`Log::LogAll(bool flag)`
Turns logging of info, warning, error and debug messages on and off depending on the flag being true or false. In the case of debug messages - the range of codes to be displayed must be provided. By default, only debug messages (from 0 to 65535) are turned off. If the range is negative ($s > e$) the debug messages won't be displayed. The last option turns displaying all of the above messages on and off.

From program version 1.0.2, a new option, `Log::SetWarningLimit(int limit)`, is available. Only the first `'limit'` warnings are then displayed. The default for `limit` is 100. If `limit`=0 is chosen then no limits on the warnings displayed will be set.

The memory leak tracking function allows checking of whether all memory allocated within `TAUOLA Interface` is properly released. However, using the debug option significantly increases the amount of time needed for each run. Its use is therefore recommended for debugging purposes only. In order to use this option, modify `make.inc` in the main directory by adding the line:

```
DEBUG = -D"_LOG_DEBUG_MODE_"
```

Recompile the interface. Now, whenever the program is executed a table will be printed at the end of the run, listing all pointers that were not freed, along with the memory they consumed. If the interface works correctly without any memory leaks, one should get an empty table.

It is possible to use this tool within a user's program, however there are a few limitations. The debugging macro from "Log.h" can create compilation errors if one compiles it along with software which has its own memory management system (e.g. `ROOT`). To make the macro work within a user's program, ensure that `Log.h` is the last header file included in the main program. It is enough to compile the program with the `-D"_LOG_DEBUG_MODE_"` directive added, or `#define _LOG_DEBUG_MODE_` placed within the program before include of the `Log.h` file[25].

C.11 Plots for Debugging and Monitoring

This section describes the basic functionality of the plotting tool. Detailed explanations are given in the `/src/utilities/Plot.h` and `/src/utilities/Plot.cxx` files.

The `Plot` class allows generation of data for several plots we use to monitor the interface. At present, τ polarization, as taken from the `SANC` library and used by the interface (including its interpolation algorithm) is monitored in this way. The program generates data files during execution to be used later for graphic output. This code is not expected to be of a large interest to users. It is mainly for testing and debugging purposes, but may be of interest for installation tests as well.

Since version 1.0.3, this class has been simplified. Possible future extensions were kept in mind. In order to generate input data for plotting, a few methods have been provided which can be accessed after adding `#include "Plots.h"` in the main program:

- The following public methods are now available: `Plots::SANCtest1()`

[25] Note that `Log.h` does not need to be included within the user program for the memory leak tracking tool to be used only for `TAUOLA Interface`.

`Plots::SANCtest2()`
`Plots::SANCtest3()`
`Plots::SANCtest4()`
Each one runs the appropriate test for `SANC` tables. All or only some of these tests can be run at a time. Note that our Figures 3, 4, 5(a) and 5(b) can be reproduced with these methods.

- `Plots::setSancVariables(int flavour, double cosTheta)` - sets the arguments of the first two tests: the flavor of the incoming particle (the default is 1), and scattering angle $\cos(\theta)$ (the default is -0.2).

A source file `/examples/testing/EW-PLOTS/plots.c` has been provided which runs all available tests. It is compiled when `make` is executed in the `examples/testing` directory. The files generated with this tool can be used to make plots with external scripts; the `/examples/testing/EW-PLOTS/draw.C ROOT` script has been provided. If `root draw.C` is executed, it first checks, by name, which input data files exist and then corresponding plots are drawn. Since the generated files contain the test data only, without much explanation of their meaning, to interpret them one needs to look into the `Plot` class source files for a description.

C.12 Other User Configuration Methods

The following auxiliary methods are prepared. They are useful for initialization or are introduced for backward compatibility.

- `Tauola::setUnits(Tauola::MomentumUnits,Tauola::LengthUnits)`
  Set output units for momentum (Tauola::GEV / Tauola::MEV) and decay vertex position (Tauola::MM / Tauola::CM). Methods are only available for `HepMC 2.04` or higher.

- `Tauola::setTauLifetime(double lifetime)`
  Set the mean τ lepton lifetime in mm, `lifetime=0.08711`. If the user wants a vertex position to be generated by his own method, then the numerical value of the τ lifetime should be set to 0.0 here.

- `Tauola::setInitializePhy(double iniphy_param)`
  Initializes some constants related to QED corrections. The variable `iniphy_param` is at present a dummy variable. It is prepared for transmission to some old style production code and is kept for backward compatibility.

- `Tauola::setRandomGenerator(double (*gen)())`
  In `tauola-fortran` the random number generator `RANMAR` is used. It is also used in our auxiliary methods which temporarily remain in `FORTRAN`. `RANMAR` may need to be replaced or a particular seed may need to be chosen. It can be easily done and is explained in [8]. In the C++ part of the code a user can simply set the pointer to the replacement for an internal random number generator `Tauola::RandomDouble`. The generator must return a `double` between 0 and 1. `Tauola::setRandomGenerator(NULL)` will reset the program back to the default generator.

D Appendix: Modifying Electroweak Corrections Module

D.1 SANC Unit Initialization and Input Parameters

In this section we describe details of the `SANC` library initialization. Input parameters and constants are collected in several files located in the directory `/SANC`. The file `s2n_declare.h` contains a declaration of all `FORTRAN` variables used by the `SANC` NLO Library. The particle masses and coupling constants are initialized by the `sanc_input.h` header file. It is called by the `s2n_init` subroutine (see file `s2n_init.f`), which initializes other parameters like particle mass ratios, particle charges and weak isospin projection, as well as the value of the fictitious photon mass (`thmu2`) used by IR singularity regularization and the soft/hard radiation separator (`omega`). Several user controlled flags are defined:

- **iqed** = 1/0 – NLO QED correction is switched on/off. Default **iqed** = 0
- **iew** = 1/0 – NLO EW correction is switched on/off. Default **iew** = 1
- **iborn** = 0/1 – NLO correction are switched on/off; Default **iborn** = 0
- **gfscheme** = 1/0 – calculation schemes: 1 - Fermi Scheme, 0 - Alpha Scheme (default).

These flags are used to configure table preparation by the program `SANC/SANCinterface.cxx`. For the `SANC` module structure and its project details, see Refs. [28, 29].

Together with the program, the pre-calculated tables are distributed. They are supposed to be used for the implementation of spin effects and the density of tabulated points is rather low. For precision applications, such as Z/W line shape studies, further work is needed and we leave it for a future publication.

D.2 Structure of Files with Pretabulated R_{ij}

In order to generate all pretabulated files the `SANC/SANCinterface.cxx` program is used. When compiled along with the `SANC` library, the `SANC FORTRAN Interface` and the modules located inside the `SANC/modules` directory, the program generates two files - `table1-1.txt` and `table2-2.txt` for the down and up quarks respectively. The program is invoked with the command `make tables` from the directory `SANC`. A third file, `table11-11.txt`, representing tabulated results for electron beams will not be generated automatically.

The structure of each generated file consists of several blocks:

- Initialization block

  Dimensions - NS1, NS2, NS3 and NCOS values used as dimensions of the generated tables

  Ranges - minimum and maximum values for all three pretabulation ranges of s

- Information block:

  Date and time of the generation

Full path of the generating program

`SANC` library information block

`SANC` initialization parameters list

- Data block:

  BeginRange1 statement

  tables of NS1 * NCOS lines for first range

  BeginRange2 statement

  tables of NS2 * NCOS lines for second range

  BeginRange3 statement

  tables of NS3 * NCOS lines for third range

  End statement

Lines within **Data block** consist of 4*4 numbers for R_{ij}, two extra numbers for the electroweak weight and an endline character. Statements used within **Data block** (**BeginRange1, BeginRange2, BeginRange3, End**) are mandatory and must be present in exact form. They mark the beginning and end of the appropriate data set. The program also checks whether the data block has been read completely and verifies if variables read at initialization were consistent.

D.3 Importing SANC Tables into `TAUOLA Interface`

The three files generated by the `SANC` module are loaded into `TAUOLA Interface` during the initialization step, if they are located in the directory of the main program. When `Tauola::initialize()` is called, the interface searches for the appropriate files and if they are found - attempts to import the data.

For a file to be loaded correctly, the dimensions of the input file must match the interface settings. The content and size of the information block is arbitrary. The information of this block will be printed only, but not used otherwise. After reading all the tables from one file, `TAUOLA Interface` checks if the end of the data block has been reached and eventually proceeds to the next file.

If the dimensions do not match, the file is inconsistent with the structure (the end of the data block has not been reached or the file has insufficient data) - it will not be used by `TAUOLA Interface` even if the file was found. In that case the default density matrix will be used. `TAUOLA Interface` will attempt to read all of the three files, but if either one is incorrect or missing, only the data from those files that have been loaded correctly will be used.

If the need arises to modify the default tables distributed with `TAUOLA`, the `SANC` folder includes all routines needed to generate new tables along with a `Makefile` with a few options, including:

- `make` - used to recompile the library, modules and LoopTools needed for generation.
- `make clobber` - might be needed to remove the previous compilation.
- `make tables` - used to compile the interface code and generate the tables.

The C++ interface to the `SANC` library is located in the `SANC/` folder. `SANCinterface.cxx` represents the main program with options setting for table making. Options for table layout are explained in comments within the file. The interface will be recompiled every time `make tables` is used. If needed, further program options, such as initialization of electroweak coupling constants or particle masses for the calculation of electroweak corrections can be modified. The `SANC/SANCtable.cxx` file consists of routines for the actual calculation of table entries from spin amplitudes calculated from `SANC`. Changes which may be of interest to advanced users should be done in this file.